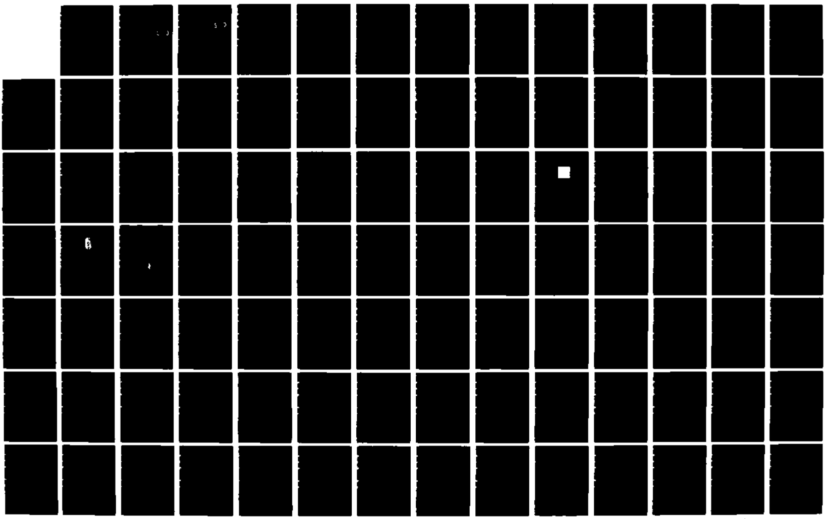
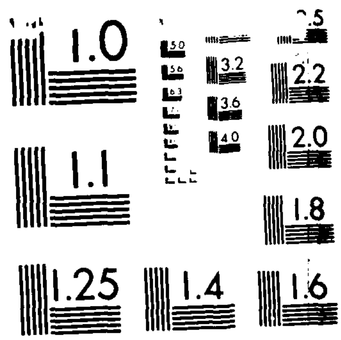
AD-A164 036 GYRO AND ACCELEROMETER BASED NAVIGATION SYSTEM FOR A MOBILE AUTONOMOUS RO.. (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. 1/3

UNCLASSIFIED R J BLOOM ET AL. 02 DEC 85 F/G 17/7 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

AD-A164 036

DTIC FILE COPY

DTIC
ELECTE
FEB 1 3 1986
S D
D

GYRO AND ACCELEROMETER BASED
NAVIGATION SYSTEM
FOR A MOBILE AUTONOMOUS ROBOT

THESIS

Roland J. Bloom
Captain, USAF

William J. Ramey, Jr.
Captain, USAF

AFIT/GA/GE/ENG/85D-33

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GA/GE/ENG/85D-33

GYRO AND ACCELEROMETER BASED NAVIGATION SYSTEM FOR A MOBILE AUTONOMOUS ROBOT

THESIS

Roland J. Bloom
Captain, USAF

William J. Ramey, Jr.
Captain, USAF

AFIT/GA/GE/ENG/85D-33

AFIT/GA/GE/ENG/85D-33

GYRO AND ACCELEROMETER BASED NAVIGATION SYSTEM FOR A MOBILE AUTONOMOUS ROBOT

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Astronautical Engineering

and Electrical Engineering, Respectively

Roland J. Bloom, B.S.A.E
Captain, USAF

William J. Ramey Jr., B.S.E.E.
Captain, USAF

December 1985

Approved for public release; distribution unlimited

Preface

It is only a matter of time until the autonomous mobile robot becomes a reality. The key to achieving this goal lies in the development of a navigation system capable of accurate position determination and intelligent, efficient, collision-free, path planning through the robot's environment. Hopefully, our efforts have provided some advancement toward creating such a robot navigation system.

The success of this thesis was a direct result of the endless support provided by several individuals and organizations. A special thanks goes to our thesis advisor Dr. Matthew Kabrisky for having the confidence to turn us loose on this project. Additionally, we would like to thank our sponsor, Tim Anderson of AFMRL; Robert Durham, Orville Wright, Dick Wager, and Stan Bashore of AFIT/ENG; Carl Short and Ron Ruley of AFIT/RMF; Mrs. Allis Moore of AFWAL; Allen Cooper, Berny Swagert, and George Kelsh of King Radio Corporation; Diane White, Ed Freedman, and Cal Watson of Analog Devices Incorporated; Bill Lee and Roger Heilman of Sundstrand Data Control Incorporated; and a generous thanks to King Radio Corporation for their donation of equipment to the project.

We would also like to express our appreciation to Tom Clifford and Bert Schneider for creating the MARRS-1 robot and putting together the robot laboratory. Their efforts made our endeavor possible. In addition, we owe Tom

Clifford a special thanks for his assistance and advice throughout the project.

We would also like to extend our gratitude to Lt. Col. Dan Biezad for his tireless assistance from the very beginning. To him we credit the aquisition of our directional gyro system.

Most of all we wish to acknowledge the many sacrifices made by our wives and children. Without their patience, love, and understanding we could not have completed this thesis.

Contents

List of Figures

Abstract

A navigation system for a mobile autonomous robot is presented. The navigation system is based upon a directional gyroscope and a single axis accelerometer which enables a robot to navigate independent of wheel optical shaft encoders and other commonly used positioning apparatus. The computer controlled navigation system is capable of providing absolute heading, heading rate (angular velocity), and linear velocity to a user computer. These data from the navigation system (heading and velocity) are used to compute the present location of the robot. In addition, the heading data is used to form a closed loop feedback control system for maintaining the robot on a desired course. The navigation system was designed specifically for application on an existing Air Force Institute of Technology (AFIT) robot; however, it could be easily adapted to any robot system with a standard IEEE RS-232 serial communication interface. Test results are provided which demonstrate the use of closed loop heading control on the AFIT robot and which identify problems associated with the use of an accelerometer system for distance measurement. This thesis includes all schematics, parts lists, software listings, and operating instructions for the navigation system. A new robot world modeling and path planning technique is also presented.

GYRO AND ACCELEROMETER BASED NAVIGATION SYSTEM FOR A MOBILE AUTONOMOUS ROBOT

I. Introduction

BACKGROUND

In general, robots fall into three broad categories: fixed arm robots, mobile robots, and mobile autonomous robots. It is important to understand how these three classes of robots differ.

A fixed arm robot is a machine modeled after the human arm and hand. It is capable of armlike movement and has a handlike manipulator, but by itself is incapable of movement from one location to another. Fixed arm robots are by far the largest category of present day robots, encompassing virtually all of the currently available robots. They are used primarily for repetitive manipulative tasks such as industrial assembly line work. Because these robots do not in general possess the ability to move themselves about their environment, they will not be discussed further.

A mobile robot is "a robot mounted on a moveable platform" [11:17]. They are distinguished by their ability to move freely about their environment, but with command and control provided external to the robot. Underwater salvage robots provide an excellent example of mobile robotics. They can move freely about their environment, but are controlled from a surface vessel by a team of human operators. Since mobile robots are not self controlled, they too will not be discussed further.

A mobile autonomous robot is "a robot acting independently and of its own volition" [11:17]. They are distinguished by their ability to move freely about their environment completely under internal control independent of external machine or human assistance. Mobile autonomous robots have many potential applications, but currently they do not exist outside of research labs. Expansion of this concept is the basis of this thesis.

In recent years, there has been a substantial increase of interest in mobile autonomous robots. A report regarding the First World Conference on Robotics Research, held in early 1985, noted that one of the most actively researched fields was mobile autonomous robots [11:17]. This surge of interest is a direct result of the "microelectronics revolution" which has resulted in today's microprocessors and digital integrated circuits (IC's). Tremendous computational power is now available in very small packages making the internal control required for a mobile autonomous robot feasible.

The potential to develop a mobile autonomous robot has generated many possible applications. Chief among these is the performance of tasks hazardous to human personnel such as fire fighting, bomb disposal, nuclear waste disposal, underwater salvage and repair, deep sea exploration, chemical production, mining, sentry duty, and military reconaissance and attack missions.

This effort is being led by the Department of Defense (DOD). The Defense Advanced Research Projects Agency (DARPA) currently has a 17 million dollar contract with Martin Marrietta Corporation to develop an autonomous land vehicle [21] and has asked General Dynamics and others for proposals on battlefield robots [13:48]. DARPA is also funding multi-legged mobile robotics research at Ohio State University [13:48].

The Army's General Officer Steering Committee for Artificial Intelligence and Robotics recently issued a report outlining near-term (1980's) robotic systems applications [14]. They include: a light weight robotic vehicle to mount antitank guided missiles, mortars, and small arms; a robotic obstacle/mine breaching tank; a robotic transport and resupply vehicle; a security sentry robot; an explosive ordnance disposal robot; and a robot scout vehicle.

The Air Force Medical Research Laboratory at Wright-Patterson Air Force Base (WPAFB) is currently working on the development of a mobile autonomous robot to service aircraft in a nuclear, biological, or chemical (NBC) contaminated environment. This thesis is an extension of that effort.

All of the above applications require the robot to be able to perform three primary functions: computation, manipulation, and navigation.

Computation is the robot's ability to make job application related control decisions. For example, an aircraft servicing robot would have to decide which subsystems to test and then which part to replace. In the field of artificial intelligence, this would be called an expert system. Since numerous expert systems of the type required for robotics have already been developed, this function will not be discussed further.

Manipulation is the robot's ability to skillfully handle and move objects in its environment. This function has been well developed and is commercially available in the form of a fixed arm robot. Therefore this function also will not be discussed further.

Navigation is the robot's ability to direct itself toward some destination. This problem has yet to be adequately addressed and is the major impediment to the development of a mobile autonomous robot. Robot navigation is the focus of this thesis.

Navigation requires that the robot be able to follow a given or calculated course of travel making course corrections as necessary. This means the robot must be capable of finding its current location in reference to the desired course. This information is required for steering control feedback in order to accurately follow a given course.

In addition, the robot must know its starting location and final destination on the same frame of reference as its current location. This will give it the ability to know when is has arrived at the desired location.

Finally, a navigation system must provide a means for recognizing obstacles, both known and unknown. It must allow for course changes to avoid these obstacles and yet still arrive at the destination in a time efficient manner. A robot must be able to deal with a dynamic environment.

Physically, the navigation system is composed of sensors and steering control. The sensor subsystems collect position and obstacle data. the steering control subsystem uses the position and obstacle data to make any necessary course changes. All but the most simple steering control subsystems require many intelligent decisions to be made and therefore require a computer as the controller.

Past research work on mobile robot and mobile autonomous robot navigation has produced vision, embedded wire, beacon, shaft encoder, and sonar based navigation systems, where the environmental data from each system's sensors is interpreted by a computer to control the path of travel.

Vision navigation systems use television cameras to "see" the desired path and obstacles much like a human. Vision systems today are relatively crude in spite of modern advances in high speed computer systems. They are not capable of interpreting the digital video information fast enough to allow real time navigation.

Dr. Hans Moravec has done considerable research in the area of robotic vision. During his Ph.D. work at Stanford University, he experimented with the "Stanford cart", a mobile robot which he equipped with a video camera [17]. The cart's ultimate achievement was travelling through an obstacle laden area to a goal about 60 feet away. The trip took a total of about five hours. In addition, the cart did no onboard processing of the video images. They were transmitted by radio link to a VAX 780 computer which interpreted the data and issued the resulting steering commands. This reduces the robot to the mobile category as opposed to the mobile autonomous classification.

Dr. Moravec, now a professor at Carnegie Mellon University, is still diligently pursuing his research into creating human like vision for a mobile robot. However, he admits that to reach his goal he needs a computer which is about 1000 times faster than his current computer [9:30]. Present vision systems are not fast enough to allow real time navigation and the computers are not small enough to be built on board the robot for a mobile autonomous system.

Embedded wire systems find the desired path by following wires on or beneath the travel surface like a train on railroad tracks. This system is very simple, requires little computational ability, and navigation can be done in real time. However, it is not very flexible. Places of travel are limited to where the wire tracks are located and continuity of the wire from start to destination. Obstacle avoidance is impossible without additional sensors and even then would be limited to stopping and waiting for the obstacle to move.

Beacon systems determine the desired path from position reference information broadcast by beacons found near the path of travel. This system is also simple, easy to implement, and provides a real time navigation capability. It suffers from the need to provide beacons in the robots working environment for navigation. Additional sensors are required to provide an obstacle avoidance capability.

Optical shaft encoder systems rely on "dead reckoning" navigation by starting at a known position and moving precise distances as measured by the optical shaft encoder. This is also a relatively simple system, easy to implement, and can be done in real time. It is more flexible than the embedded wire system because it is not limited to following wire tracks, but it suffers from errors in sensor data. The distance moved data must be very accurate or error will be introduced that is cumulative and will eventually result in the robot becoming lost.

Optical shaft encoders have proven to be fairly accurate under ideal conditions. However, several sources of position error accompany their use. These position errors are unbounded and cumulative. One error source is wheel slippage, resulting in an inaccurate measurement of actual distance traveled.

A second problem lies in knowing the exact circumference of the wheel. The distance traveled by the robot is determined from the number of revolutions made by the wheel; one revolution of wheel travel equals the distance of one circumference in linear travel. As the tire wears out, the error in the measured distance grows larger.

A third problem results from computational errors. This error is most prominent when the robot has negotiated many turns, requiring numerous calculations based on geometric approximations to determine the current location of the robot. After a short period of travel, the robot will begin to stray off course due to an accumulation of error. The performance of optical shaft encoders degrades considerably as the number of maneuvers performed by the robot increases.

A fourth problem with optical shaft encoders results from not having an absolute heading reference. All heading computations from encoder data produce a relative heading to some initial heading. Initial heading of the robot must be externally provided and can itself be a very large error source. In Figure 1.1, the effect of initial heading error is illustrated. Without an accurate initial heading, the robot will eventually become lost. A robot system using

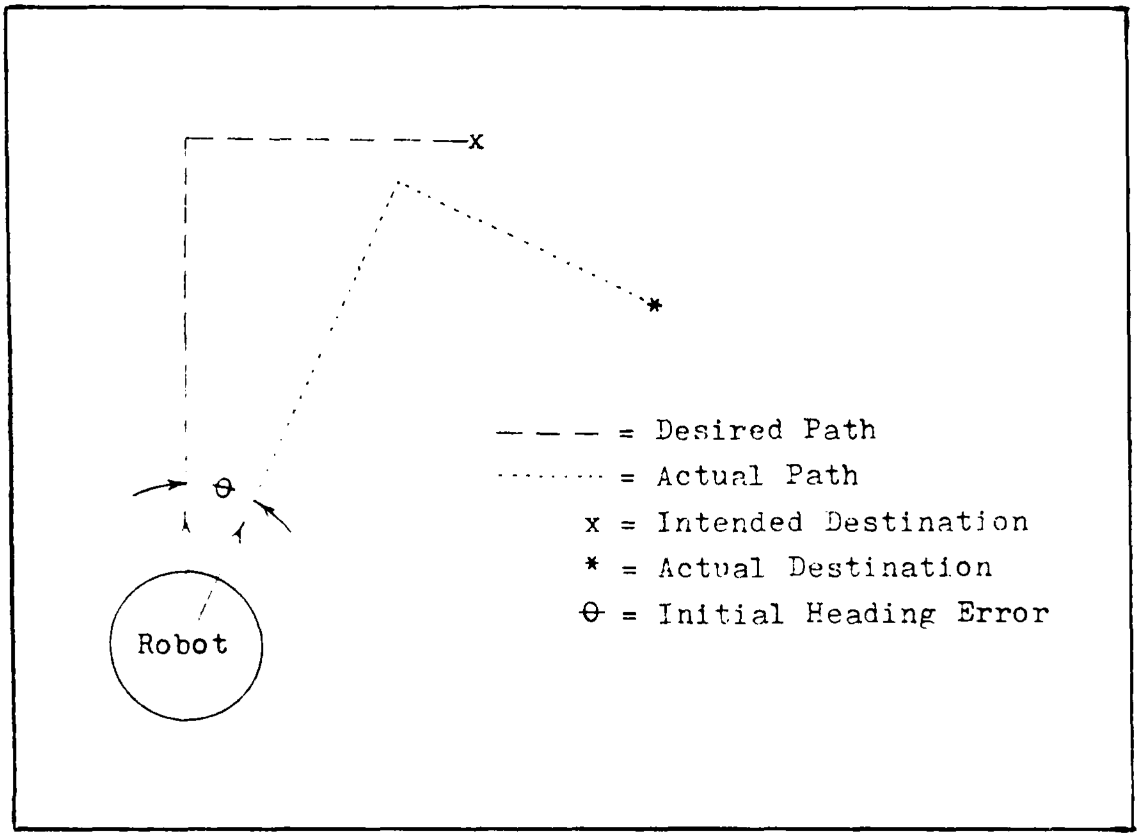


Figure 1.1. Effect of initial heading error.

optical shaft encoders for navigation is highly dependent on this initial heading reference.

Finally, optical shaft encoders offer no means for detecting obstacles. Therefore, additional sensors are required to provide an obstacle avoidance capability. Sonars (ultrasonic ranging units) can be used to complement the optical shaft encoders and together provide position and obstacle detection information. In addition, the sonars can be used to compensate for errors in the encoders by providing range information to known obstructions. In other words, the sonars can provide a position "fix" so the robot knows its location with respect to its known environment.

Dr. James Crowley [12] is experimenting with just such a system. He uses a focused rotating ultrasonic ranging device to maintain a description of the robots external environment. This description is called a sensor model. The estimated position of the robot, obtained from optical shaft encoders, is compared to the position of the robot determined from the sensor model to create a composite model. This composite model represents an average position of the robot. In this way, Crowley seeks to maintain an accurate estimate of the robots position.

It is not very feasible to use sonars as the only source of navigation data since their range is limited (making navigation through a large open area impossible) and unknown obstacles could be misinterpreted as known obstacles causing the robot to become lost. It is also important to note that sonar data is useless for navigation without an accurate heading reference. A detected object cannot be used as a reference without knowing the direction of the object in the navigation reference frame.

Previous thesis work at the Air Force Institute of Technology (AFIT) has produced a mobile robot, the Mobile Autonomous Robot Research System-1 (MARRS-1), that can map its environment and provide obstacle detection with sonar sensors. It can also measure distance travelled with optical shaft encoders. In addition, the robot has an onboard computer for dedicated navigation calculations. However, MARRS-1 is not capable of autonomous navigation. The current navigation system can only traverse a course preprogrammed into the drive computer through the robot's keyboard or by use of the teaching pendant. It can only collect, not use for navigation, sonar obstacle information and shaft encoder distance data (as presently programmed). In addition, it is not able to compensate for errors in wheel direction and distance measurements, which may cause MARRS-1 to wander from the programmed course (open loop steering control).

It is obvious that past robot navigation research, both at AFIT and abroad, has not produced an accurate, real time, autonomous, mobile navigation system. A new approach is indicated by the weaknesses and liabilities of past systems. First, an absolute and directly measurable robot heading reference is required. Second, an accurate and preferably non-mechanical method of distance measurement is necessary. Third, an accurate, long distance (5 to 50 feet or more) method of obstacle detection is required. Finally, a navigation algorithm is required that models the sensor data into a real world map and issues robot steering commands based on this mapping.

This thesis attempts to solve these problems by adapting an aircraft directional gyroscope system for the heading reference, an accelerometer for distance measurement, and sonars for obstacle detection. Robot "world modeling" and path planning is discussed, but not implemented.

PROBLEM AND SCOPE

The problem is to design and fabricate a real time, point to point, closed loop, mobile autonomous robot navigation system for the MARRS-1 robot. In addition, it must be capable of detecting and avoiding both previously known and unknown obstacles.

Real time navigation is defined to be navigation at the maximum constant motion travel speed of MARRS-1. Point to point refers to navigation from a given starting point to a given destination and includes in-course obstacle detection and avoidance. Closed loop refers to the ability to detect and correct course errors. Autonomous, in this case, will be broadened to mean needing no external support except power.

Design and fabrication will include: construction of a new third body tier to house the gyro and accelerometer based navigation system (GYRAC); physical and electrical modification of MARRS-1 to allow GYRAC integration; fabrication of a GYRAC control computer; implementation of GYRAC computer software; fabrication of a digital electronic interface between the gyro/accelerometer and GYRAC computer;

integration of the MARRS-1 drive and navigation computers with the GYRAC computer; a simple point to point navigation control program (no obstacle avoidance) to demonstrate course tracking; full testing of all new hardware and compilation of test results; complete schematic diagrams and operating manuals for all new hardware; and fully documented software listings for all operational and test programs.

Obstacle detection and avoidance were not implemented, but are available since this has been demonstrated on a previous thesis [19]. Finally, the design options available were limited by hardware and software restrictions imposed by systems previously added to MARRS-1, limited space internal to the robot, limited time, and insufficient funds.

Assumptions

Several basic assumptions are necessary for a navigation system based on a directional gyroscope and an accelerometer. This thesis is predicated on the following assumptions:

1. Assume no local disturbances will be present in the earths magnetic field. This assumption is necessary since the directional gyro output is slaved to a magnetic flux detector for absolute reference to magnetic north.

2. Assume the operational environment of the robot is a perfectly smooth and level surface. This assumption is necessary since a single accelerometer cannot distinguish true acceleration from local gravity. Therefore, any tilt of the accelerometer input axis into the vertical plane will induce an error in measured acceleration.

3. Assume a perfect integrator. An operational amplifier integrator circuit is used to integrate accelerometer output to obtain velocity.

4. Assume velocity of the robot is constant over sample period. This is necessary for simplicity and ease of calculation.

5. Assume sample time is known precisely. Using accelerometer output to ultimately obtain distance traveled by the robot is a time dependent problem.

All the above assumptions, excluding the first, are tied directly to the use of an accelerometer. Each of these assumptions will be addressed again in Chapter V.

Evolution and Capability of MARRS-1

The MARRS-1 began as a Heathkit HERO-1 robot. This original HERO-1 has since undergone substantial modification and today bears little resemblance to its former self. Lieutenant Owen [19] was the first to modify the original HERO-1. He added a laser barcode scanner and several Polaroid ultrasonic ranging units to the robot. The scanner was used to determine the location of the robot by reading barcodes taped to the floor. The first step had been taken toward creating an autonomous mobile robot. The HERO-1 could collect useful but limited navigation data on its own.

Follow-on work by Clifford and Schneider [10] was a major leap toward the goal of creating a mobile autonomous robot. Under their thesis effort, the HERO-1 was completely rebuilt. The following is a list of the major modifications performed by Clifford and Schneider:

1. A new main body was fabricated. This new body is 12

sided and consists of two separate levels. The top level can rotate with respect to the lower level.

2. The HERO-1 CPU (MC6808) was upgraded with the addition of a Virtual Devices Inc. MENOS-1 MC6801 CPU board which added RS-232 serial communication capability to the HERO-1.

3. Two dozen Polaroid sonar transducers were attached to the new robot body (one on each segment of the upper and lower body decks).

4. Optical shaft encoders were placed on the two rear wheel shafts and on the front (drive) wheel steering shaft to provide distance moved data.

5. An MC6802 based computer was added to control the optical shaft encoder subsystem and the sonar subsystem. This computer is called the Navigation computer.

The resulting heavily modified HERO-1 was renamed MARRS-1. MARRS-1 was then used to generate sonar range data and wheel distance data. This data was post processed on an external computer to create a composite map of the robot's local environment relative to the robot (whose position was determined from the shaft encoder data). This represents a significant improvement over the capability of the Owen modified HERO-1. However, like Owen's modified HERO-1, the MARRS-1 served only as a collector of navigation data. The actual movement of the robot was controlled by a human operator. MARRS-1 was not yet an autonomous robot, but it possessed the capability to become one.

The capability of the MARRS-1 at the beginning of this thesis can be summarized as follows:

1. All functions of the original HERO-1 still existed, except for the arm [10:III-3]. This capability included programable movement of the robot.

Unfortunately, this programed motion was open loop, resulting in unreliable and nonrepeatable movement. The robot could not follow a straight line path.

2. The laser barcode scanner was removed but all provisions for attaching it (both mechanically and electrically) to the robot still existed.

3. Sonar and optical shaft encoder data could be gathered by the navigation computer and relayed via an RS-232 computer interface to an external computer for storage on a floppy disk. Firmware was resident onboard the Navigation computer to obtain and transmit the data.

4. Four RS-232 computer interface ports were available for use. One port allowed communication with the main CPU (MENOS-1 upgrade board). The other three ports allowed access to the navigation computer.

General Approach

Development of a mobile autonomous robot navigation system for the MARRS-1 required a rigorous analysis of past and present robot navigation research literature. Based on this literature search, a new approach to world modeling for mobile autonomous robots was developed.

This new navigation approach required collection of specific types of data. The current systems and sensors on the MARRS-1 were compared against these new requirements and a list of sensors and systems to be constructed were compiled. A prime requisite during this analysis was to limit the data collected so that onboard computers could process the data fast enough to allow real time navigation.

The required sensors and subsystems were built and tested as was the necessary software to drive each subsystem. Modifications were then made to the MARRS-1 system to allow the GYRAC subsystem to be physically, electrically, and functionally (through software) integrated onto the MARRS-1 platform. Navigation system testing was performed and supporting data collected to validate the new navigation system.

Finally, MARRS-1 system software was developed, tested, and support data collected to demonstrate the feasibility of gyro based steering control. Conclusions and recommendations are provided based on testing at each of the various stages.

Sequence of Presentation

Chapter One provides a detailed problem background, problem statement and scope, assumptions, evolution of MARRS-1, general approach, and sequence of presentation. Chapter Two covers the GYRAC system design and theory of operation. Chapter Three discusses GYRAC system integration onto the MARRS-1 platform. Chapter Four presents a new mobile autonomous robot navigation theory for world modelling and path planning. Chapter Five discusses testing results and analysis of the various stages of the GYRAC development. Chapter Six gives a system summary and presents conclusions and recommendation. The appendices contain hardware data sheets, schematic diagrams, device layouts, wiring diagrams, connector diagrams, an equipment list, and a parts list. Included are software structure charts, program listings, collected data tables, and software operating instructions.

II. GYRAC Design and Theory of Operation

The hardware design goal of this thesis is to create a navigation data system capable of providing absolute heading and velocity data, in binary digital form, to an external computer. Three primary requirements are the basis for this design:

1. A KCS-55A Pictorial Navigation System (produced by King Radio Corp.) was ordered after selection as the best directional gyro system for this project and a QA-1100 accelerometer (produced by Sundstrand Data Control Inc.) was already on hand at AFIT. Therefore, the system must be centered around the KCS-55A Pictorial Navigation System and the QA-1100 accelerometer.

2. The navigation data system must be compatible with the MARRS-1 but also easily transportable to another robot system.

3. The system must be attainable with resources readily available to AFIT.

Since the foundation of the system is a gyro and an accelerometer, the navigation data system will hereafter be refered to as the "GYRAC" system (for gyro and accelerometer system).

Summary of GYRAC System

A general layout of the GYRAC system can be seen in the block diagram in Figure 2.1. The gyro subsystem provides analog heading information which is then converted to a 12 bit TTL (Transistor Transistor Logic) digital data signal. Angular velocity is a byproduct of this conversion in analog signal form. Thus, the analog angular velocity is digitized

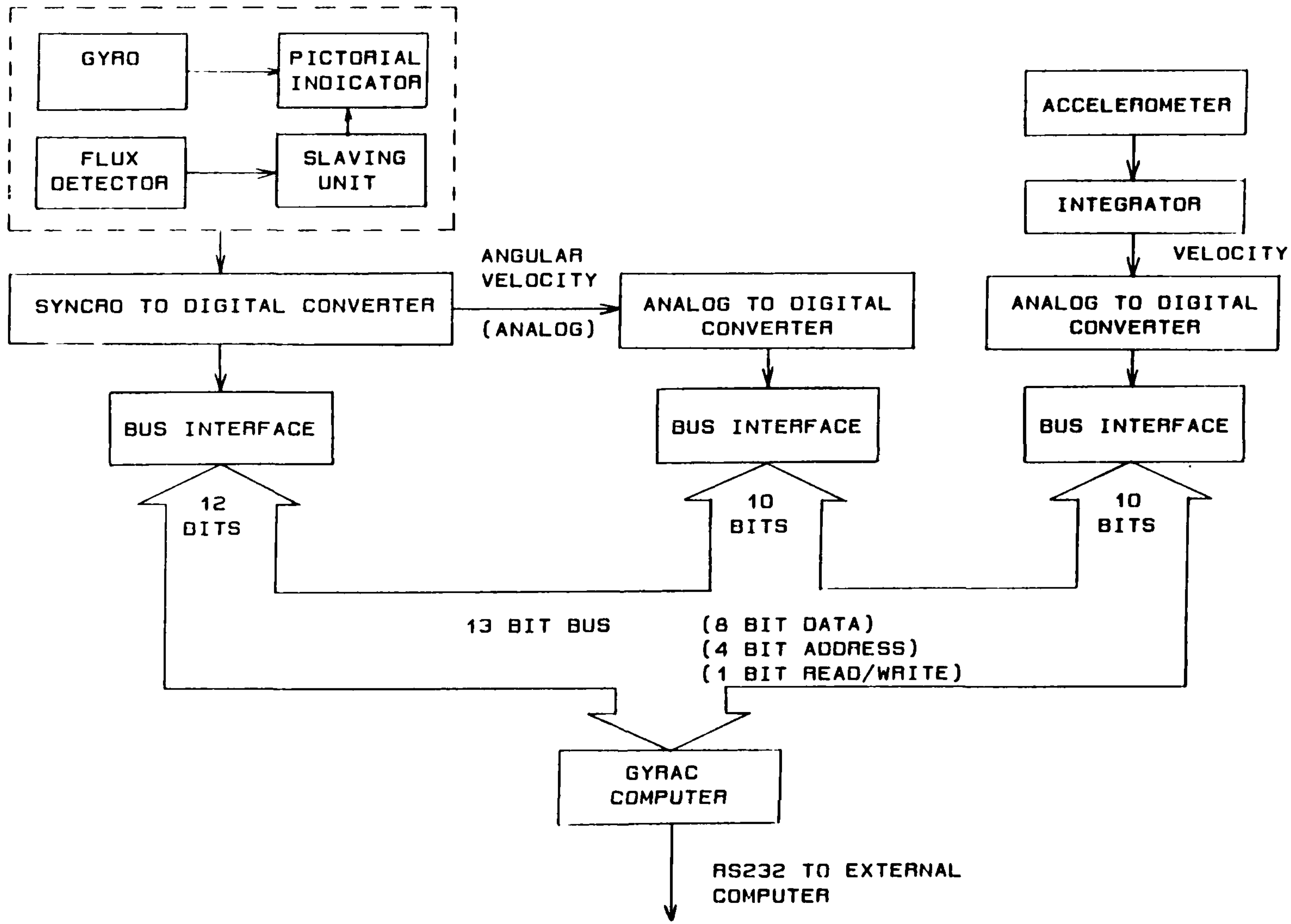


Figure 2.1. GYRAC System Block Diagram

through an analog-to-digital converter resulting in a 10 bit digital data signal. The accelerometer produces an analog signal which is integrated via an operational amplifier circuit and converted to digital TTL data by an analog-to-digital converter. The three digital data signals are connected to a common thirteen wire bus containing eight data lines, four address lines, and a read/write line. This bus serves as a standard sensor interface to the GYRAC computer. The GYRAC computer interprets commands received from an external computer via an RS-232 serial data link and acquires the appropriate sensor data as directed by the external command. The GYRAC computer then performs any necessary preprocessing of the data, converts it to serial format, and transmits it to the external computer via the RS232 link.

The Gyro Subsystem.

The gyro subsystem is composed of four major elements: a directional gyro; an indicator unit; a magnetic flux detector; and a slaving unit. The directional gyro provides a gyro stabilized magnetic heading to the indicator. The directional gyro consists of two primary components: the gyro itself and a base assembly, see Figure 2.2.

The gyro is a spinning mass precision gyro with two degrees of freedom, see Figure 2.3. Relative angular displacement is sensed by an optical encoder assembly mounted on the gyro's outer gimbal. Electrical outputs from

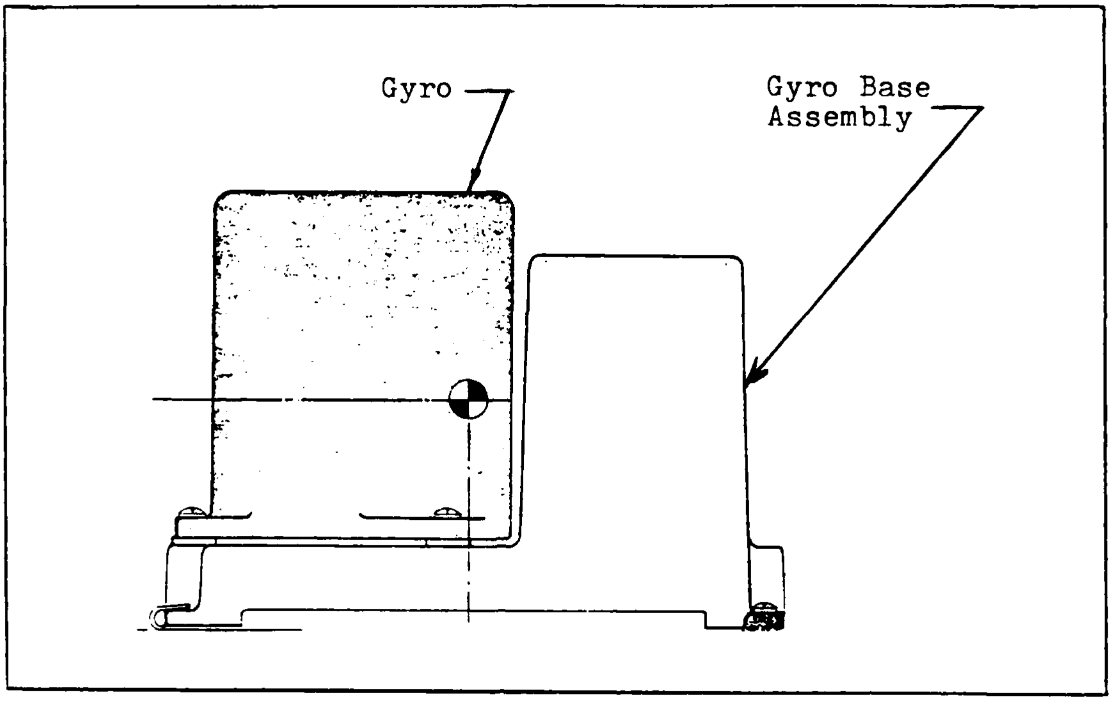


Figure 2.2. Directional Gyro [1:2-11]

the optical encoder are two square waves which are used to drive a stepper motor in the indicator. The gyro base assembly contains the control logic for the gyro and the slaving logic for the indicator. The gyro base also serves as the power supply for the entire gyro subsystem. From the single 28 volts DC input into the gyro base, the following internal voltage supplies are generated: 26 VAC 400 Hz for the gyro spin motor, flux detector excitation, and heading syncro excitation; + and - 15 VDC regulated supply for the analog circuitry in the system; +15 VDC unregulated voltage fo~ the stepper motor in the indicator; and +5 VDC regulated supply for the system digital logic circuitry [3:5-33].

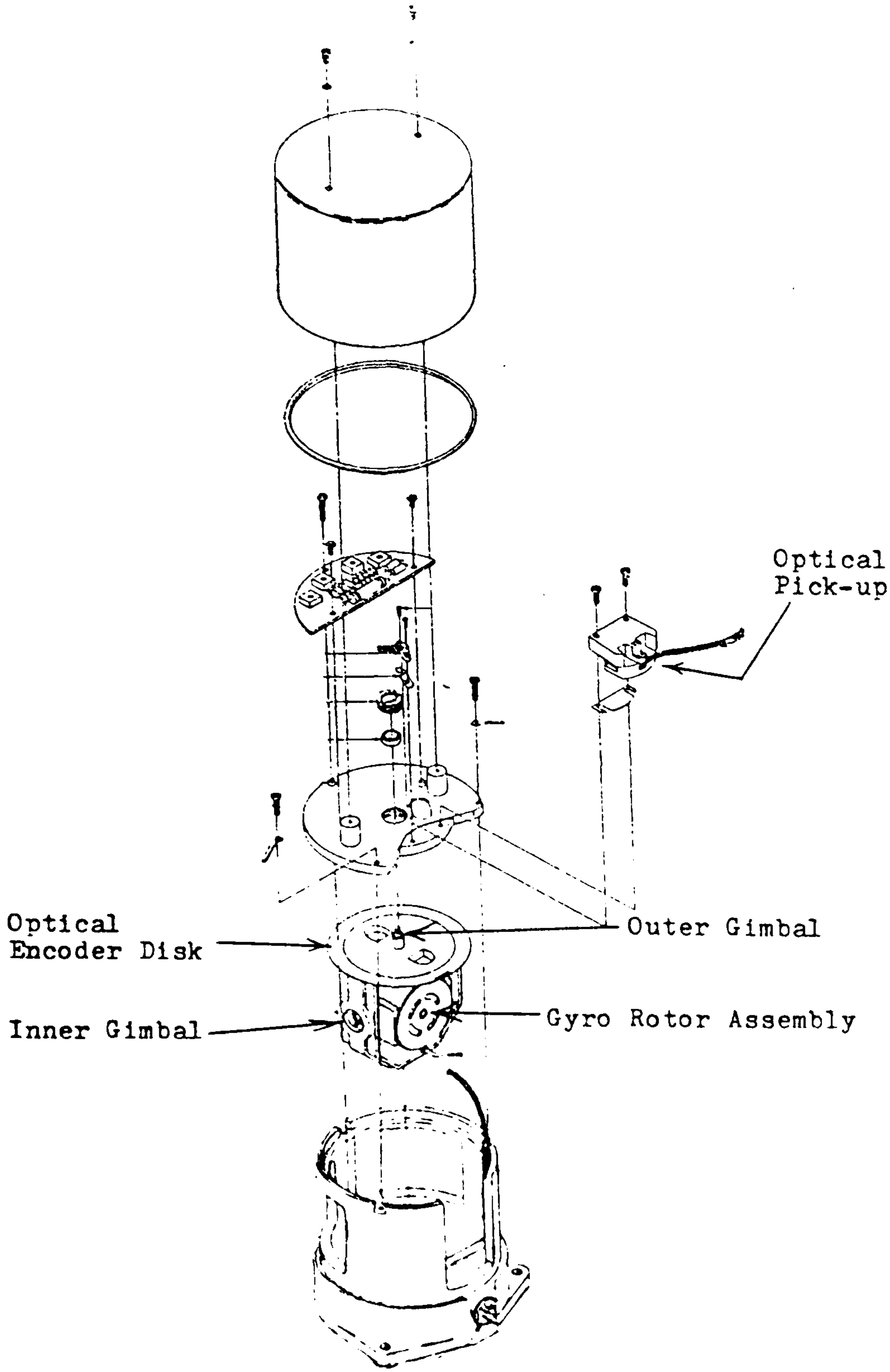


Figure 2.3. Gyro, Exploded View [2:5-33].

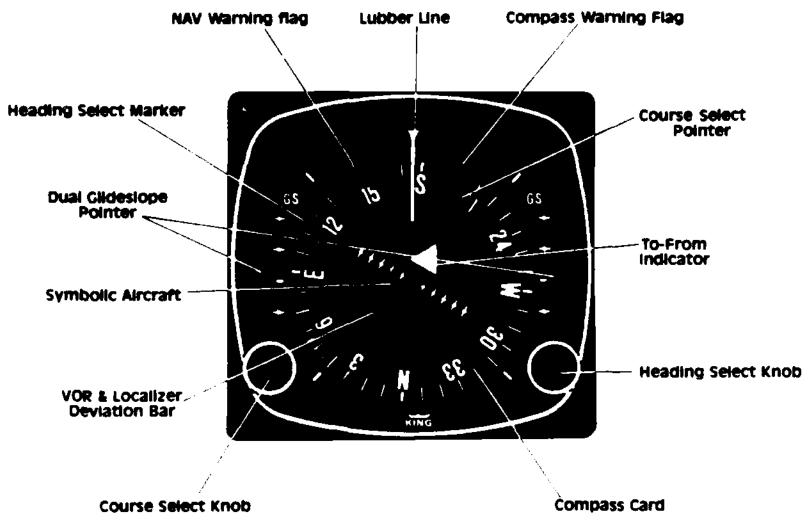


Figure 2.4. The Heading Indicator [1:2-9]

Also, separate regulated grounds are maintained for the analog and digital circuitry. For the GYRAC system, the above mentioned supplies and grounds are routed to a central power distribution panel to provide the necessary power for other GYRAC hardware (see Appendix E).

The indicator is typical of the type seen in the cockpit of small aircraft, as shown in Figure 2.4. A digital stepper motor is used to drive the heading display in response to the signals generated in the directional gyro. The signals from the gyro consist of a two phase excitation drive that is connected to the four stepper motor leads as shown in Figure 2.5. Each time the A or B waveforms (see

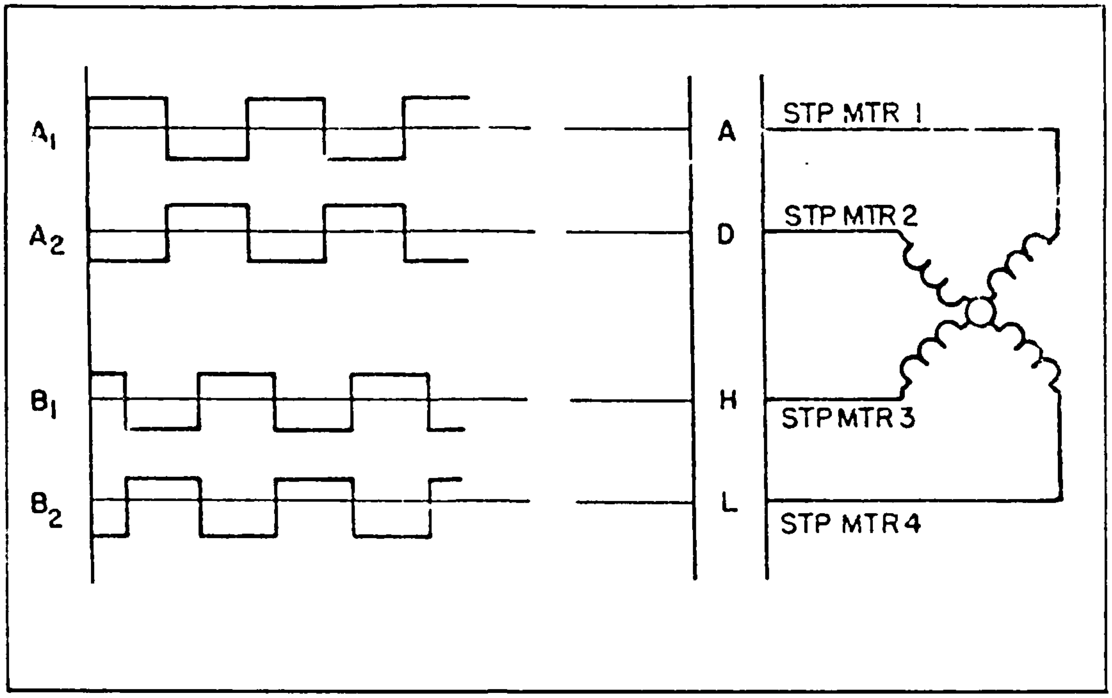


Figure 2.5. Stepper Motor Drive Circuit [2:4-1]

Figure 2.5) change state, the motor shaft moves nine degrees in a direction determined by the previous state of the waveforms. This motion is reduced to a 0.25 degree card rotation by a 36:1 gear train assembly [2:4-1]. Thus, the display card moves in increments of 0.25 degree thereby limiting the resolution of the heading angle to + or - 0.25 degree movement of the indicator display can be tracked by a syncro control transmitter (CX) which is mounted internally on the rear of the compass card shaft (see Figure 2.6). This CX is intended to provide a slaving signal to another display, but can be used to get an absolute electrical representation of heading. This fact is crucial

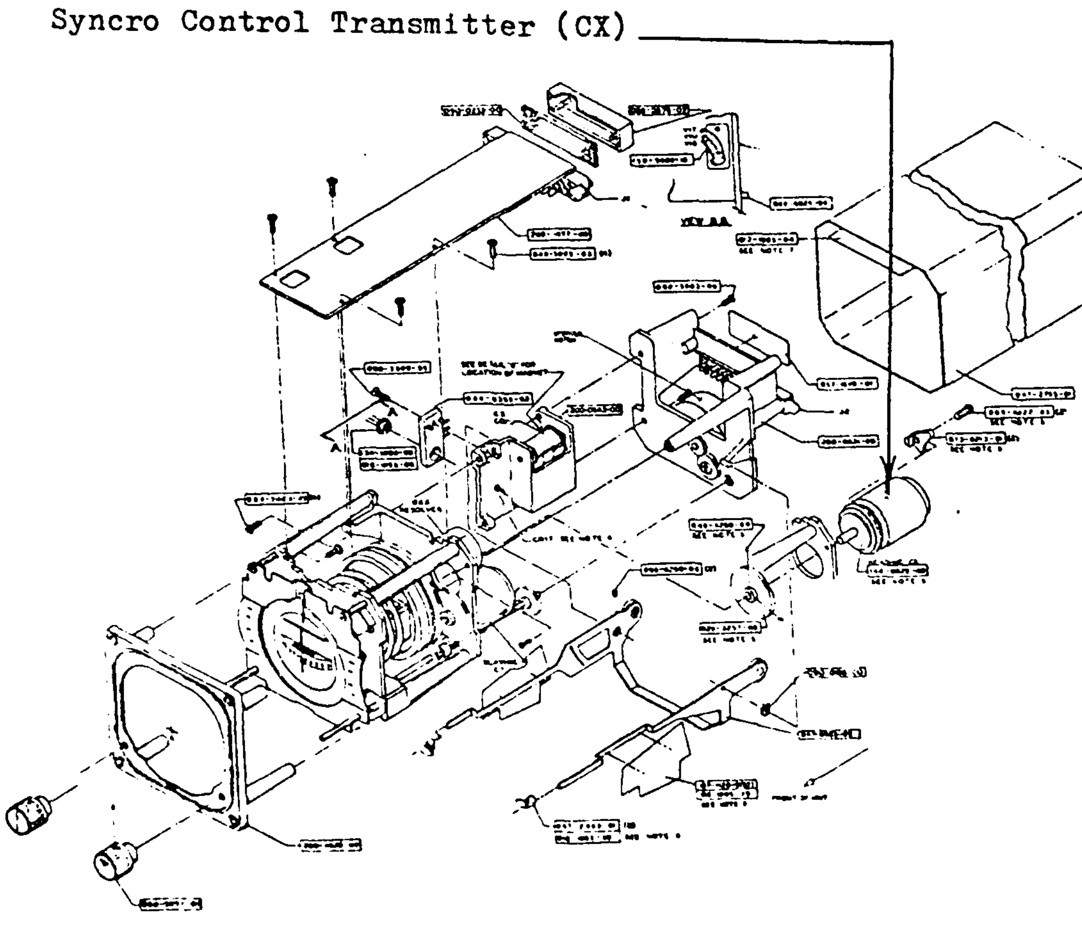


Figure 2.6. Indicator, Exploded View [2:5-5] .

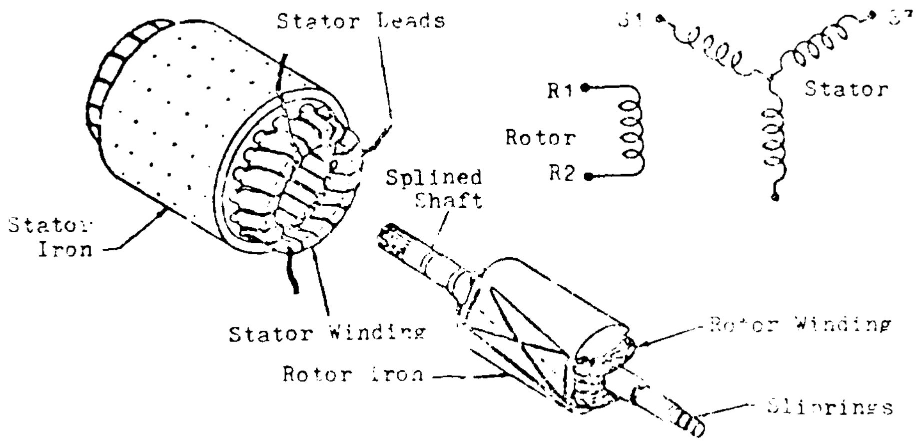


Figure 2.7. Internal Structure of a Syncro Control Transmitter and its electrical representation. [18:2]

to the realization of the GYRAC system. If the rotor of the CX is excited with a reference voltage (AC), then syncro format voltages will appear as output across the S1, S2, and S3 terminals (see Figure 2.7) [18:2]. These voltages are a function of the shaft angle O. For example, if the rotor (which has a single winding) is excited by a reference voltage across R1 and R2 (see Figure 2.7) of the form:

$$A \sin(wt)$$

Then the voltages which will appear across the stator terminals will be:

$$S1 \text{ to } S3 = A \sin(wt) \sin \theta$$

$$S3 \text{ to } S2 = A \sin(wt) \sin(\theta + 120)$$

$$S2 \text{ to } S1 = A \sin(wt) \sin(\theta + 240)$$

where θ is the shaft angle.

These voltages are known as syncro format voltages [18:2]. A desirable result of the syncro output is that it can be easily converted to a digital signal with a standard syncro-to-digital (S/D) converter. An SDC1700 12 bit S/D converter made by Analog Devices is used (see Appendix A and C) to provide a TTL digital binary representation of the heading angle (shaft angle). The SDC1700 also provides an angular velocity output in analog form which will be converted to digital by an analog-to-digital (A/D) converter. The A/D converter to be used is also produced by Analog Devices and is a 10 bit converter (Part # AD573, see Appendix A and C for detail).

The magnetic flux detector senses the direction of the earth's magnetic field and converts this information to a three-wire syncro format, much like the CX in the indicator. This information is transmitted to the indicator for slaving purposes, see Figure 2.8 for an exploded view of the detector. The flux detector can be oriented so the gyro system displays a heading relative to some artificial North direction. This feature is used to align the GYRAC system

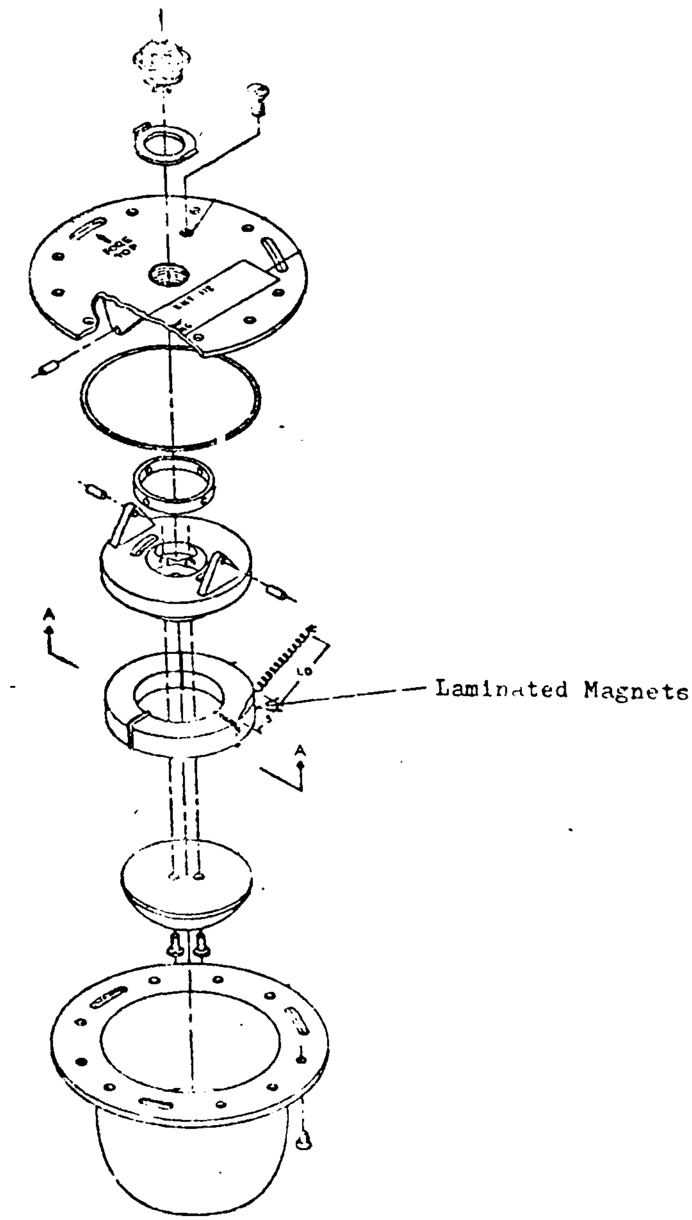


Figure 2.8. Flux Detector, Exploded View [4:5-3].

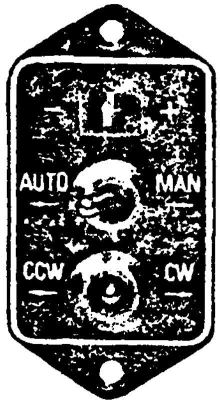


Figure 2.9. Slaving Unit [1:2-19]

with some convenient navigation coordinate system in the test area.

The slaving unit, shown in Figure 2.9, contains a slaving meter, slaving switches, and corrector circuitry which can compensate for the effect of local magnetic disturbances on the flux detector. The meter current is generated in the directional gyro base assembly (slaving logic) and represents the difference between the flux detector sensed heading and the heading displayed on the indicator. The slaving switches allow the gyro system to be operated in either a free-gyro mode (no slaving with the flux detector) or in the slaved mode (automatic slaving with flux detector). There is also a manual slaving switch which can be used to rotate the display card in the indicator either clockwise or counter-clockwise. In addition to the slaving meter and slave switching functions, the slaving unit also includes a compensation circuit. This circuit causes a shift in the magnetic direction vector and thus can compensate for "hard iron" effects caused by nearbye ferrous materials.

The Accelerometer Subsystem.

The accelerometer used in this thesis is a QA-1100 servo-type single-axis accelerometer produced by Sundstrand Data Control Incorporated (see Figure 2.10). The sensor, as

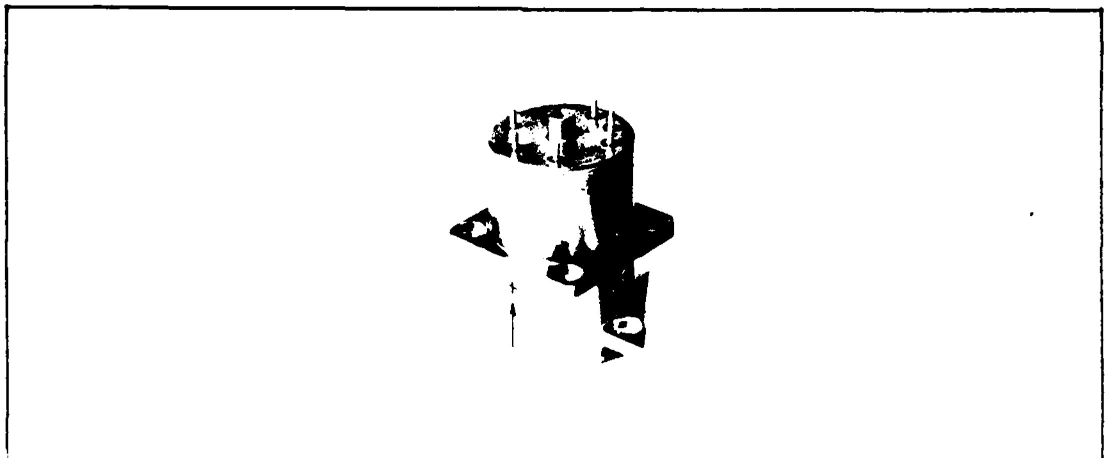


Figure 2.10. The QA-1100 Accelerometer [20]

shown in Figure 2.11, consists of the following key elements [7:1-3]:

1. A proof mass, pendulously supported and ideally constrained so as to allow only one degree of freedom about a well defined axis fixed within the

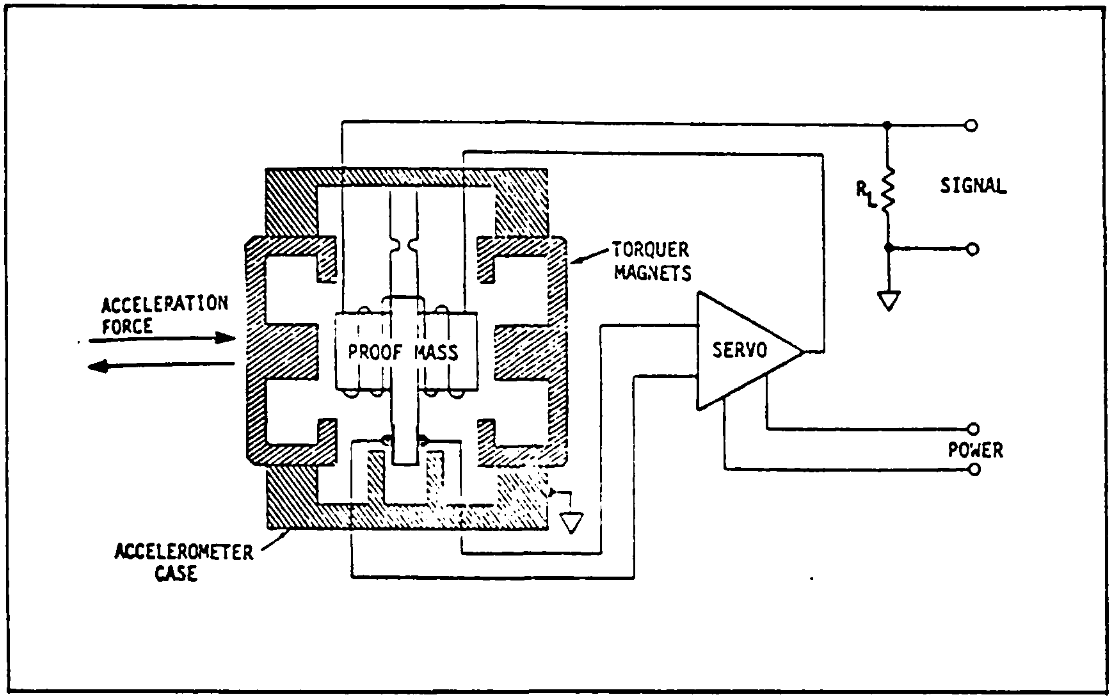


Figure 2.11. Basic Structure of a Servo-Type Accelerometer [7:1-3]

sensor.

2. A pick-off that can sense extremely small displacements of the proof mass.

3. A torquer, which is a coil positioned within a permanent magnetic field and attached to the proof mass, allowing force to be applied to the proof mass in response to a current passed through the coil.

4. A restorer circuit, or servo, that causes an electrical current to flow through the torquer coil in response to a pick-off signal. The resulting electromagnetic force balances the inertial reactive forces.

The basic operation of the accelerometer is that of a linear single axis electro-mechanical device for measuring acceleration. The operation is based on movement of the proof mass during acceleration. A pickoff senses the displacement of the mass and the servo amplifier develops a current which is supplied through the torque coil to rebalance the proof mass. Thus, the rebalance current is proportional to the sensed acceleration and is a very accurate measure of acceleration. As more acceleration is applied to the accelerometer, the assembly will maintain the proof mass position and rebalance current will increase with increased acceleration until the sensor saturation limit is reached. An exploded view of the actual sensor assembly can be seen in Figure 2.12.

This type of accelerometer does not come with an internal (factory set) load resistor (R_L in Figure 2.11). Thus, an external load resistor must be provided. This is desirable since the ranging, or sensitivity, of the accelerometer output can be chosen to suit specific applications. The value of the external load resistor is determined by the following formula [7:3-6]:

$$R_L = \frac{\text{voltage sensitivity desired}}{\text{current sensitivity of the accelerometer}}$$

The current sensitivity of the QA-1100 is approximately 1.3mA/g (where g is the acceleration due to gravity). For this thesis, the accelerometer is configured to have a

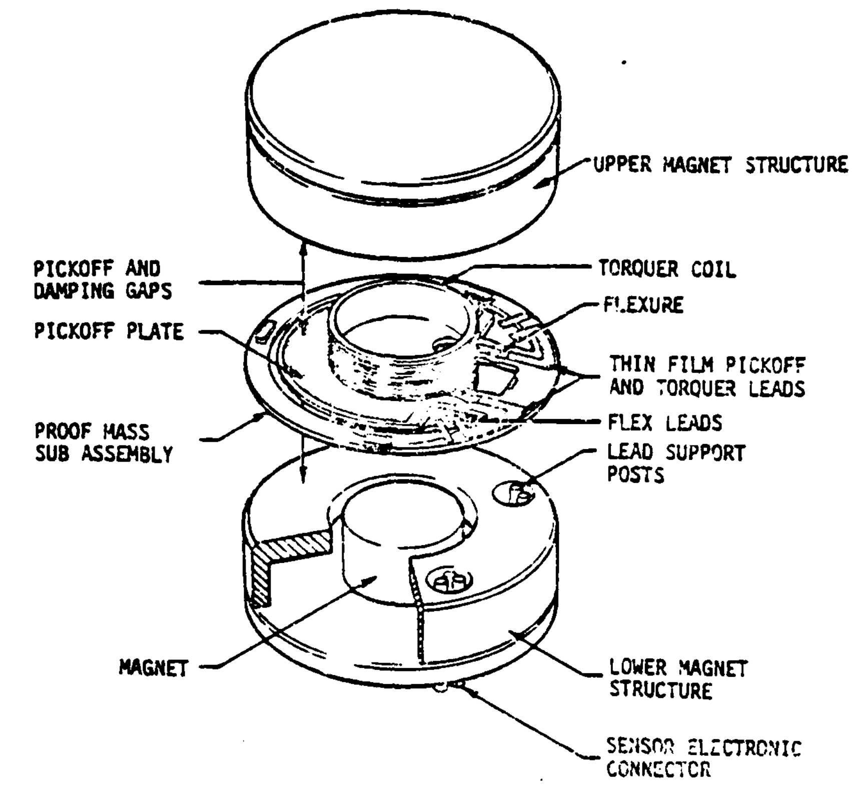


Figure 2.12. QA-1100 Sensor Assembly, Exploded View [7:1-4]

sensitivity of 2 volts/g. R_L consists if a single precision resistor in series with a 10 turn trim-pot for fine tuning of the sensitivity (see Appendix B for schematic detail).

As with any measurement device, the accelerometer has a scale factor and bias error. However, without the use of a centrifuge, these values are very hard to determine to a substantial degree of accuracy. Nonetheless, a tumble test can be performed and has been performed. A tumble test consists of positioning the accelerometer with the input axis exactly vertical, pointing downward and then upward. The sensor will detect the earth's gravity vector. The two measurements (input axis up and input axis down, refered to as V_{OUT}) are then used in the following equation:

$$V_{OUT} = V_{ACT} \times \text{Scale factor} + \text{Bias}$$

Here V_{ACT} is 2 volts (since 2 volts/g is the sensitivity of the accelerometer). Use of this equation results in two equations and two unknowns (scale factor and bias). Preliminary testing of the accelerometer has resulted in a very small value for bias (about 2 milivolts) and a scale factor of very near unity. Thus, for this thesis, the scale factor is assumed to be equal to one and the bias is assumed to be zero. This assumption will be discussed further in Chapter V under Review of Assunptions.

Accelerometers cannot distinguish between gravity and true acceleration. This fact is a major concern in the GYRAC system and must be accounted for. A special mounting platform has been built for the accelerometer allowing for complete leveling. The accelerometer platform will be initially adjusted until a very near zero reading is established from the accelerometer. However, during movement of the GYRAC system it is highly likely that errors will occur in the accelerometer output due to travel over a non-level surface. This problem is discussed further in Chapter

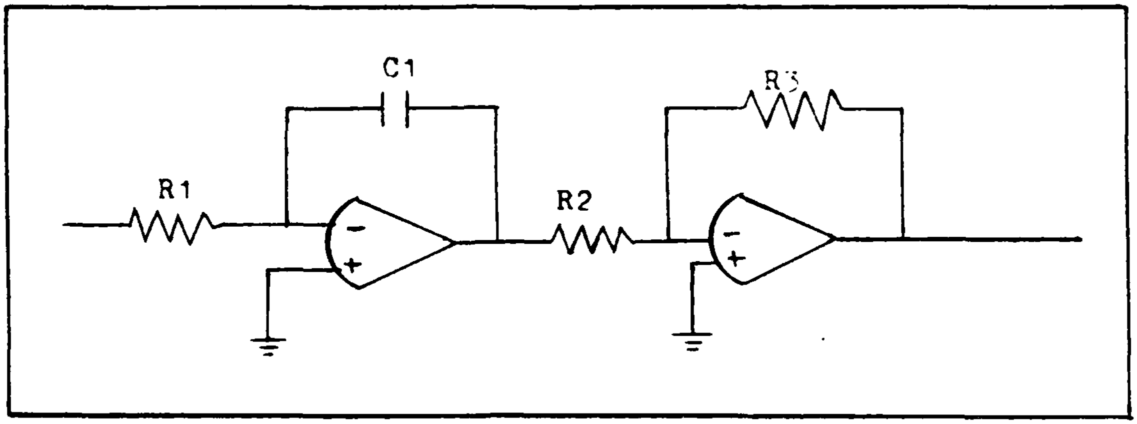


Figure 2.13. Accelerometer Integrator and Scaling Circuit

V under Review of Assumptions.

The output from the accelerometer is connected to an integrator circuit, shown in Figure 2.13. The output, which is velocity, is scaled such that one volt is equal to one foot per second of velocity. This analog voltage is then fed into an A/D converter (another AD573) resulting in a 10 bit binary representation of velocity.

GYRAC Computer and Interfacing Subsystem.

Up to this point, the gyro subsystem and the accelerometer subsystem have been discussed. The resulting output of these systems will be in digital form as mentioned earlier. The remaining task is to transmit these signals out to an external computer where they can be used for navigation purposes. This is accomplished through a bi-directional GYRAC sensor bus, the GYRAC computer, and an RS-232 interface. The sensor bus contains eight data lines, four address lines, and one read/write line. Since the gyro and accelerometer data is all larger than 8 bits, the data from each of these devices must be gathered in two separate parts. This causes timing problems since the S/D and the A/D converters constantly update themselves with the most current measurement. This means that after a data signal is obtained, the value in the converter will change before the second half of the data signal can be transmitted.

This problem was solved by latching the data into a set of tri-state latches. These latches will hold the data as long as necessary, allowing sufficient time to transmit both bytes of the data signal. See Appendix B for more design detail. Also, since each data signal from each converter is divided into two parts, a separate address is used for each part. Thus, six addresses are needed to obtain all the gyro and accelerometer data. A seventh address is used to reset the integrator constant to zero (by discharging the capacitor over the op-amp). This reset is required to insure no initial condition exists on the integrator and can be used to reset the integrator periodically when the GYRAC system is not moving. See Appendix B for detailed design layout of address decoding.

All computer interface devices, the integrator circuit, the A/D converters, the accelerometer load resistor circuit, and a set of 7-segment LED displays are located on a general purpose wire-wrap card. The LED displays are used to read the heading information coming from the S/D converter and display it in hexidecimal format. This information is used for initial alignment of the flux detector and troubleshooting. A layout of the wire-wrap card and the LED circuitry can be seen in Appendix B.

The GYRAC computer was originally a custom printer interface card built for AFIT, but was modified to its present state. The processor is a 6802 based microprocessor with 1k of ROM and 128 bytes of RAM. A modification was made to add an additional 2K bytes of static RAM to increase the memory capability. The new memory map is detailed in Appendix D. The computer also contains an asynchronous communication interface adapter (ACIA) which converts eight bit parallel data to RS-232 format serial data and handles all handshaking to an external computer. A parallel interface adapter (PIA) is also resident on the computer card which acts as the interface between the GYRAC sensor bus and the GYRAC computer bus. A power modification was also made to the computer to create its necessary + and - 12 volts and -5 volts from the + and - 15 volts available from the gyro base. The GYRAC computer software is present in the 1k EPROM (see Appendix F for program listing) allowing the computer to receive and respond to commands from an external computer via an RS232 serial link. A schematic diagram of the GYRAC computer showing the RAM and power modifications can be seen in Appendix D. Appendix E contains the edge connector wiring diagrams for all the GYRAC system circuit boards (S/D converter card, interface card, and computer card) showing all interconnecting plugs.

GYRAC System - Theory of Operation

The purpose of this section is to provide a clear picture of how the GYRAC system functions as a whole. It is intended to supplement the previous subsystem descriptions. This discussion begins by explaining what occurs on system power-up and ends with a description of how the system responds to a command input. The slaving switch is assumed to be placed into the "slaved" position (representing a full up configuration of the GYRAC). The GYRAC must be stationary upon power-up to allow for stabilization of the flux detector.

Once power has been applied to the system, the rotor (spinning mass) in the gyro begins to rotate. Output from the gyro (the two square waves) is paused while the rotor comes up to operational speed (16,000 rpm). During this same time, a red HDG flag (compass warning flag - see Figure 2.4) is displayed on the indicator face. This red flag is a visual indication that the displayed heading is not valid. While the rotor is coming up to speed, the slaving signal from the magnetic flux detector is allowed to pass to the indicator providing the reference signal for magnetic north. The compass card in the indicator is rotated at the fast slaving rate, 360 deg/min [6], until the reading on the indicator is in agreement with the magnetic flux detector slaving signal. Once the rotor has reached operational speed, the red HDG flag is removed and the indicated heading is valid. This usually occurs about one to two minutes after power-up. Any robot system using the GYRAC must account for this spin-up and alignment time (perhaps through a timed delay before requesting initial GYRAC data).

The absolute heading of the GYRAC system will be accurately shown on the indicator and on the LED display in 12 bit hexidecimal. Any rotational movement of the GYRAC will be sensed by the gyro which provides the signal to keep the indicator accurately positioned. In addition, the indicator will respond to deviations from the flux detector slaving signal at the slow slaving rate, 3 deg/min [6]. This slow rate is used to prevent the indicator from trying to follow an unstable reading from the flux detector. The flux detector is very sensitive to movement, so its output can only be trusted after it has stabilized. At this point (after the initial alignment), the flux detector signal serves primarily to compensate for gyro errors, such as drift. This particular gyro has proven to be a very accurate and stable reference. The drift rate of this gyro is less than 0.25 deg in 12 hours [8]. For this reason, the slaving signal from the flux detector could be turned off (switch to "free gyro" mode) after initial alignment is obtained.

After the heading data becomes valid, the GYRAC is ready to receive a command input. The firmware operating in the GYRAC computer is continuously checking for an input. Once an input arrives, it is compared to a list of acceptable commands. An acceptable command is a single byte of data in ASCII format representing the capitol letters A to O, see Appendix F for command definitions. If it is a valid command, the firmware program sets the appropriate address on the bus to enable the requested data (be it heading, heading rate, or velocity). The desired data is collected over the data bus (one byte at a time), converted to serial format and transmitted out via the RS-232 interface. The RS-232 interface is a simple three wire interface consisting of transmit data, receive data, and ground. See Appendix E for more detail.

It is important to note that the digital heading output is in a right-handed reference system. That is, the heading angle increases with counter-clockwise robot rotation. This is backwards from the visual indicator unit. The indicator displays increasing heading angle for clockwise rotation. Therefore, the digital output from the GYRAC and the LED displayed heading will not agree with the visual indicator except at 0 and 180 degrees. The GYRAC digital heading output was intentionally made to conform to the more conventional right-handed reference system.

III. Integration of the GYRAC System onto MARRS-1

Structural

The entire GYRAC system is contained in a new third body tier which has been added to the top of the existing MARRS-1 physical structure. It is separated and supported from the lower body tiers by eight 10.0 inch by 3/8 inch diameter threaded and tapped aluminum rods. The all aluminum third tier is 12 sided and 20.5 inches by 20.5 inches by 7.0 inches high and contains two swing down removable-pin hinged doors to allow easy access to internal components. An 18.0 inch U-shaped aluminum tower extends above the third tier to provide support and ferro-magnetic isolation for the gyro's magnetic flux detector.

In addition, four aluminum plates were constructed and attached to the first and second body tiers locking them together into a single rigid body. This was done because the original robot design allows for separate body rotation of the first and second tiers. The GYRAC requires a fixed orientation relative to the entire body and can not tolerate rotation without introducing navigation errors.

Electrical

The electrical and mechanical subsystems of the GYRAC are completely isolated and independent of the remainder of MARRS-1. Power for the GYRAC is supplied over an external cable and connects to the body tier via a four pin DIN plug (see Appendix E for detailed power distribution). All gyro commands and data are passed to and from the GYRAC via a standard three wire RS-232 serial interface. Connection is made on the GYRAC body via a standard RS-232 DB25 cable connector (see Appendix E for pin out details). These are the only two external connections required to operate the GYRAC. It is important to note that both of these connections and system operation is independent of MARRS-1. Therefore the GYRAC could easily be removed from the MARRS-1 structure and mounted on a different platform.

Utilization of the GYRAC system by MARRS-1 for navigation requires communication between three different onboard computers and a single external disk based computer for program transmission and data collection. Figure 3.1 illustrates the required interconnections.

The navigation computer, a Motorola 6802 based system resident in the first body tier, is the navigation system control computer. Its purpose is to collect sensor data from the GYRAC and drive computers, transmit collected data to the external computer, analyze this data and decide how to move, and then issue the appropriate commands to the drive computer.

The GYRAC computer, a Motorola 6802 based system contained in the third body tier, accepts requests for data, formats the data if necessary, and then transmits the requested data.

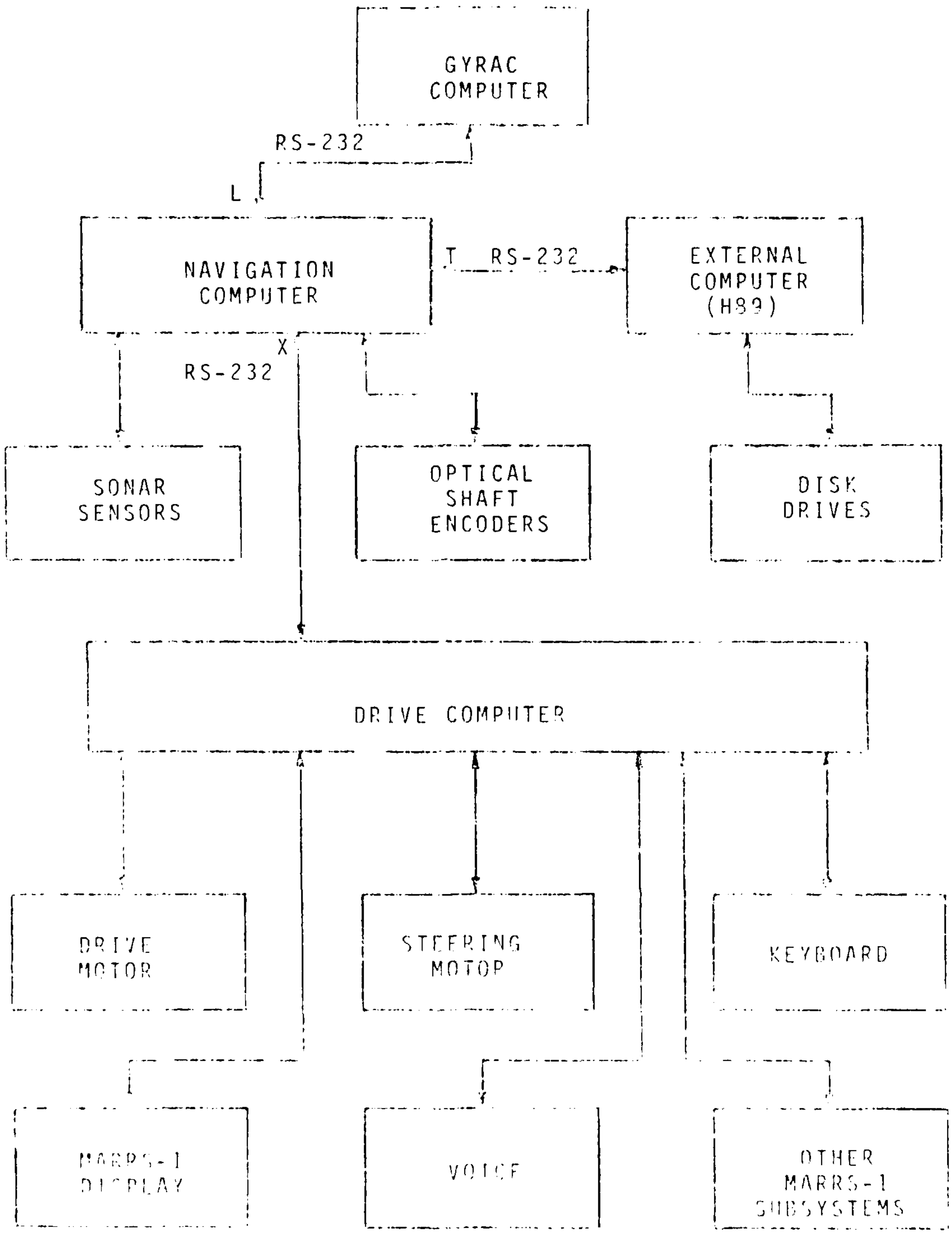


Figure 3.1 MARRS-1 Computer System

The drive computer, a Motorola 6801 based system by Virtual Devices located in the first body tier, is the main robot computer. It controls all robot sensors and devices except the sonars and optical encoders, which are controlled by the navigation computer and the gyro and accelerometer which are controlled by the GYRAC computer. This computer is able to respond to both requests for data and commands to activate a device. However, as used in this thesis, the drive computer only accepts commands to move the steering wheel and start and stop the main robot drive motor.

The external computer, a Z80 based CP/M system by Heathkit, would not be required in a field deployed operational robot. However, as used in this thesis for data collection, it must be connected in order for the navigation software to function correctly.

All communication between the four computers is done via standard three wire RS-232 serial data links at 9600 baud. A cable is connected between the navigation computer Port X and the drive computer MENOS port. A second cable is connected between the navigation computer Port L and the GYRAC computer. The last cable is connected between the navigation computer Port T and the external computer. All cable connections are made with standard RS-232 DB25 connectors. They are located on the robot's rear lower panel, except the GYRAC connector which is on the back of the third body tier. Notice that all inter-computer communication must go through the navigation computer.

Port L of the navigation computer was not originally designed to support 9600 baud. Therefore, a modification was made to the navigation computer board to allow Port L to select from one of eight switch selectable baud rates. It is now identical to the layout of Ports X and T [10]. All ports are currently set to 9600 baud.

In addition, the DB25 connector on the lower rear panel was wired in parallel to an existing internal cable to provide both laser barcode communication at 300 baud (original cable) and GYRAC communication at 9600 baud (new connector). Note that both functions can not be used simultaneously.

Software

The MARRS-1 GYRAC system consists of four different custom software programs which can be run in three different system configurations to provide both test data and MARRS-1 navigation.

The first configuration allows direct communication with the GYRAC computer to allow testing, calibration, and checkout of the GYRAC subsystem. It makes use of the GYRAC program resident in read only memory (ROM) on the GYRAC computer board. An RS-232 cable must be connected between the GYRAC and the external computer. The modem 720 program (M72) is executed on the external computer to provide outside communications capability. Commands are typed on the external computer's terminal and the corresponding data from the GYRAC is displayed. See Appendix F for complete operating instructions, structure charts, and program listings. Note that not all data is displayed since the GYRAC data is transmitted in a raw eight bit serial format which produces occasional non-printable characters.

The second configuration allows for collection and storage of heading, velocity, and angular velocity data at precise 0.1 second intervals. In addition, time mark data and distance moved from all three wheel's optical shaft encoders is provided. All data is reformatted to printable hexadecimal format which may be displayed on the external computer's terminal, saved to disk, or printed on the printer. It makes use of the GYRAC monitor program, in the GYRAC computer, and the GTEST overlay program, in the navigation computer (see Appendix F and G for GYRAC and GTEST program details). An RS-232 cable connection is required between the GYRAC computer and the navigation computer Port L and between the external computer and the navigation computer Port X. The M72 program is executed on the external computer to provide communication with MARRS-1 to send appropriate commands and receive data. See Appendix G for complete operating instructions, structure charts, and program listings. The MBASIC programs CONVERT and POSITION (see Appendix I) may be run on the saved data to produce a data plot.

The third configuration demonstrates limited mobile autonomous robot navigation (using only heading data) and collection and storage of gyro heading data. The heading data is reformatted to printable hexadecimal format which may be displayed on the external computer's terminal, saved to disk, or printed on the printer. It makes use of the GYRAC monitor program, in the GYRAC computer, the MARRS.NAV program in the drive computer, and the NAV program, in the navigation computer. An RS-232 cable connection is required between the GYRAC computer and the navigation computer Port L, between the navigation computer Port X and the drive computer (MENOS), and between the external computer and the navigation computer Port T. The M72 program is executed on the external computer to provide communication with MARRS-1 to send appropriate commands and receive data. See Appendix H for complete operating instructions, structure charts, and program listings. Note that the NAV and MARRS.NAV software demonstrates a very simple method of navigation and inter-computer communication. They are not intended to form the basis of a field application, but to illustrate gyro functionality.

IV. General Robot Navigation Theory

With the recent growth in research in the area of mobile and autonomous robotics, it is only a matter of time before a truly autonomous mobile robot becomes a reality. This robot will possess a navigation system capable of gathering and processing sensory information to accurately determine its location. In addition, the navigation system will also maintain a world model of the robots environment, perform path planning (determine travel routes around known obstructions), and provide for dynamic obstacle avoidance (method of surmounting unknown obstacles). The task of the navigation system will be very complex and its future development is crucial to the realization of a mobile autonomous system.

Two major aspects of the robot navigation problem, world modeling and path planning, will be the topic of this chapter. Dynamic obstacle avoidance is considered beyond the scope of this thesis and will not be covered. First some governing assumptions will be discussed. Second, an overview of several popular approaches to world modeling will be presented. Third, a new world modeling technique will be introduced. Finally, this chapter will conclude with a detailed presentation of path planning based on this new world model.

ASSUMPTIONS

Since the world model is intended for use by a land based robot (MARRS-1 in particular) which can only move in two dimensions, only a two dimensional "floor plan" type world model will be considered. Robots that could extend or shrink themselves vertically would constitute a special category which is beyond the scope of this paper. For more information on three dimensional modeling and path planning see [15]. This section will also be concerned only with a robot which can be modeled in two dimensions as a circle (consistent with the use of MARRS-1). Some techniques for treating robots of other geometries can be found in [15]. Finally, it is assumed that all locations on the world map can be represented directly in an absolute reference frame.

PAST APPROACHES

World modeling can be thought of as providing a description (in essence a map) of the robots known operating environment. This information must be expressed in terms that the robot can easily understand and best utilize. Virtually all models to date represent the physical world of the robot in two dimensions using an outline picture method. Two approaches have been used to describe the robots world. One approach has been to model all the obstacles in the robots world. The other approach has been to model the free space or safe areas of travel for the robot. Basically, the choices are to model where the robot can or cannot go.

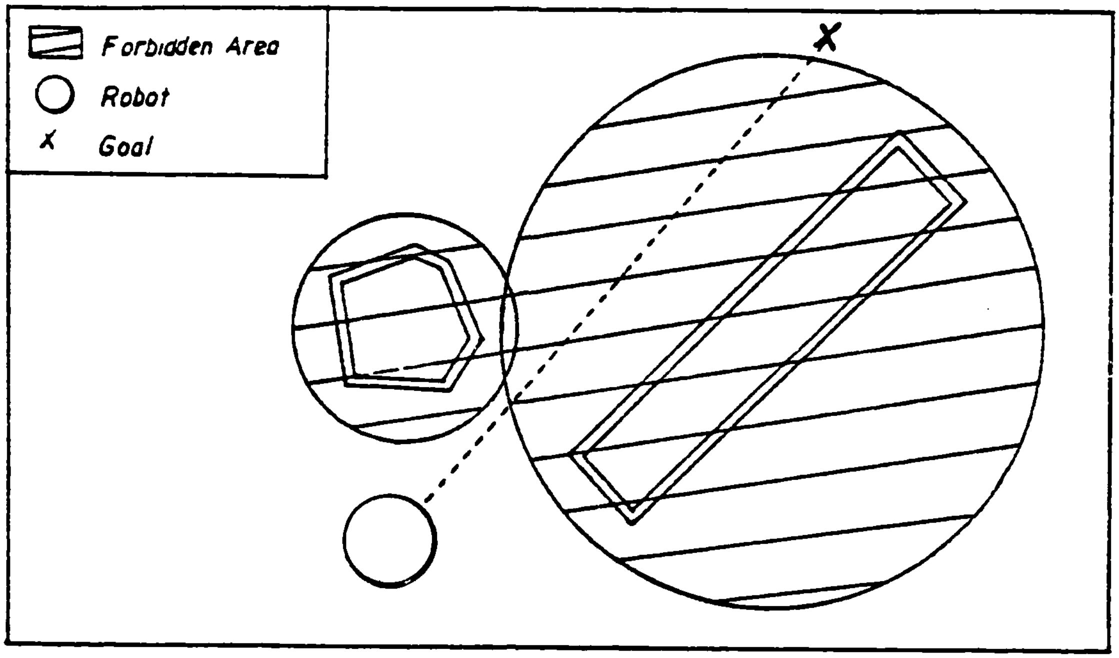


Figure 4.1. Circular approximations of physical objects [16:24]

Moravec [17] proposed modeling all physical obstacles with their enclosing circles. The radii of the enclosing circles could be increased by a small amount to provide a clear area of buffer space surrounding the obstacle. This would help prevent collisions between the robot and the obstacles. The primary drawback of this method is the waste of useful free space (see Figure 4.1).

A better way to model physical objects would be to use straight line polygonal closed surface approximations. The lowest order polygon possible would be the best choice. Lozano-Perez [15] has done considerable work in this area.

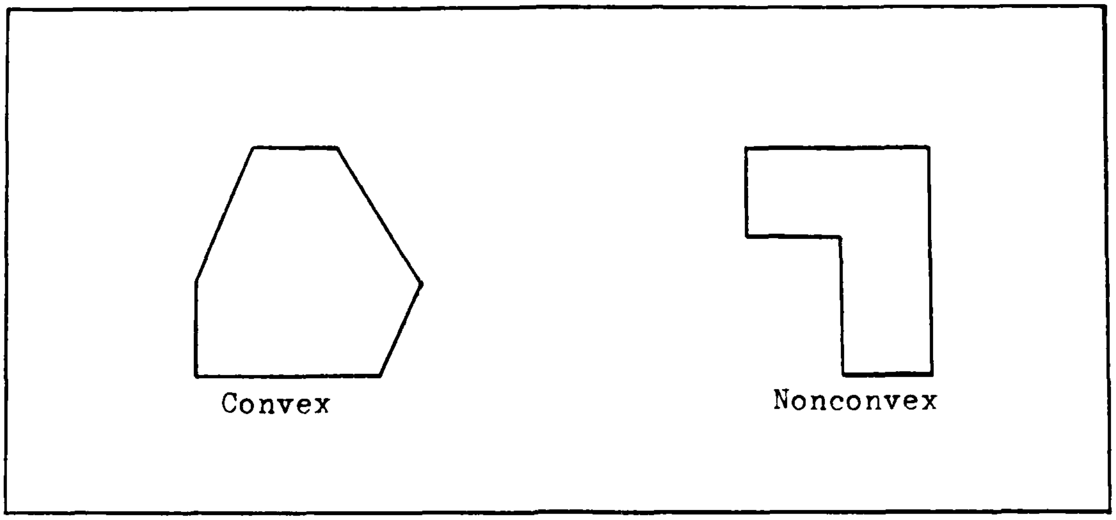


Figure 4.2. Polygon approximations to real world obstacles

He not only chooses to model physical objects as polygons but as convex polygons. A convex polygon is a polygon with no internal angle greater than 180 degrees (see Figure 4.2).

Given that all obstacles are represented as convex polygons, a path can be found around an obstacle by searching for a path around the vertices or corners of the polygon. For example, to go from point A to point B, in Figure 4.3, a path is planned going through each vertice of the polygon obstacle. Only the paths that do not cross the obstacle are considered possible. Either path 1 or 2 could be taken. Both traverse the outside perimeter of the obstacle and result in the shortest paths available Physical objects such as that shown in Figure 4.2 which are not convex in shape are either modeled as convex anyway or modeled as overlapping convex polygons by Lozano-Perez (see Figure 4.4).

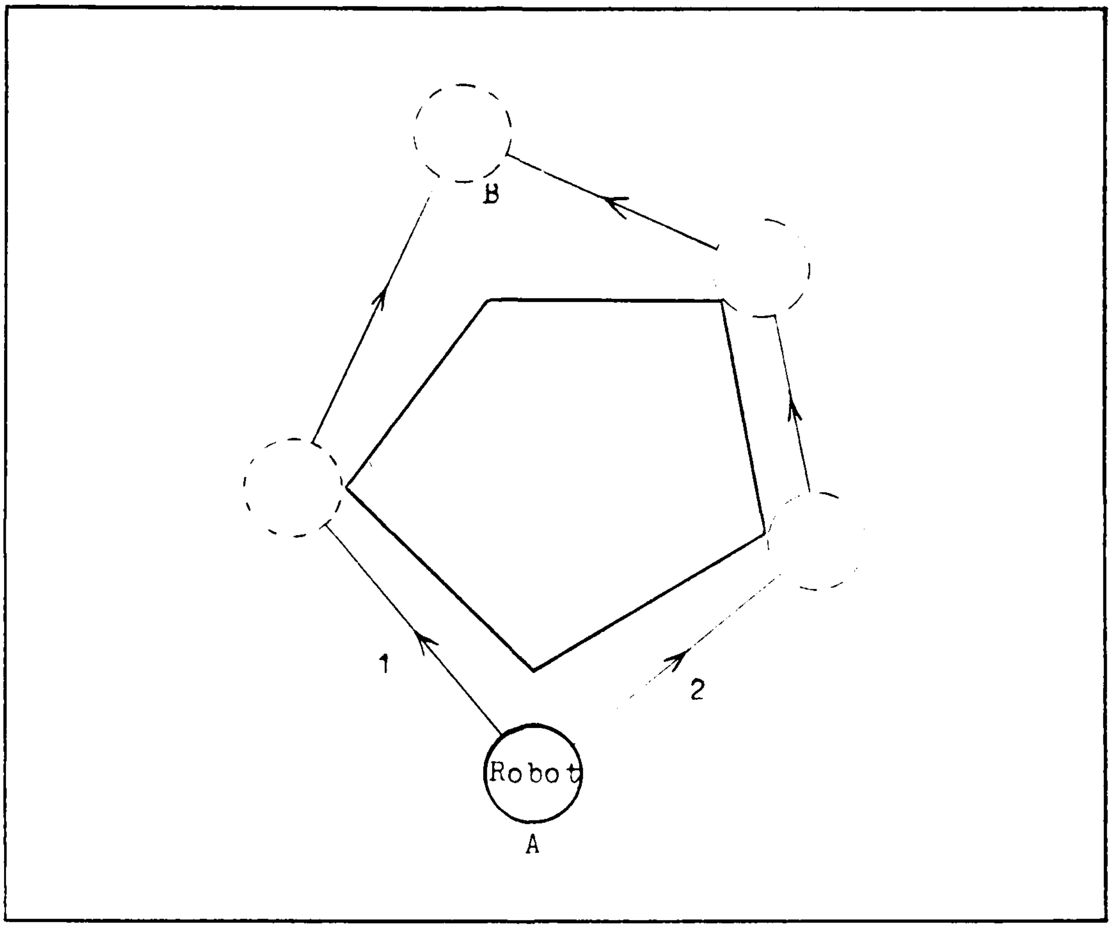


Figure 4.3. Technique of Lozano-Perez for going around an obstacle.

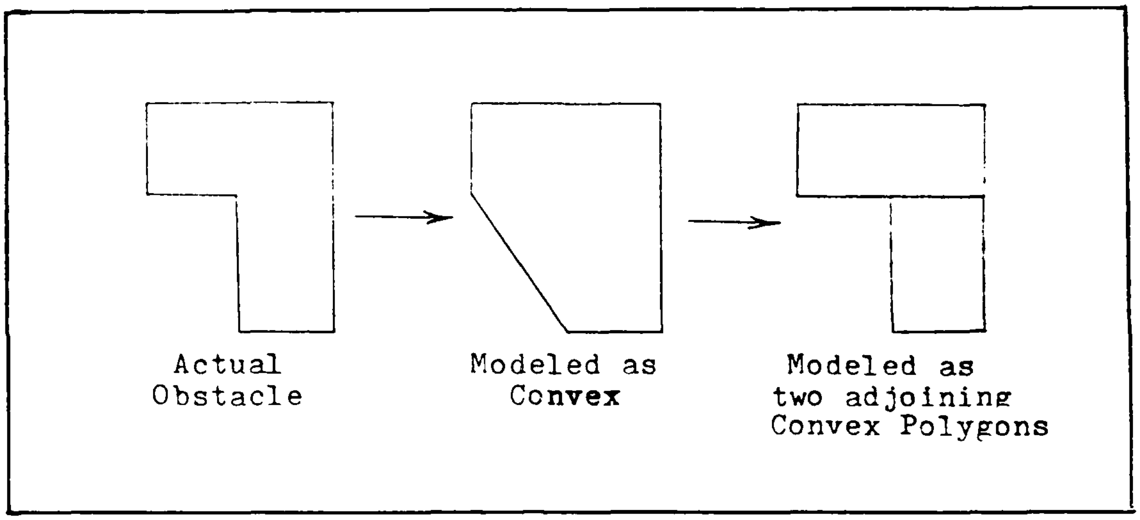
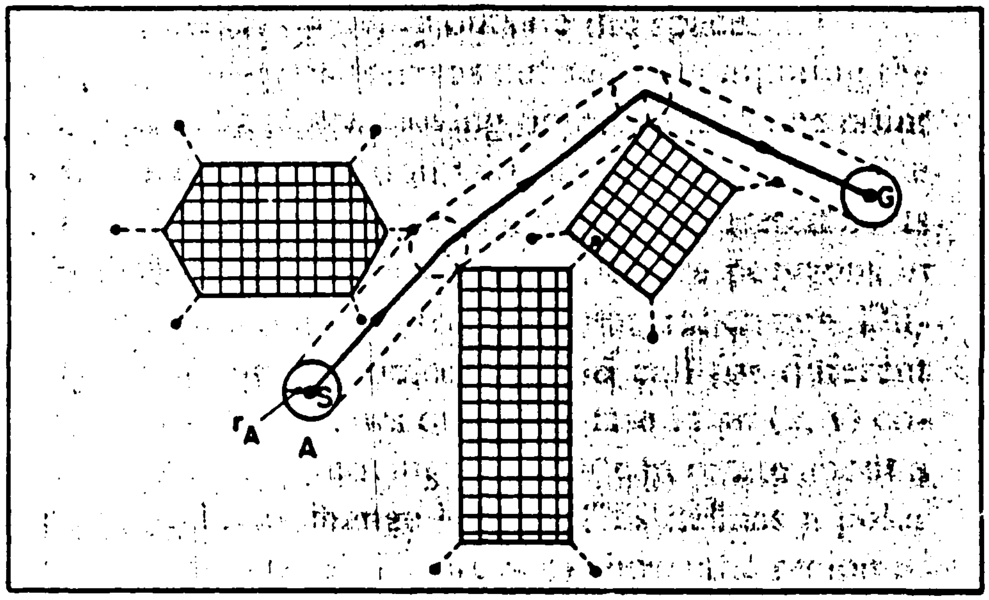


Figure 4.4. Examples of how an obstacle may be modeled using Lozano-Perez technique.

For a circular shaped robot, Lozano-Perez proposes a technique of displacing the vertices of an obstacle by the radius of the robot [15:562]. Thus, the robot can be treated as a point; thereby, greatly simplifying the path finding problem. This technique is illustrated in Figure 4.5. Notice how the robot (now a point) is made to pass through the extended vertices.

The technique of Lozano-Perez has several disadvantages. It can be wastefull of free space and computationally inefficient because physical objects must be modeled as convex polygons. In addition, this technique forces the robot to hug an obstacle as it goes around it. Relatively small errors in the world map or in the navigation data greatly increase the probability of a collision.



A = Robot r_A = radius of Robot

S = Starting location G = Goal location

Figure 4.5. Vertices of all obstacles are extended outward so the robot can be treated as a point. [15:562]

Monaghan [16] also proposes using polygon approximations for obstacles, but does not restrict the polygons to convex shapes only. This results in a better representation of the actual shape of an obstacle with a minimum number of total vertices. He also shrinks the robot to a point mass and enlarges the obstacles by a likewise amount by extending the sides of the polygons (remember Lozano-Perez extended the vertices). Monaghan's path finding technique is similar to that of Lozano-Perez where a

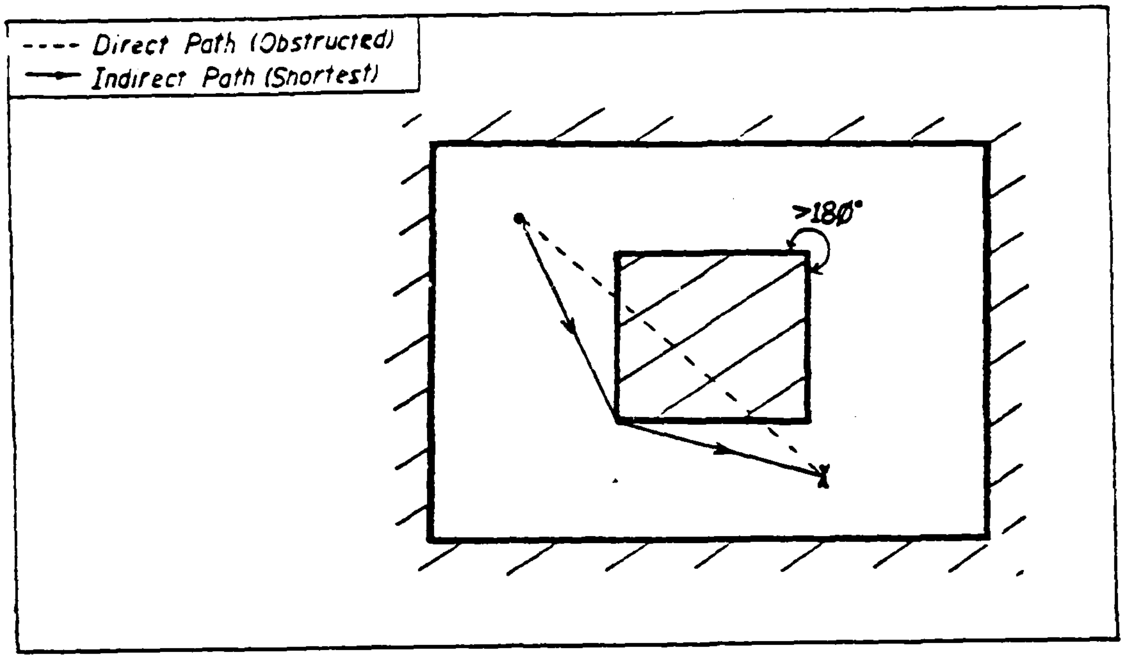


Figure 4.6a. Monaghan's modeling technique [16:47]

search of the vertices of an obstacle is performed to find a way around it. Monaghan's work emphasizes finding the shortest path to the goal point. Thus, a vertice of an obstacle is used as a "way point" as shown in Figure 4.6a. However, an inside corner (resulting from the use of nonconvex obstacles) is never considered as a way point (see Figure 4.6b). This technique does give the shortest path, but it is certainly not the safest (due to possible collision).

So far, the obstacles modeling approach to world modeling has been discussed. We have seen that obstacles may be represented by either their enclosing circle or a polygonal approximation. Another approach to world

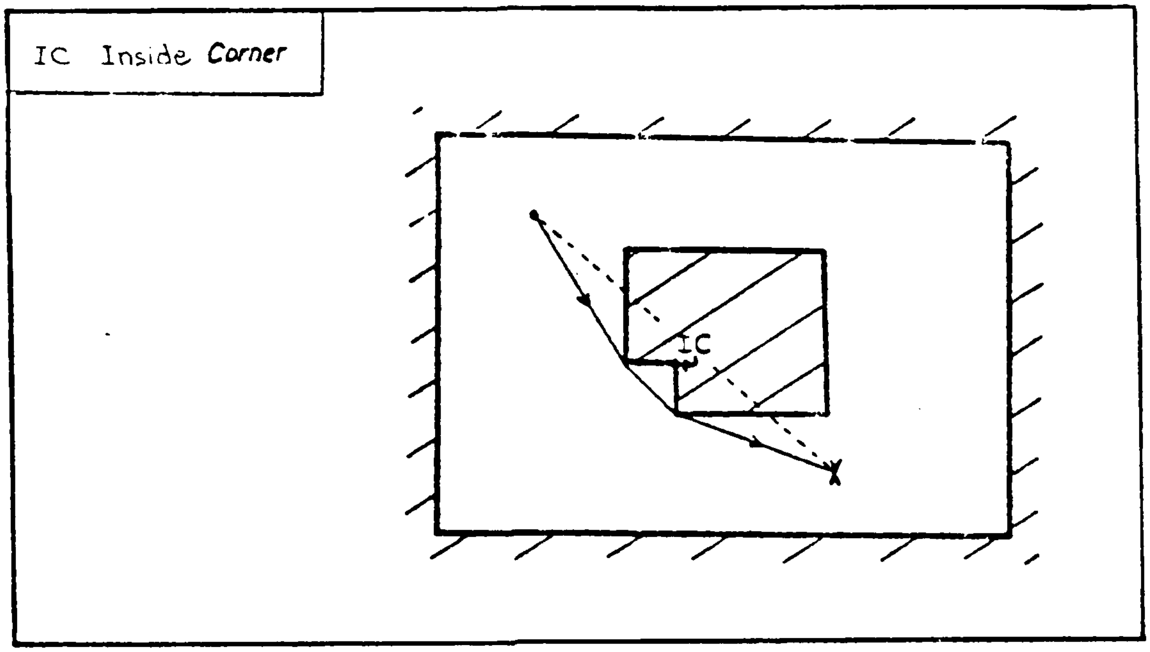


Figure 4.6b. Inside corners are not used as "way points" [16:48]

modeling is to model the free space which a robot may occupy.

Brooks describes the free space which a robot may travel as a network of cones [16:25]. Obstacles are polygon shaped and the free space between the faces of these polygons can be formed into generalized cones or "freeways" (see Figure 4.7). The robot is restricted to travel along the center or "spine" of these cones. This technique is less prone to collision since the robot is required to remain at the centerline of free space. The major problem with this method, as pointed out by Monaghan [16:28], is "the difficulty of modeling the map to account for movement

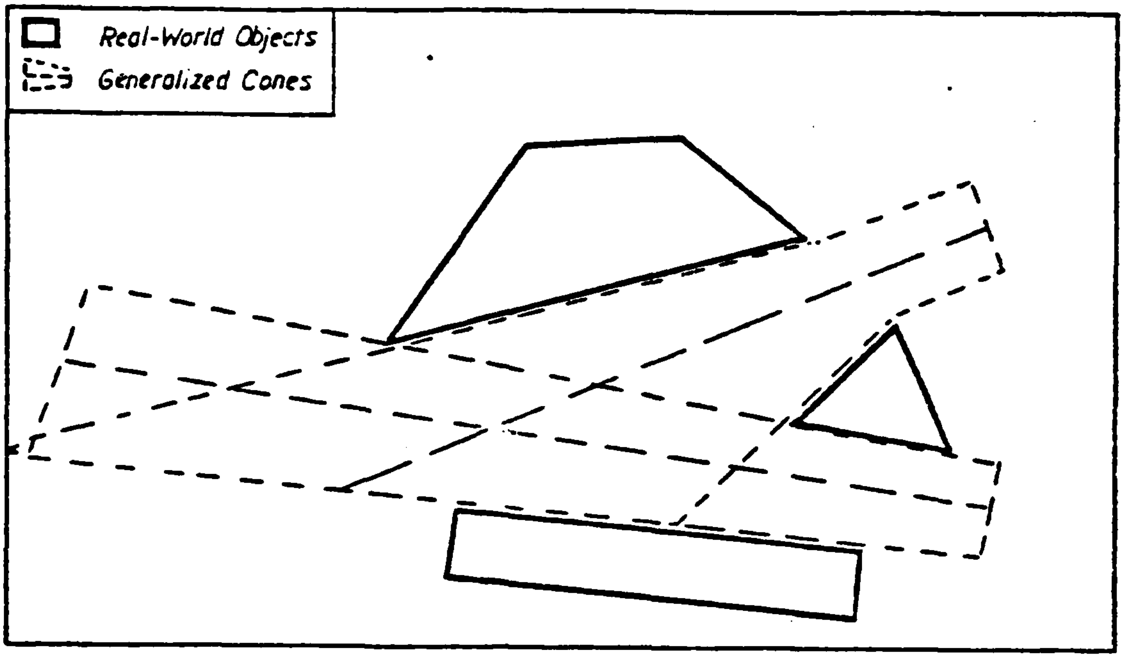


Figure 4.7. Generalized cones form freeways between obstacles [16:27]

of any obstacles. Repositioning a single object could involve comparing each of its faces with all those of every other obstacle to recompute the adjacent free space cones."

Another free space modeling technique directly models the regions through which the robot may travel. This technique is attributed to Crowley [16:28]. Crowley models the free space around objects through a series of convex polygons (see Figure 4.8). It is important to note that any two points within a convex polygon can be connected with an unobstructed line (see Figure 4.9). Thus, movement confined to within the borders of a convex polygon is guaranteed to be collision free. Motion is restricted, however, when it is necessary to travel to other regions (adjacent convex

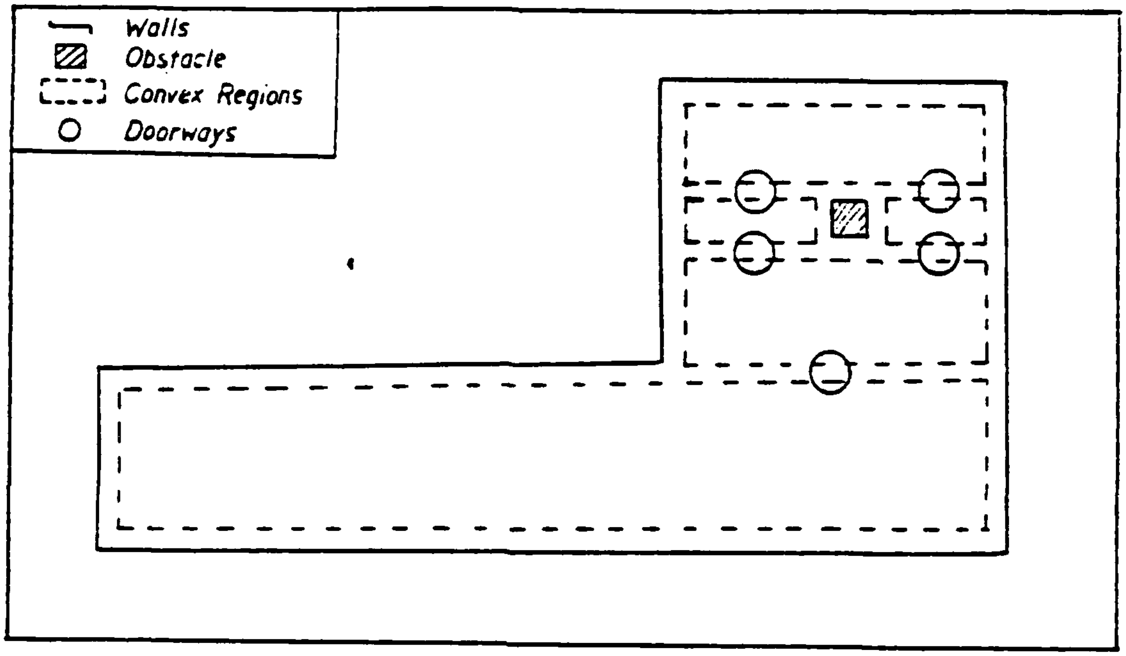


Figure 4.8. Convex regions separated by doorways represent free space [16:29] .

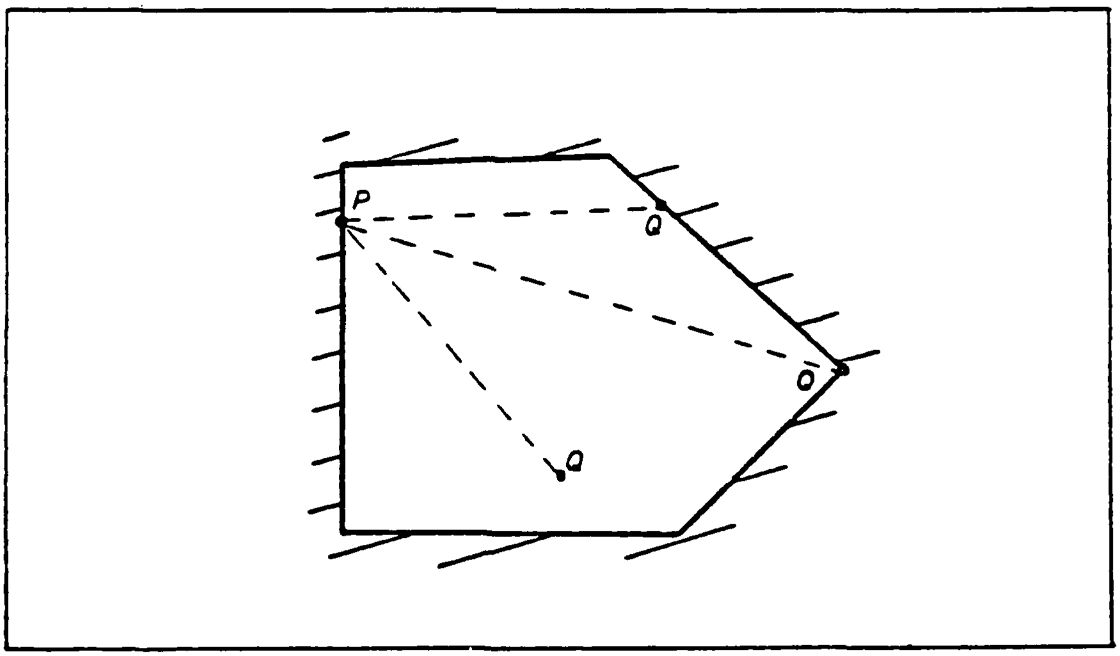


Figure 4.9. Any two points, P and Q, in or on a convex polygon may be connected by an unobstructed straight line [16:45] .

polygons). This can only be done through a defined "doorway" (see Figure 4.8). With this modeling technique, finding a path results in searching the network of doorways and free spaces.

Crowley also treats the robot as a point much the same way Lozano-Perez does. However, while Lozano-Perez enlarges obstacles to account for the robots size, Crowley shrinks his free space by an amount equal to the robot radius. One problem with Crowley's technique can be seen in Figure 4.8. Notice that for just one obstacle, five free space regions must be stored into memory. Also, if an obstacle is moved many free space regions must be recomputed. Crowley's use of doorways, however, is very appealing and will be expounded upon later.

A NEW TECHNIQUE

A brief discussion of the current schools of thought for modeling a robot's world has preceded this section. By combining some of these ideas, a better method can be obtained. Consider the following approach:

1. Obstacles will be modeled as polygons (not just convex).

2. The obstacles will be enlarged so the robot can be treated as a point.

3. Abstract safe points will be established such that at least one safe point can be reached from anywhere in the robots environment.

This method is a combination of obstacle modeling and free

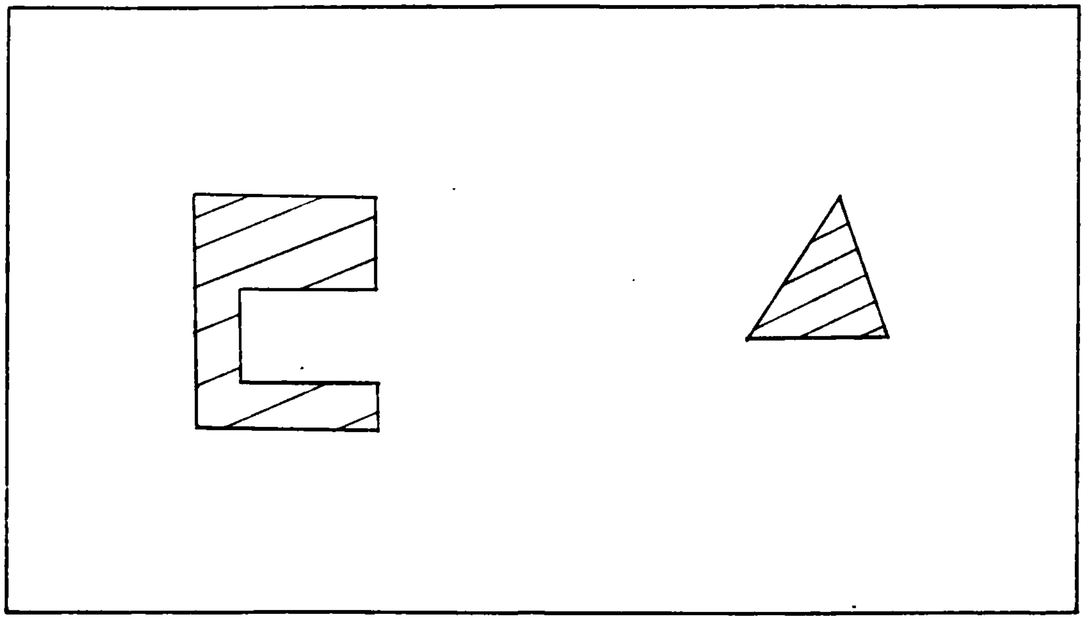


Figure 4.10 Room with two obstacles.

space modeling techniques. For example, in Figure 4.10, a room is depicted with two obstacles. Note that one obstacle is convex in shape and the other is not. Now, a series of doorways can be established much the same way as in Crowley's technique. The free space is divided into adjoining convex polygons as in Figure 4.11. Then, doorways are established between adjacent convex regions. Next, the free space boundaries are removed leaving only the obstacles and the doorways (which are represented as a series of points - see Figure 4.12). These doorway points are called "safe points". If a direct path is obstructed, a search is made of the "safe points" and indirect paths can be obtained as in shown Figure 4.13.

Unlike Crowley's technique, requiring a doorway be used

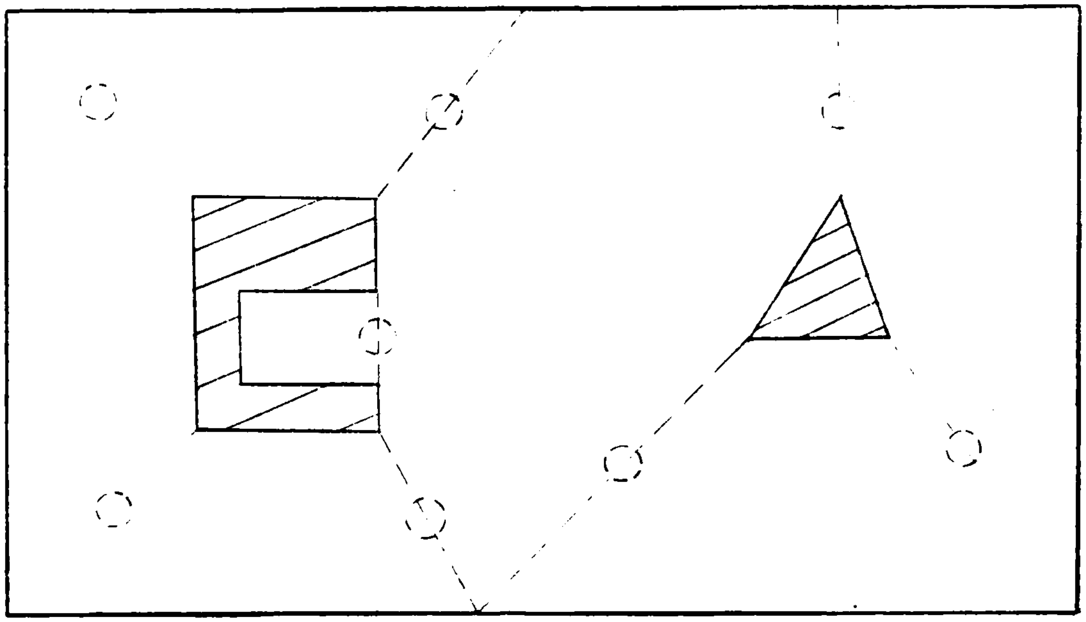


Figure 4.11. Free space is divided into convex regions to define "safe points."

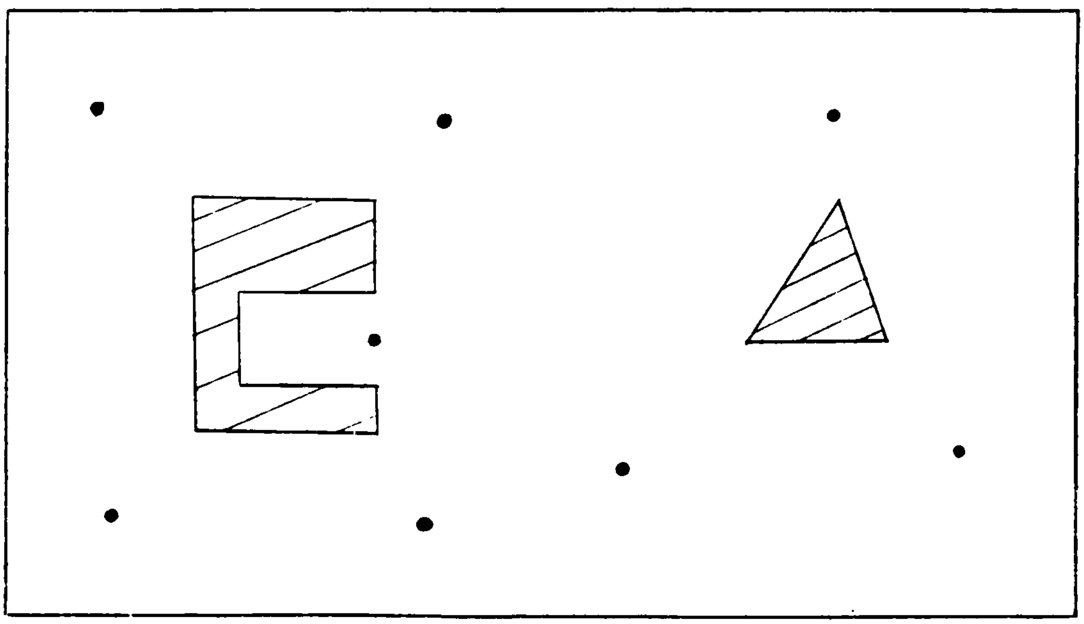


Figure 4.12. Only obstacles and safe points are modeled.

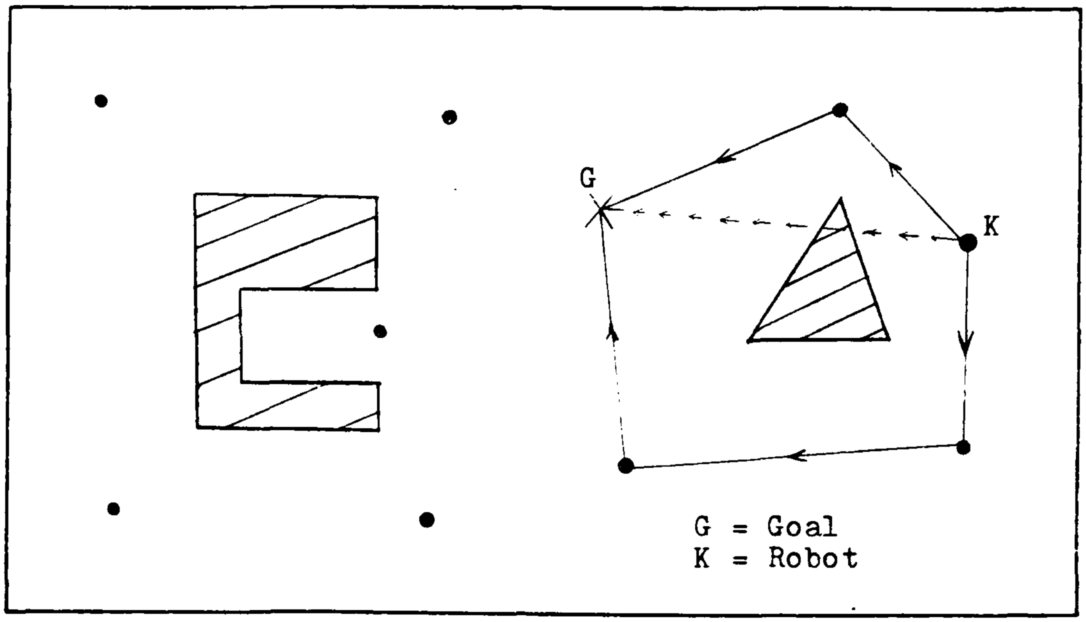


Figure 4.13. Indirect pathways pass through safe points.

as passage between free space regions, this technique uses doorways or safe points only when the goal is obstructed by an obstacle. Direct passage can take place anywhere in the room as long as the pathway is unobstructed.

To avoid having pathways which run very near the side of an obstacle, the free space boundaries must be carefully chosen when establishing safe points. For example, Figure 4.14 shows again the way Crowley separates a room into free space regions. This is a poor choice since it may require sustained travel very near an obstacle or border. Notice how the path to the goal runs parallel to the wall. This increases the chance of collision. As a rule of thumb, free space boundaries should be constructed so they never run parallel to an obstacle face or exterior boundary. Figure

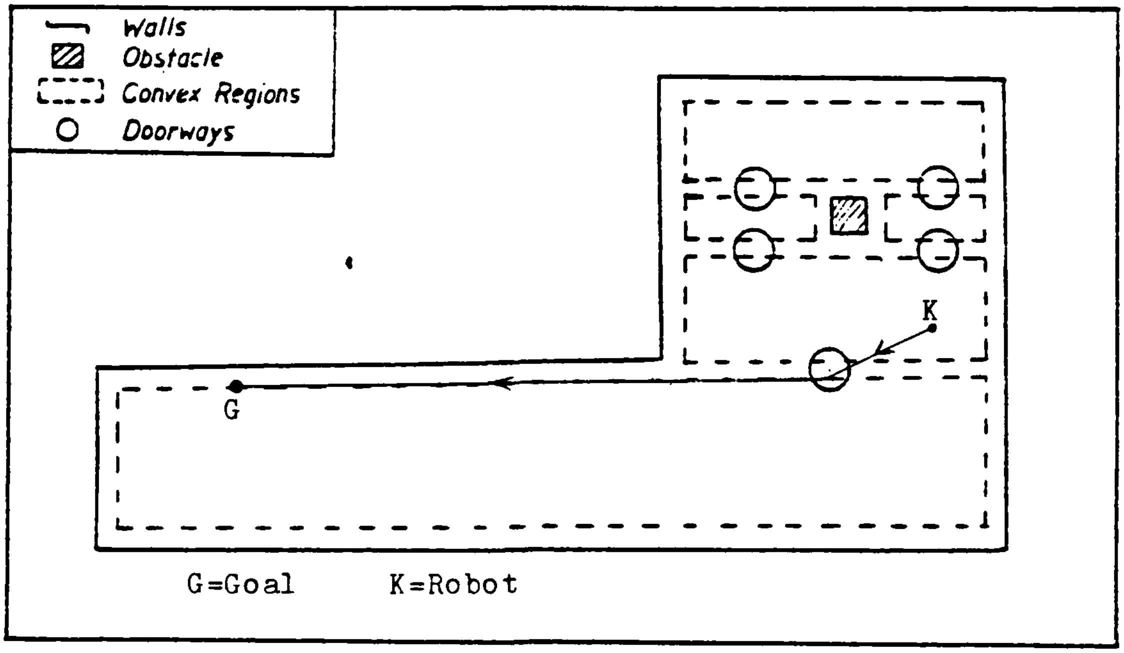


Figure 4.14. Problems occur if free space regions are not chosen correctly.

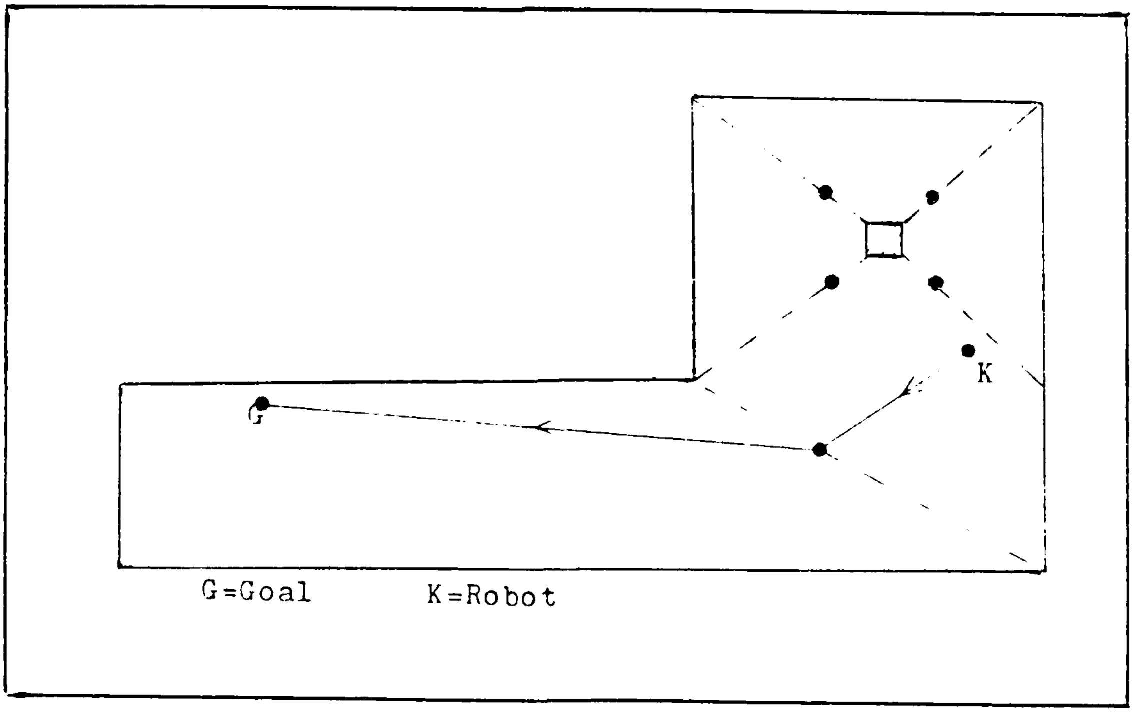


Figure 4.15. Free space regions can be chosen to minimize the probability of collision.

4.15 shows the same room only this time with different free space borders. None of the borders are parallel to an obstacle face or exterior boundary. Figure 4.15 also shows the new path for the same starting and goal points as in Figure 4.14. Notice how the path no longer hugs the wall. By using this rule of thumb, safer pathways can be planned.

This new technique offers several significant advantages. By using regular polygons to model obstacles, an accurate representation of the actual physical object can be obtained, wasting little or no free space. Treating the robot as a point precludes having to consider the volume of space occupied by the robot. Using "safe points" to plan paths around obstacles keeps the robot a safe distance from obstructions. Thus, fewer collisions should occur. Above all, this method is simple and requires minimum computer memory.

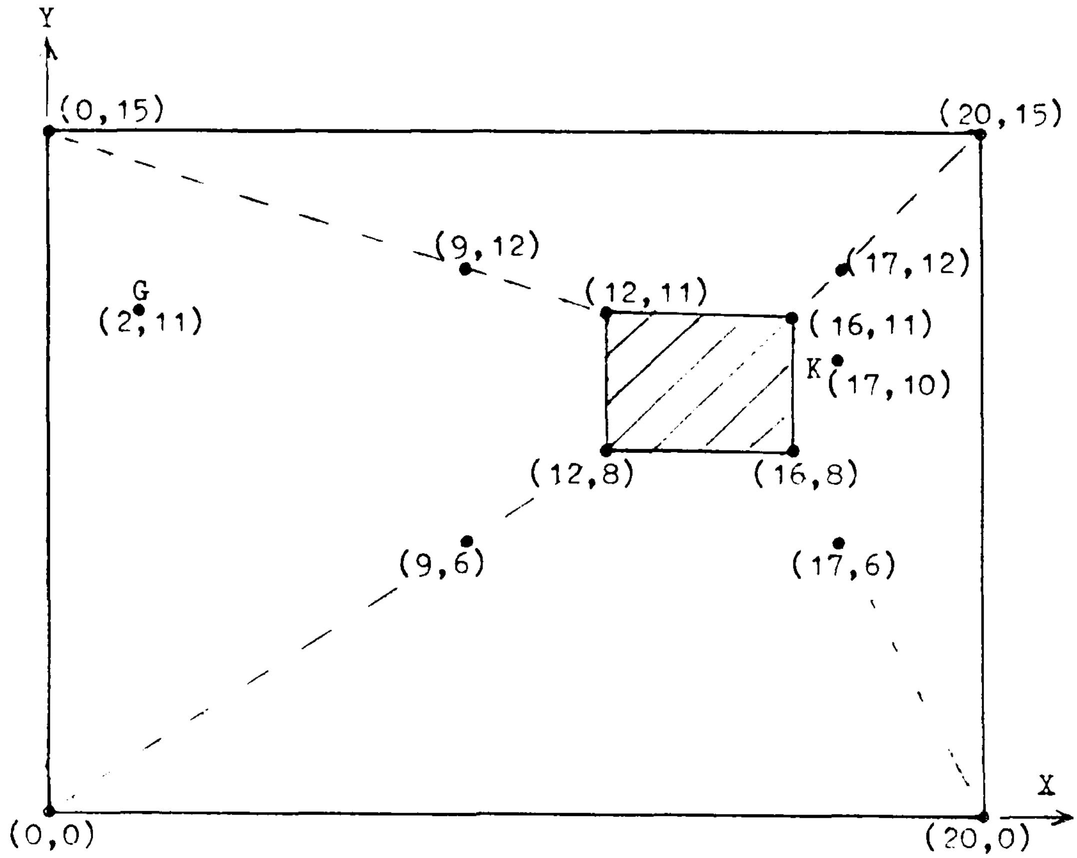
This technique also has a few disadvantages. It could be argued that not yielding the shortest path is a disadvantage. However, for a robot not under a tight energy or time constraint, the shortest route is not necessarily the best. Safety may be more important. When the world model becomes very complex, some other disadvantages appear. If an obstacle is moved, several safe points may have to be recomputed. Also, as the number of obstacles increases, the number of safe points goes up almost exponentially resulting in heavy computational loading.

DETAILED PATH PLANNING

World modeling and path planning are highly dependent upon each other. Path planning cannot take place until a world model has been determined and the best world model is one that provides for the best path planning. In the preceding discussion of world modeling, it was necessary to consider path planning in a general sense. For example, the robot must determine if an obstruction lies in its direct path to the goal. How does the robot do this? How does the robot determine the best indirect path if an obstruction exists? Details of path planning will be discussed in the following section which will answer these questions.

The world model is stored in the robots memory as an ordered list of points. All of the points (X,Y) are relative to the same reference system. Each obstacle is described by an ordered list of its vertices. Also, the vertices of all exterior boundaries are stored (to the robot, exterior boundaries are just more obstacles). Safe points are stored as a separate list of points. Thus, a simple room can be represented as in Figure 4.16.

Assume that the robot is located at (17,10) and the goal is located at (2,11) as depicted in Figure 4.16. Notice, that the direct path to the goal is obstructed. This is obvious to us, but how does the robot know this? Before answering this question lets review some geometry. Figure 4.17 shows a line segment connecting the points K and



Exterior Borders	Obstacles	Safe Points
(0,0)	(12,8)	(9,6)
(0,15)	(12,11)	(9,12)
(20,15)	(16,11)	(17,12)
(20,0)	(16,8)	(17,6)

Robot's present position, K (17,10)

Goal location, G (2,11)

Figure 4.16. Modeling of a room with one obstacle as a set of points.

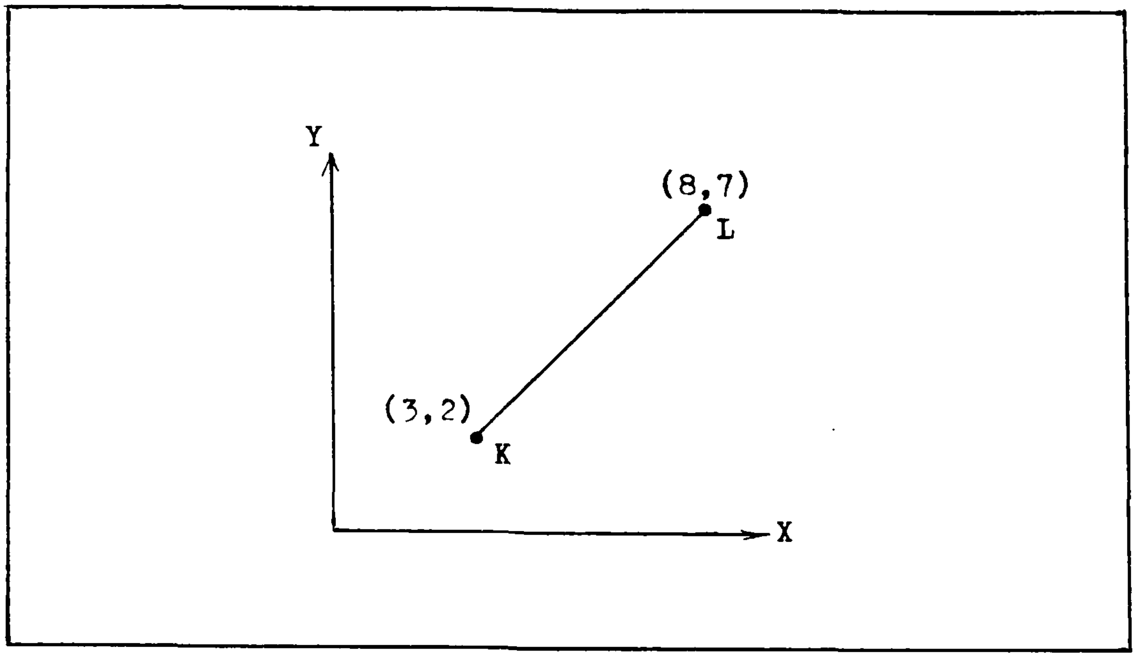


Figure 4.17. Line connecting two points can be represented through parametric equations.

L. This line segment can be represented by the following parametric equations [16:62]:

$$X = X_K + (X_L - X_K)s$$

$$Y = Y_K + (Y_L - Y_K)s \qquad (1)$$

Substituting the coordinates of K and L into the parametric equations results in the following expressions:

$$X = 3 + (8 - 3)s$$

$$Y = 2 + (7 - 2)s$$

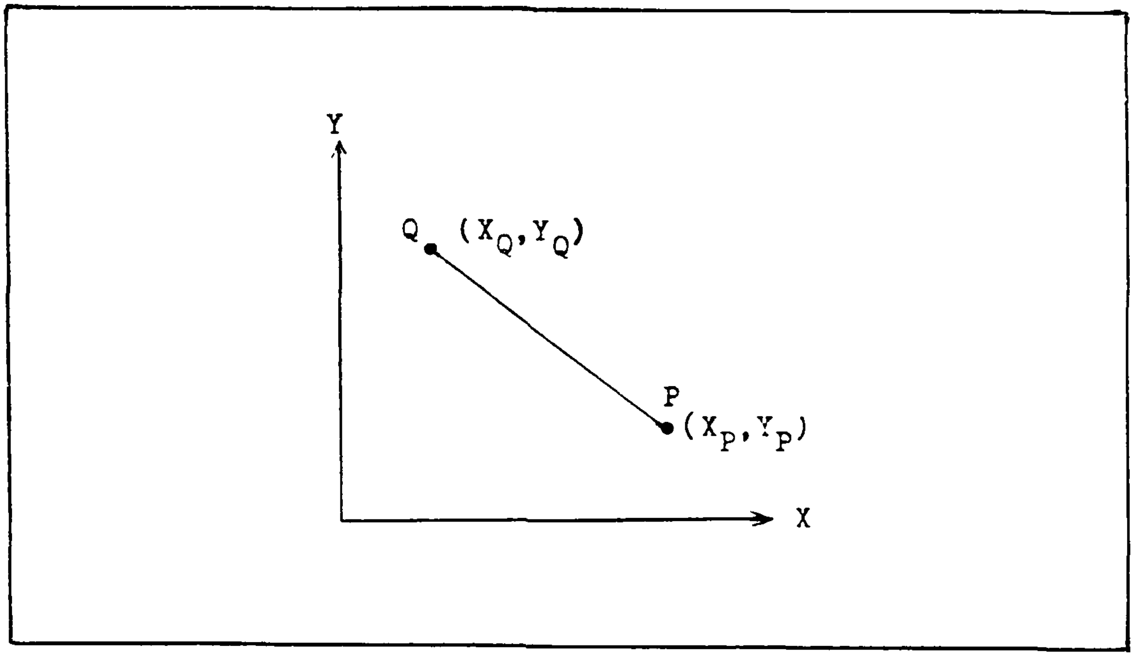


Figure 4.18. Line connecting two arbitrary points.

Simplifying

$$X = 3 + 5s$$

$$Y = 2 + 5s$$

(X,Y) obtained from these equations will always lie on the line segment for s between 0 and 1.

Now consider another set of points P and Q as shown in Figure 4.18. Let this line be represented by the following parametric relations

$$X = X_P + (X_Q - X_P)t$$

$$Y = Y_P + (Y_Q - Y_P)t \qquad (2)$$

where t is the parameter in this case. Again, if t lies between 0 and 1 then (X,Y) is on the line joining P and Q. The parametric relations (1) and (2) can be used to develop a test which can determine if two line segments intersect [3]. Solving the set of equations (1) and (2) simultaneously for the parameters s and t results in the following expressions:

$$
s = \frac{(X_Q - X_P)(Y_P - Y_K) - (Y_Q - Y_P)(X_P - X_K)}{(X_Q - X_P)(Y_L - Y_K) - (Y_Q - Y_P)(Y_L - X_K)}
$$

$$
t = \frac{(X_L - X_K)(Y_P - Y_K) - (Y_L - Y_K)(X_P - X_K)}{(X_Q - X_P)(Y_L - Y_K) - (Y_Q - Y_P)(X_L - X_K)} \qquad (3)
$$

The parameter values, s and t, obtained from the above expressions can be used to determine if two lines intersect. Two lines intersect only if both s and t take on values between 0 and 1. This test will hereafter be refered to as the parameter test.

To determine if an obstacle lies in the direct path of the robot, the parameter test is performed. The robots location and the goal point form one set of points (K and L). The vertices (P and Q) of each obstacle are then used, one pair at a time, to determine if an intersection exists. All obstacles or obstacle faces may not need to be checked

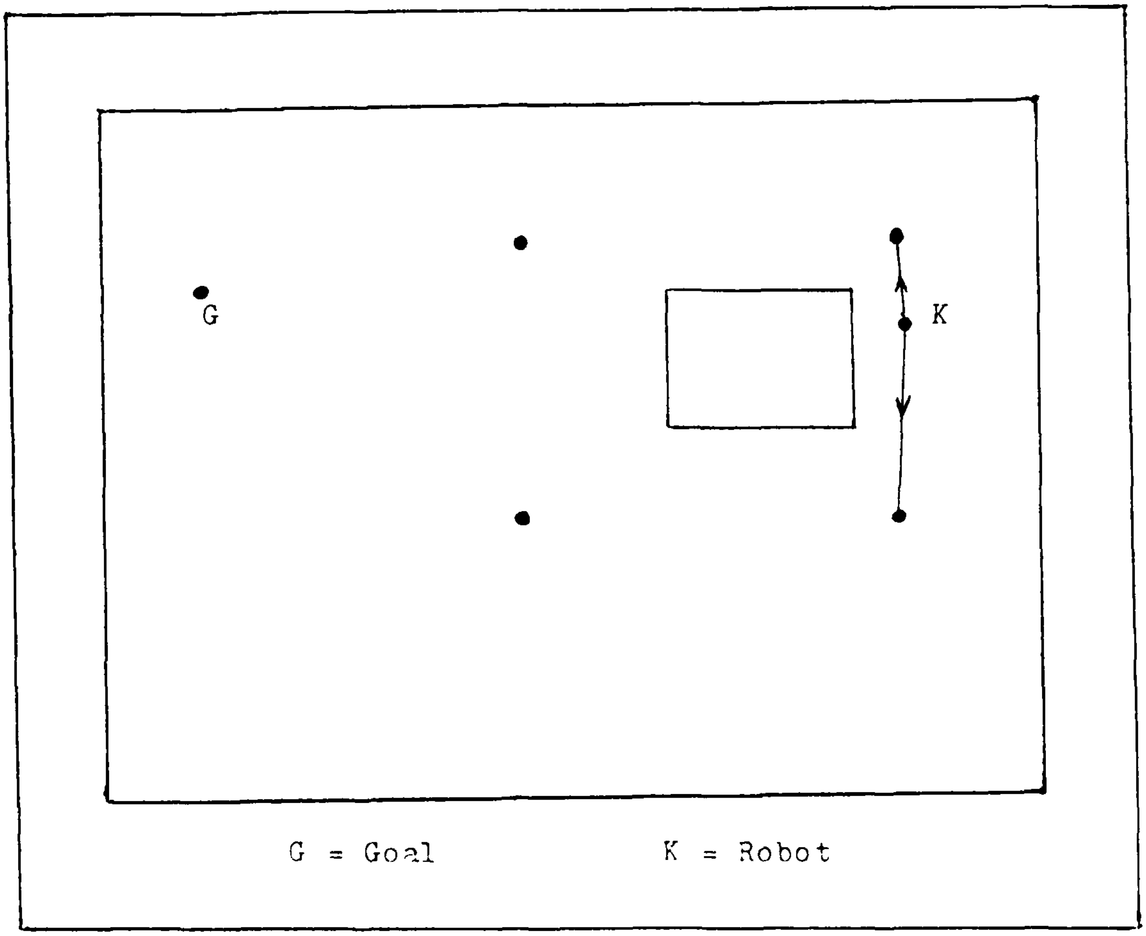


Figure 4.19 Two safe points can be reached through a direct path from the robots present position.

for intersections. No more tests are needed once the first intersection is found. Of course, if no intersections exist then the robot has a clear direct path. If an intersection is found, the robot must determine an indirect path.

To determine an indirect path to the goal point, the robot must perform a search through all the safe points and determine which ones he has direct access to. The parameter test is again used to eliminate the safe points with direct path obstructions. For our example (Figure 4.19), two safe

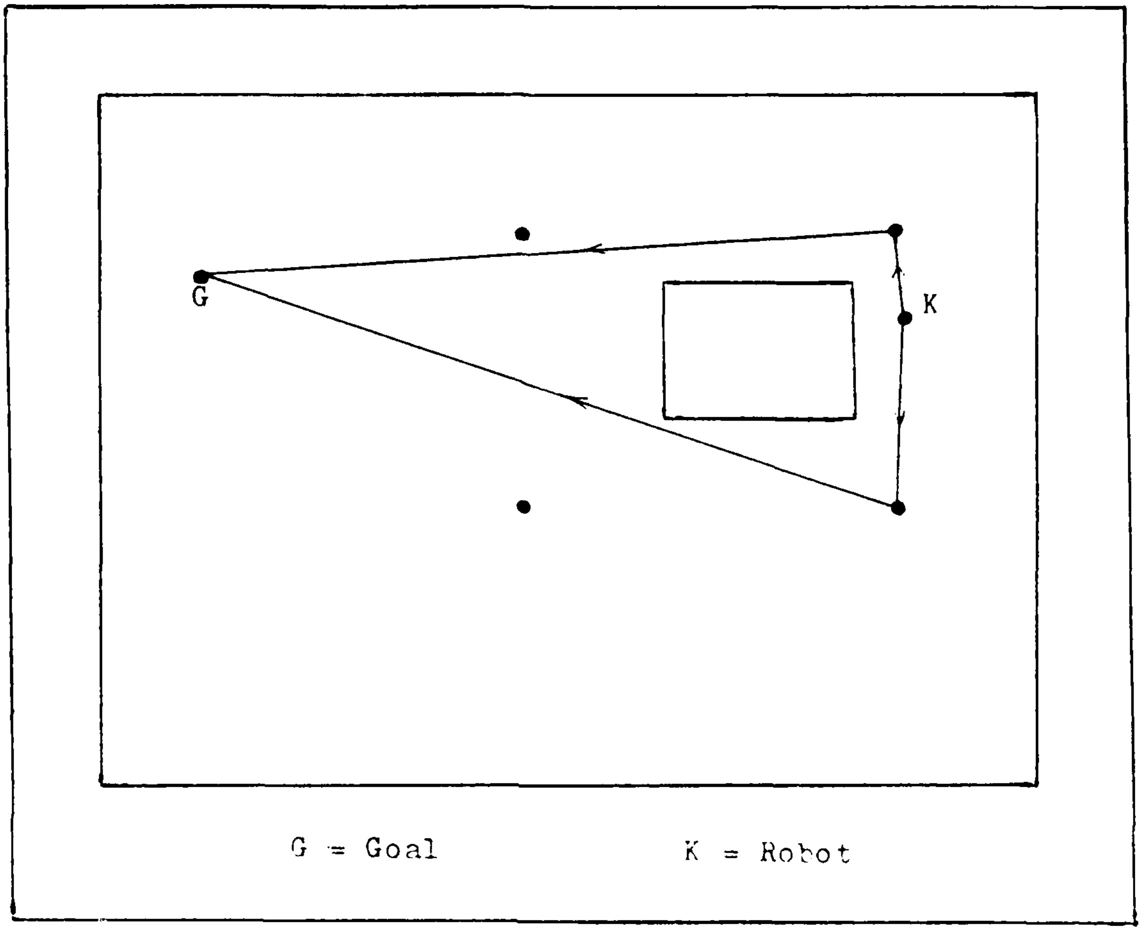


Figure 4.20. Two safe point paths lead to the goal.

points can be reached by a direct path from the robots present location. From each of these reachable safe points, a direct path to the goal is checked for obstacles (again using the parameter test). If no direct paths exist, another safe point must be found. For our example, this is not necessary since the goal can be reached directly from either safe point (see Figure 4.20). However, which path should be taken?

To select the "best" path among several possibilities, an optimization test is performed. For each possible path, a cost function is maintained. The optimum path is the one with the lowest cost value. The cost function expression is as follows:

$$\begin{aligned} COST = {} & (X_K - X_{P1})^2 + (Y_K - Y_{P1})^2 + (X_{P1} - X_{P2})^2 \\ & + (Y_{P1} - Y_{P2})^2 + \ldots + (X_{Pn-1} - X_{Pn})^2 \\ & + (Y_{Pn-1} - Y_{Pn})^2 + (X_{Pn} - X_G)^2 + (Y_{Pn} - Y_G)^2 \end{aligned}$$

where K = Starting Point
P = Safe Point
G = Goal
n = number of safe points used

This cost function is merely the sum of the distances squared of each leg of the path. Thus, the optimum path is the shortest path.

CONCLUSION

World modeling and path planning represent only a portion of the general robot navigation problem. However, their importance to the realization of an autonomous mobile robot system should not be taken lightly. Before a robot can begin to move, it must have some knowledge of its environment and it must be able to plan out a collision free route through its environment. This problem has received the recent attention of several researchers. Some of the current techniques of world modeling have been presented along with a new approach. Path planning, under this new approach, consists of finding an unobstructed pathway to the goal point. Safe points are used only if a direct path does not exist. The details of this path planning have been developed through a simple example.

V. Testing, Analysis, and Results

The testing of the GYRAC system was divided into three primary phases. The goal of each of these phases is listed below:

<u>Phase I.</u> Verify the functionality of the GYRAC system.

<u>Phase II.</u> Determine if the data from the GYRAC can be used to accurately track the location of the robot (MARRS-1) as it moves about the test area.

<u>Phase III.</u> Demonstrate the capability of MARRS-1 to use GYRAC heading data to follow a programmed heading exercising closed loop steering control.

Phase I

The primary thrust of this phase was to verify that every part of the GYRAC system operated properly. This turned out to be a tremendous task consuming a substantial portion of the allotted thesis time.

For Phase I testing, the GYRAC was connected to an H89 computer through an RS-232 interface and interrogated via M72 modem software. A logic state analyzer, and oscilloscope (see Appendix L) were used to troubleshoot the GYRAC hardware, firmware and verify correct operation of the GYRAC computer. Excluding the accelerometer subsystem, all hardware and software was eventually verified to function exactly as planned. The accelerometer subsystem could not be completely verified until subjected to motion. However, under static test several problems were encountered.

Output from the accelerometer integrator circuit was continually changing. Within a few minutes after power-up of the GYRAC system, the integrator output would become saturated. Operational amplifier (op amp) integrator circuits of this type, operating normally, eventually integrate into saturation under a constant input. However, the rate at which the ouput from the GYRAC integrator increased was much faster than anticipated. The input to the integrator (output from the accelerometer) was not constant, due to the extreme sensitivity of the accelerometer to movement, but it was very small (about 0.1mv). Such a small input should not cause saturation of the integrator so rapidly. An identical integrator circuit was breadboarded for testing.

The breadboarded integrator circuit was tested without an input (zero input voltage) and within a few minutes after power up it would integrate into saturation (just like the actual circuit). This was unexpected. After consulting with Analog Devices Corporation, it was discovered that the observed drift rate could be modeled mathematically through the following equation:

$$R = \frac{I_B}{C}$$

where R = drift rate
I_B = current bias of operational amplifier
C = capacitance of integrator feedback

It can be seen from the above equation that in order to decrease the drift rate, it is necessary to decrease the current bias of the op amp or increase the capacitance in the circuit, or both. The AD544 op amp (see Appendix A), manufactured by Analog Devices, was selected as a replacement due to its low current bias of 10 picoamps. Also, the integrator circuit was redesigned to contain a higher capacitance. Both a 200 microfarad and a 2000 microfarad capacitor were ordered. After obtaining the capacitors, they were measured for actual capacitance and resistors were chosen to achieve the appropriate gain for the integrator circuit. The 2000 microfarad capacitor was selected for installation into the GYRAC due to the very low drift rate achieved in the test circuit with this capacitor. The lowest possible drift rate was desired since the input signal to the integrator circuit is also very small.

After installation into the GYRAC, the accelerometer circuit was again tested. The results were much better than originally obtained. However, the output from the integrator was erratic and inconsistent. Through a process of elimination, another problem was found. The active CMOS switch used to reset the integrator (through software) was leaking current into the integrator circuit and charging the capacitor thereby causing the inconsistent and erratic output. The switch is presently disconnected from the circuit and a manual reset of the integrator must be performed by physically shorting across the capacitor.

At the end of Phase I testing, the accelerometer subsystem appeared to be functioning correctly. The GYRAC system was ready for Phase II.

Phase II

The objective of Phase II was to collect GYRAC heading and velocity data while the GYRAC was under motion and post process the data to determine the location of the robot (MARRS-1) in the test area. The computed location of the robot could then compared with the actual location to test the performance of the GYRAC system.

For this phase of testing, the GYRAC was fully integrated with the MARRS-1 test bed. A memory overlay program, GTEST.A, was created to take advantage of firmware already operating inside the NAV computer. Clifford and Schneider had produced NAV computer firmware for collecting data from the various optical shaft encoders and sonars onboard the MARRS-1. [10:B-1] The overlay program replaces Clifford and Schneider routines for gathering sonar data with routines for gathering GYRAC data. See Appendix G for GTEST.A structure charts, program listing, and operating instructions.

All data are received through an H89 computer via M72 modem software and are then stored on floppy disk. The collected data are in exactly the same format as Clifford/Schneider data [10:IV-11] with the GYRAC data in place of the sonar data.

The raw GYRAC data is in hexidecimal format so an MBASIC program called CONVERT (see Appendix I for listing) was created to convert the raw data to integer format. The integer data is then used in another MBASIC program called POSITION (see Appendix I for listing) which computes the position of the robot in the test area based on the GYRAC heading and velocity data and a given initial position. See Appendix J for a sample output from the POSITION program. The computed position is in terms of a cartesian coordinate system centered in a corner of the test area as shown in Figure 5.1. The GYRAC is aligned such that heading is referenced to zero degrees along the x-axis and increases in the counter-clockwise direction, right handed system.

After several test runs with consistent but unusual results, the accelerometer subsystem was again suspect. The computed position of the robot indicated almost no movement, see Figure 5.2. The velocity levels gathered from the GYRAC were much too small. Eventually, it was discovered that the integrator circuit was loading the internal accelerometer restorer circuit (servo), see Figure 2.11. The result was a total changing of the characteristics of the accelerometer. The voltage sensitivity of the accelerometer, 2 volts/g, was

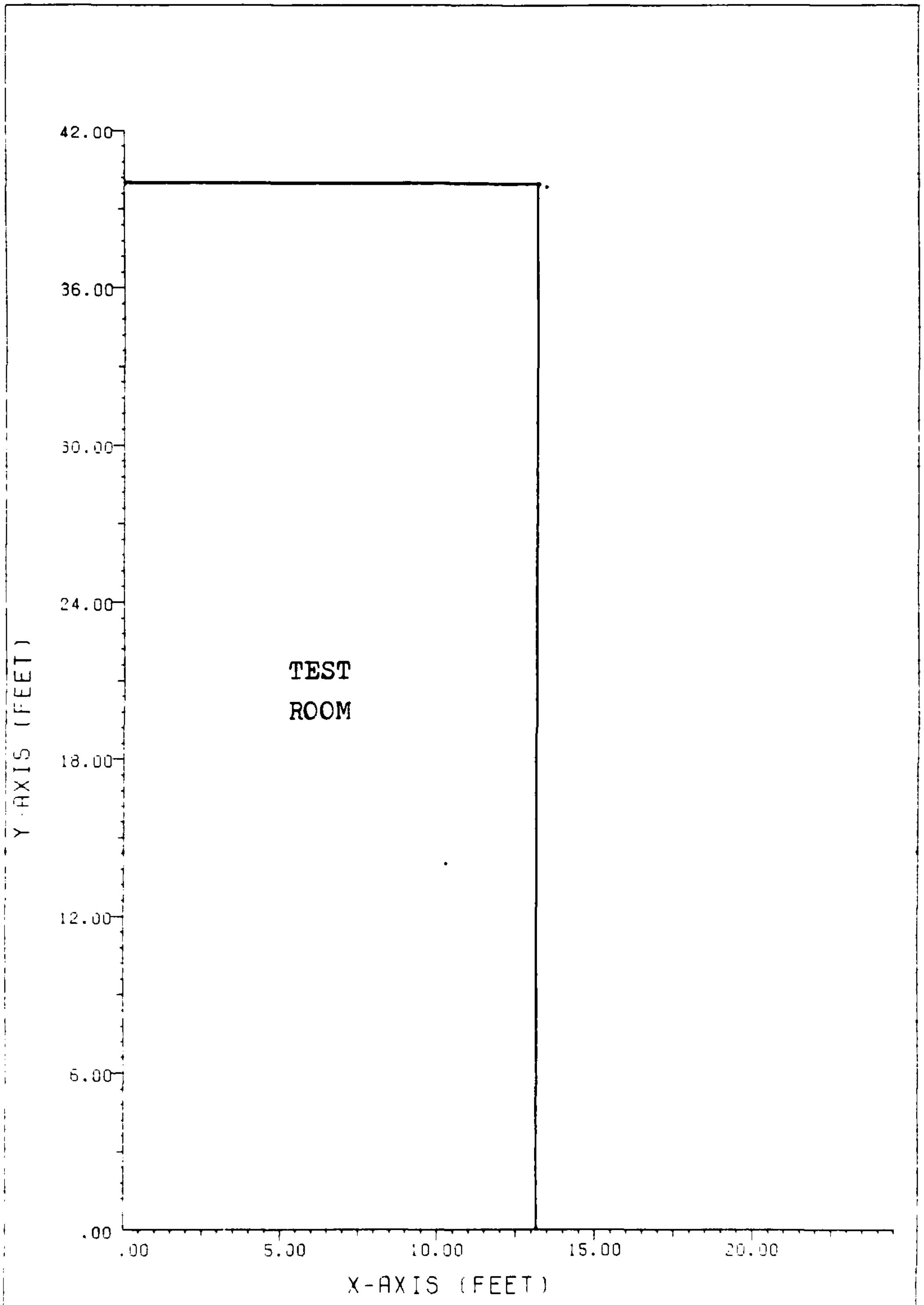


Figure 5.1. Cartesian coordinate system is centered in corner of test area.

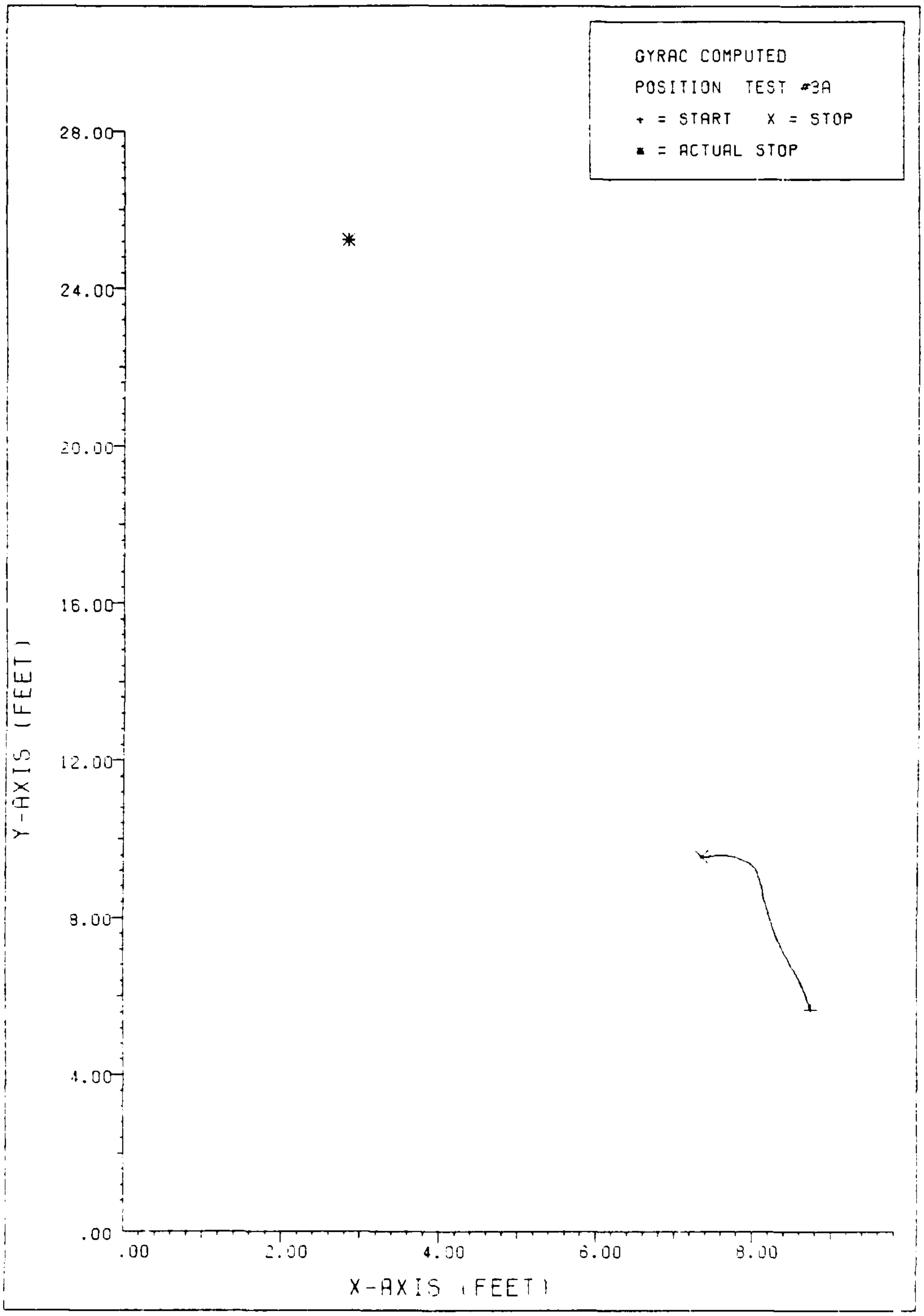


Figure 5.2. GYRAC computed position test using accelerometer sensitivity of 2v/g.

no longer valid. A tumble test of the accelerometer was performed with the accelerometer completely connected to the the rest of the GYRAC, under full electrical load. The new voltage sensitivity was measured to be 0.393 volts/g instead of the 2 volts/g desired. This explained the low velocity levels. However, the accuracy of this newly measured sensitivity was questionable since the total loading effect of the integrator circuit could not be determined. The 0.393volts/g was measured at the load resistor R_L (see Figure 2.11). Using this new sensitivity, further testing resulted in computed positions that were in error on the high side. The computed location of the robot was always downrange from the actual location, see Figure 5.3. This indicated that the actual sensitivity of the accelerometer must be higher. An average sensitivity value of 0.6volts/g was obtained by comparing test runs using the 2v/g sensitivity with those using the 0.393v/g sensitivity. This 0.6v/g sensitivity resulted in computed positions much closer to the actual positions but still only with "ball park" accuracy, see Figure 5.4. In addition, the results were not consistent, sometimes high and sometimes low, almost random. Another problem had been around from the beginning; the output from the integrator (velocity) would not go back to zero after stopping MARRS-1. These problems indicated a possible error in sensed acceleration.

Several tests were performed with only the

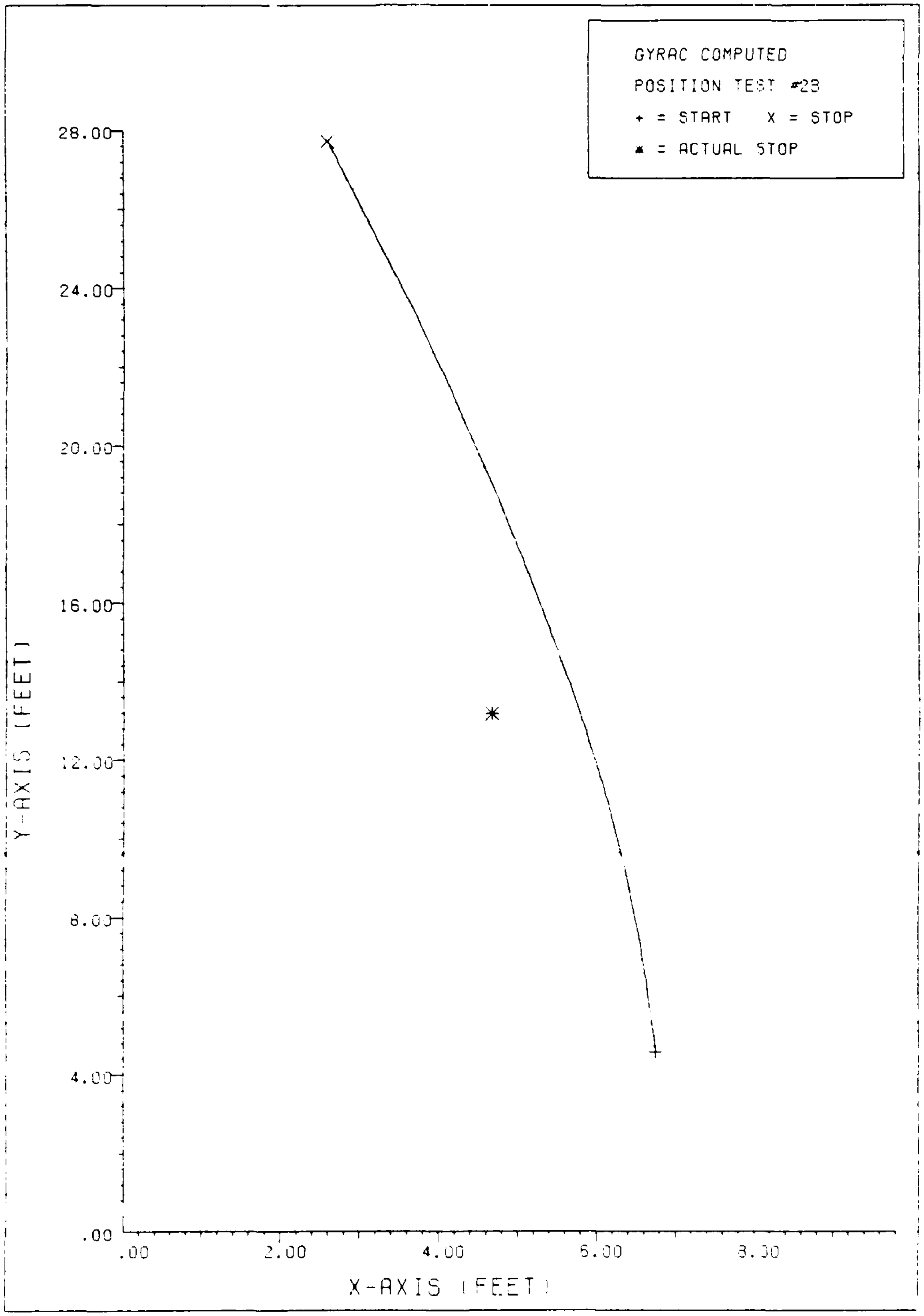


Figure 5.3. GYRAC computed position test using accelerometer sensitivity of 0.393v/g.

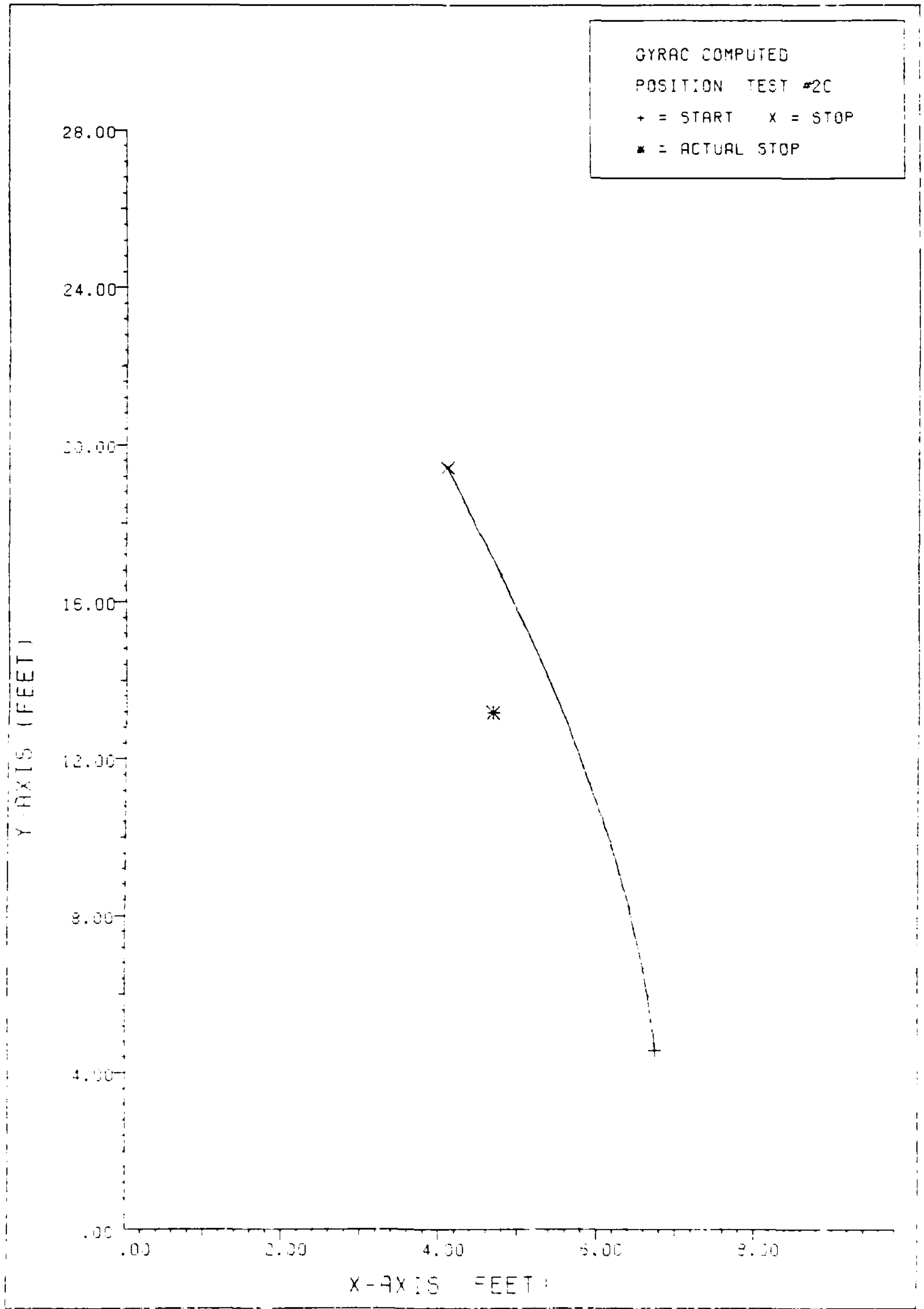
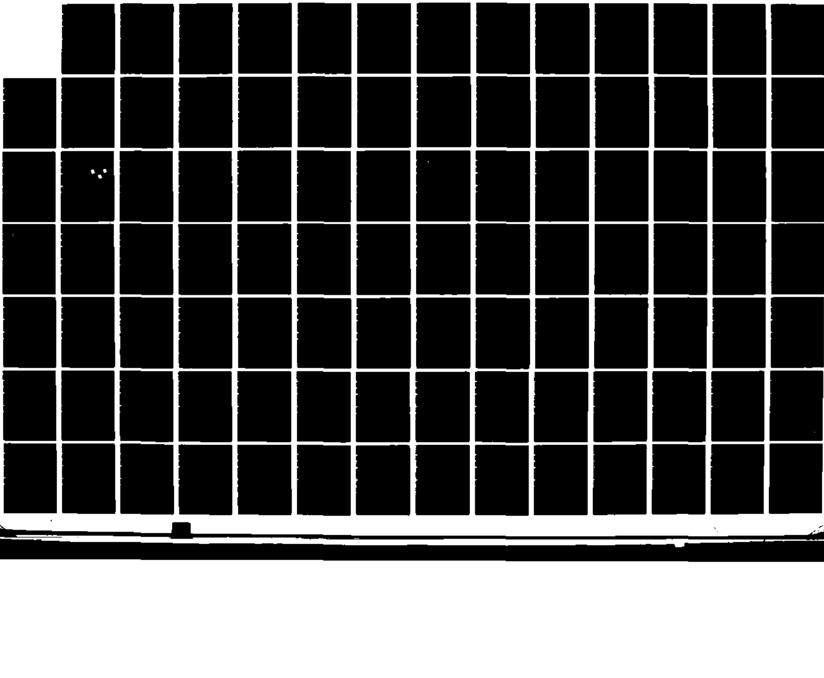
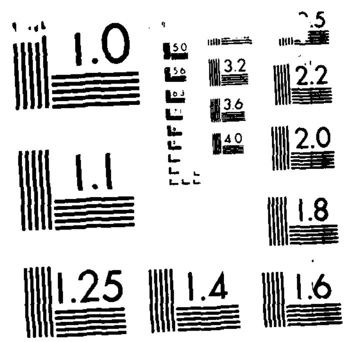


Figure 5.4. GYRAC computed position test using accelerometer sensitivity of 0.6v/g.

AD-A164 036 GYRO AND ACCELEROMETER BASED NAVIGATION SYSTEM FOR A MOBILE AUTONOMOUS RO..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. 2/3

UNCLASSIFIED R J BLOOM ET AL 02 DEC 85 F/G 17/7 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS

accelerometer in the system. The integrator circuit was completely disconnected and the accelerometer output was connected to a gain circuit which was connected to the A/D converter. Pure acceleration data was obtained to determine the levels of acceleration achieved by the MARRS-1 under normal movement about the test area. These tests were very revealing. Figure 5.5 is a plot a acceleration vs time and illustrates the random nature of the sensed acceleration. It shows that the actual acceleration due to motion sensed by the GYRAC is on the same order of magnitude as the sensed acceleration due to tilt error (sensed gravity). In essence, the signal to noise ratio of the system is about one. The MARRS-1 moves at such a slow speed that the actual acceleration never gets much over the noise level. For example, in Figure 5.5, it is not obvious when the robot began movement and when it stopped. In Figure 5.5, the robot actually started forward movement at 2.3 seconds and was at a complete stop at 13.8 seconds. Thus, the acceleration data from the GYRAC and likewise the velocity data cannot be relied upon without some type of error compensation or a stable platform. The assumption of a perfectly smooth and level surface had been violated.

Due to the results from the acceleration tests, it was not necessary to continue Phase II testing. The accelerometer subsystem could never perform adequately without major modifications. Therefore, the third phase of

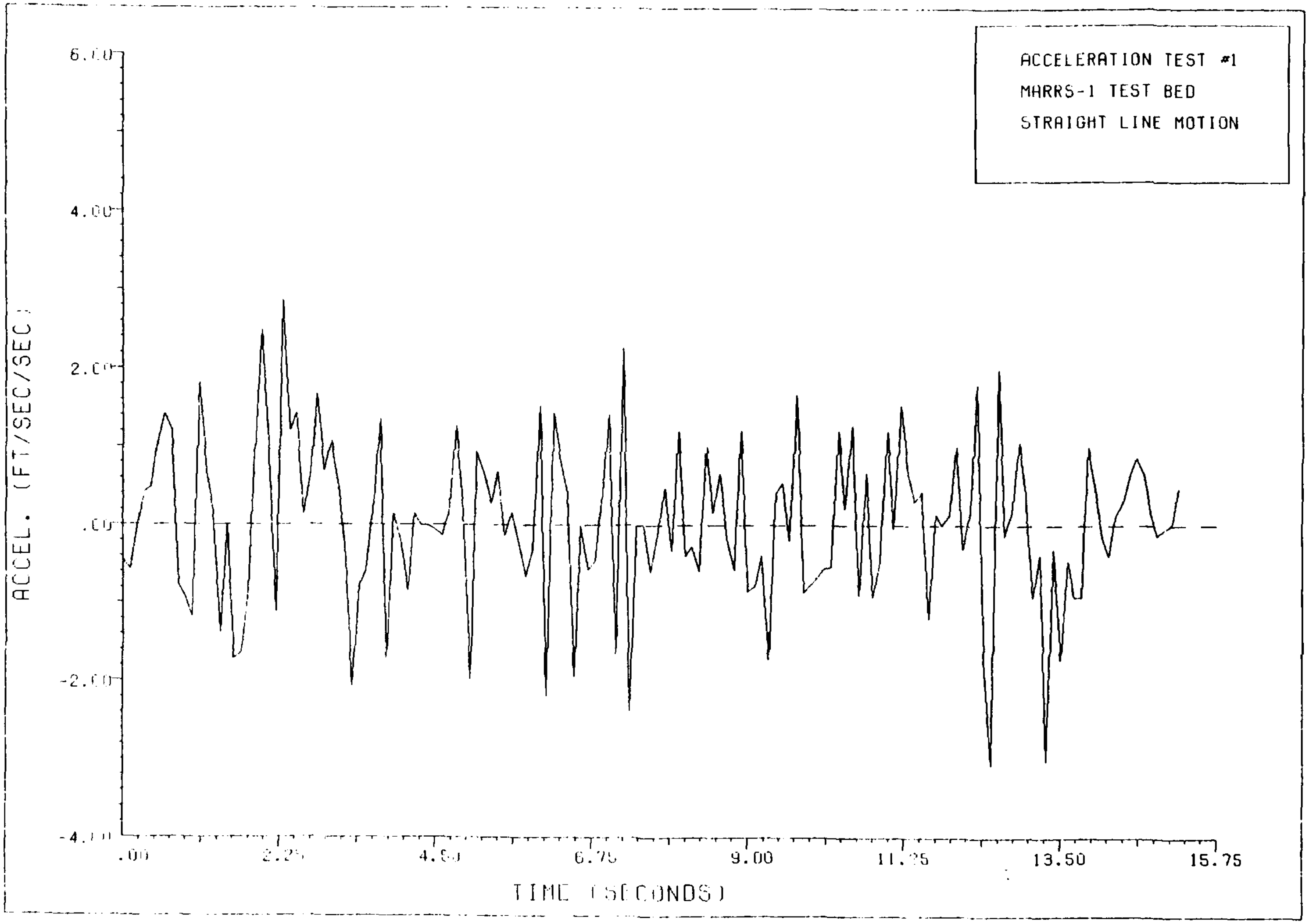


Figure 5.5 Acceleration Test #1.

testing was initiated since it did not require the use of the accelerometer subsystem.

Phase III

The purpose of Phase III testing was to demonstrate the feasibility of using gyro heading data for closed loop control of the MARRS-1 steering motor. This effort produced a navigation program for the Nav computer that requests heading data from the GYRAC system and issues commands to a control program in the drive computer (see Appendix H for listings, structure charts, and operating instructions for both programs). The robot will rotate in place until locked on the specified heading. It then follows the given heading correcting for course errors as it moves until manually stopped. In addition, at each point where a course correction is considered the heading data is transmitted to an external computer for storage and off line analysis (see Appendix K).

Three problems surfaced during the design and testing of this system. First, a communication execution speed problem; second, a steering motor response problem; and third, a steering over correction problem.

Implementation of this system of navigation routines required communication between four different computers: the Nav, GYRAC, Drive, and external computers. The manner in which these communication links and interfaces were implemented have a significant impact on the navigation performance.

The link between the GYRAC and the Nav computer is an RS-232 line operating at 9600 baud. As used in this application, one byte commands are issued by the Nav computer and two bytes of heading data are returned by the GYRAC computer. The communication programs at both ends of the link are written in assembly language to make the link perform efficiently and quickly. This link performed without error and did not significantly slow down the navigation process.

The link between the Nav computer and the external computer is very similar to the GYRAC-Nav computer link. It also performed well and did not slow down the navigation process.

However, the link between the Nav computer and the Drive computer, as implemented, slowed down the course correction process. this resulted in impaired navigation performance. Once again, a 9600 baud RS-232 link was used. However, communication over the link does not use single byte commands and is only driven by assembly language communication routines at the Nav computer end.

To simplify implementation, the decision was made to use the existing Drive computer communication interface and assembly language control routines for the steering and drive motors. The Nav computer controls the operation by sending six bytes of data representing a jump to subroutine command and a specific memory address (ASCII format). Execution of these subroutines by the Drive computer controls the steering and drive motors. Unfortunately, the Drive computer communications software interface requires a small time delay between bytes of data. In addition, each Drive computer motor control subroutine executes a voice command, READY, before returning control of the system back to the communications routine. These two unnecessary time delays limit the Drive computer to at most one command per second which limits the rate at which course corrections can be made.

The command communication problem is further compounded by a slow steering motor response. The steering motor does not move instantly from one position to another. It takes as long as four to five seconds to move 180 degrees. In addition, once the wheel is turned to the desired angle it takes a finite amount of time for this change to produce a measurable course correction. Small changes in wheel direction can produce large changes in robot heading if given sufficient time for movement, but the robot will be off course for this entire time period.

The solution used to alleviate these problems is time delay. Time delays are executed for each steering command to allow the wheel sufficient time to move to the directed position. Small time delays were also added after each course correction to allow time for the wheel direction change to take effect.

A steering over correction problem occurs when the steering wheel is turned for course corrections. The Nav computer is not able to straighten the steering wheel onto the correct heading before the robot has overshot the desired course. This causes the robot to oscillate around the given course resulting in an unstable system where the overcorrections become increasingly large.

This problem has several causes. First, the gyro heading data is measured in increments of approximately 0.088 degrees, but the robot can not set a course to this accuracy since the steering stepper motor moves in one degree increments. Therefore, it must alternately switch between two adjacent steering stepper motor settings to follow most headings.

Second, due to irregularities in the floor and an unbalanced weight distribution of the robot platform over its wheels, the robot drifts from a "straight course" even if the steering wheel is locked in the center position.

Third, course corrections are made in one degree increments each time the heading is sampled and found to deviate from the desired course. If a large course correction must be made, many wheel turn commands will be issued causing a sharp wheel angle to be present when the desired heading is detected. The wheel can only be straightened by many more wheel turn commands in the opposite direction. However, during this time the robot will continue moving in the wrong direction incurring a large overcorrection error.

The overcorrection problem was solved by defining a heading window formed by dropping the least significant four bits of the twelve bit heading reading. This makes the window approximately 1.5 degrees wide with the desired heading located on one of the sixteen possible headings inside the window (unfortunately not centered in the window). Any heading inside the window is defined to be the "correct course". The unstable oscillations are damped by moving the steering wheel to the center position (straight ahead) with a single command as soon as the edge of the heading window is detected. Detected headings within the heading window do not produce a course correction, but allow the robot to continue moving straight ahead (steering wheel centered).

Additional time delays have been provided after each course correction to allow small steering changes more time to take effect. This works well only if the robot begins its movement within or near the heading window. To insure that this always occurs, a rotate-robot-to-heading-window routine is executed before following the directed heading. This aligns the robot's heading within the heading window before forward motion is started.

Figure 5.6 graphically shows the heading data for a 33 foot robot run where the robot's initial heading was within the heading window. No rotation occurred since the initial heading was 88.15 degrees which is inside the heading window. The heading samples are not evenly distributed in time, but occur at course correction decision points. Notice that the robot still oscillates around the given heading, but the oscillations are damped making the navigation system stable. This figure also shows that few detected headings are in the heading window which produces many course corrections and therefore small oscillations around the window.

Figure 5.7 shows a 33 foot robot run where the initial heading was not within the heading window. An initial heading of 66.53 degrees is not shown on the graph since the robot rotated without forward motion to 88.15 degrees which is the first point shown in the figure. Notice that the rotate routine aligned the robot's heading within the heading window. Figure 5.7 also shows the course tracking accuracy that can be obtained when the navigation system "locks on" to a course inside the window.

As in the previous figure, Figure 5.8 does not show the initial heading of 112.67 degrees. The robot automatically aligned itself inside the heading window by rotating to a heading of 87.54 degrees. Notice that a large number of the heading points are again outside the window resulting in

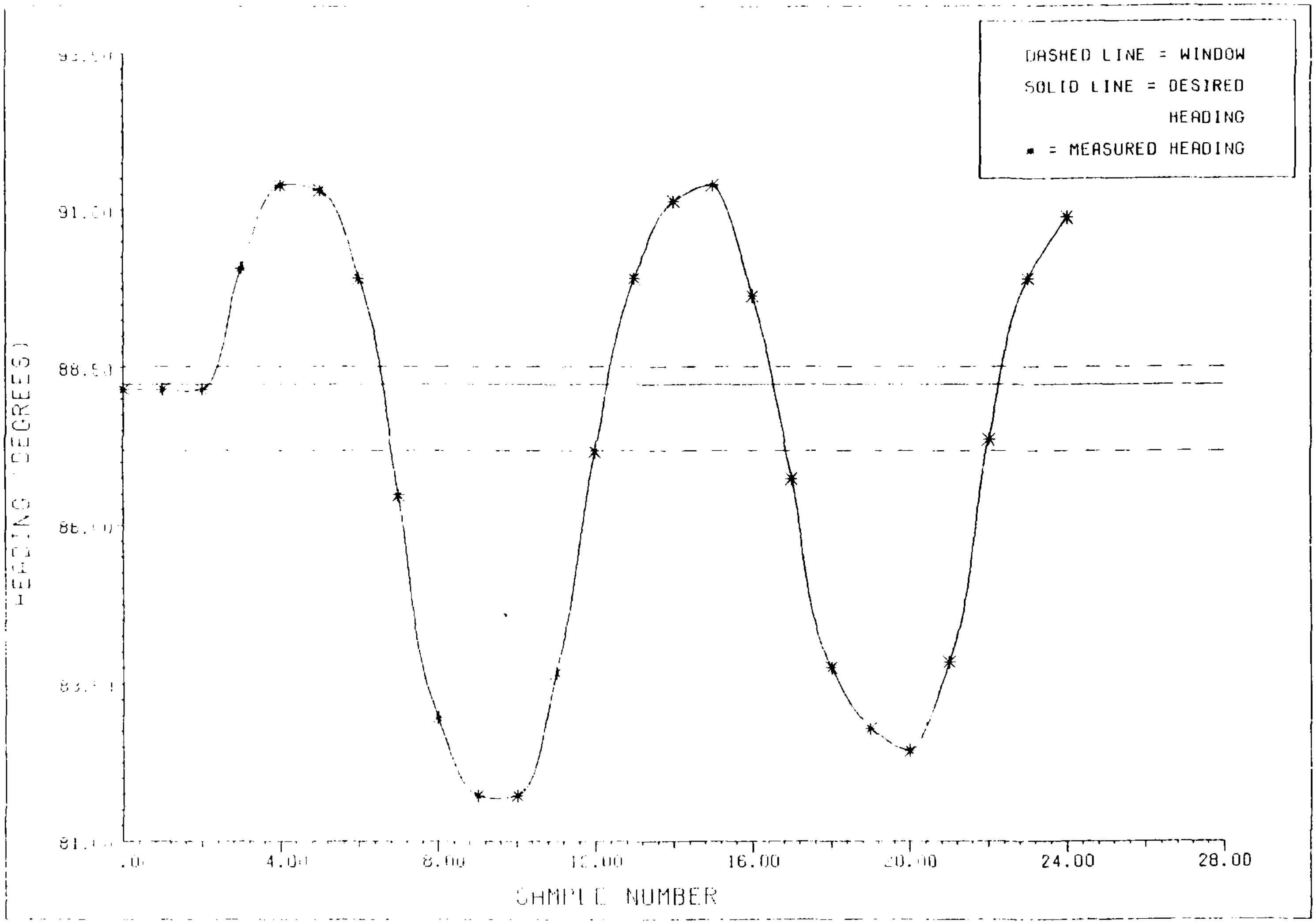


Figure 5.6. GYRAC Navigation Test #1, Automatic control of MARRS-1 Heading.

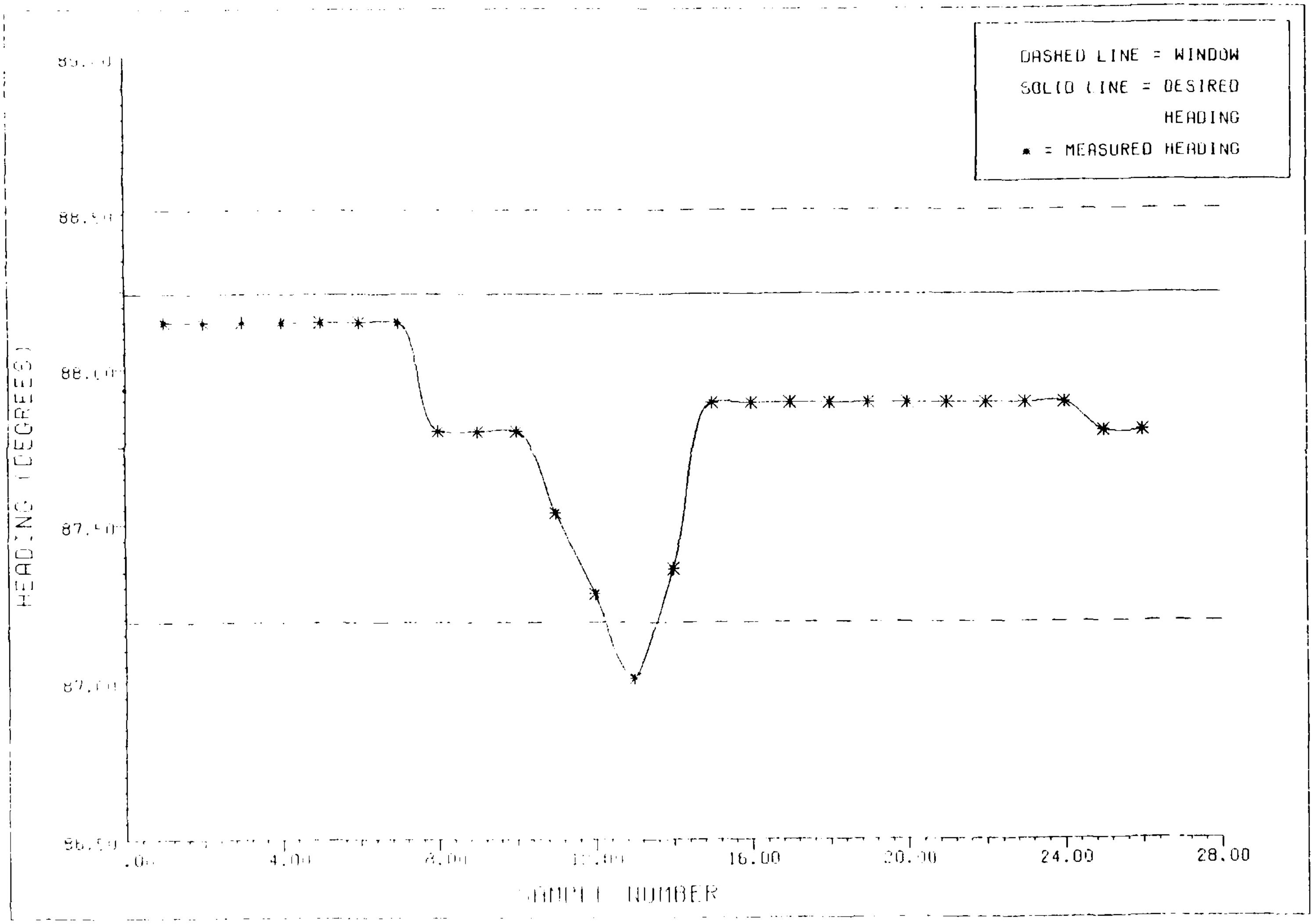


Figure 5.7. GYRAC Navigation Test #2, Automatic Control of MARRS-1 Heading.

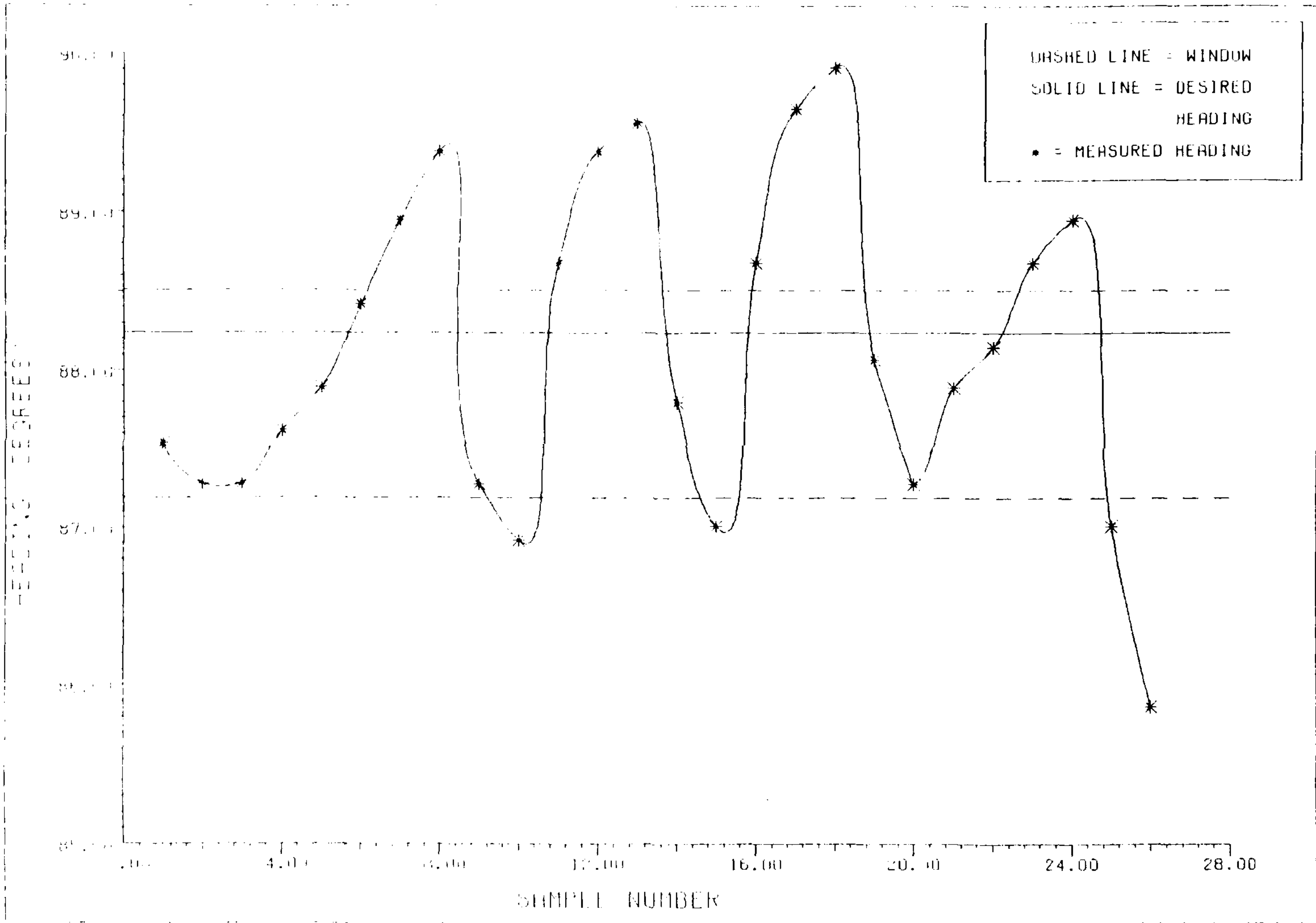


Figure 5.8. GYRAC Navigation Test #3, Automatic Control of MARRS-1 Heading.

course oscillation. However, the oscillations are not as severe as in Figure 5.6.

Careful study of all three figures indicate large oscillations occur when larger wheel angles (from the center position) are used. This causes frequent course changes to be executed because detected headings are not within the heading window. However, because course deviations oscillate around the actual course the mean course was very close to the desired course. The worst case oscillations resulted in movement of plus or minus five inches around the desired heading. No course drift was observed which is supported by a worst case final destination error of five inches. Therefore, feasibility of gyro based robot navigation has been demonstrated.

Review of Assumptions

The purpose of this section is to address the validity and impact of each assumption made in Chapter I and Chapter II.

Five governing assumptions were set forth in Chapter I. The first assumption, concerning local magnetic disturbances, proved to be valid. The magnetic flux detector was aligned only once and throughout the testing of the GYRAC, consistent heading information was obtained at all points in the test area.

The second assumption, that of a perfectly smooth and level operating surface, was the nemisis of this thesis. As mentioned under Phase II testing, the effect of accelerometer tilt error was far greater than anticipated. As a result, the GYRAC velocity data can not be used for navigation.

The validity of the remaining three assumptions (a perfect integrator, constant velocity over sample period, and precisely known sample time), all dealing with the accelerometer subsystem, could not be determined due to the inaccuracy in sensed acceleration. The effect of each of these assumptions is expected to be small given an accurate acceleration measure.

In Chapter II, it was assumed that the bias and scale factor errors would be negligible. The effect of this assumption could not be determined. However, due to the very small acceleration levels involved with the movement of MARRS-1, the bias and scale factor errors could have a significant impact on the accuracy of the sensed acceleration. In this case, they would have to be compensated for.

Miscelaneous

The gyro base assembly, which serves as the power source for the entire GYRAC system, was noticed to get extremely warm during operation of the GYRAC. To avoid damage to the base assembly, a separate source of +5 volt power was instituted. A separate external power supply is used to source the +5 volts and is provided through the same cable as is used for the system +28 volts external supply. This modification greatly reduced the load on the gyro base assembly and corrected the heating problem. The GYRAC power panel diagram in Appendix E has been updated to reflect this change.

VI. Summary, Conclusions, and Recommendations

Summary and Conclusions

The purpose of this thesis was to design and fabricate a real time, point to point, closed loop, mobile autonomous robot navigation system for the MARRS-1 robot. Specifically, the thesis was limited to three primary tasks. The first task was to develop the GYRAC system which would be capable of providing heading and velocity data. The second task was to integrate the GYRAC onto the MARRS-1 robot for verification testing, and the third task was to demonstrate autonomous navigation with the MARRS-1/GYRAC system.

Each of the these three parts was completed with the last task being completed to a limited extent. The GYRAC system is a complete and functional unit. However, the velocity data from the GYRAC is not usable for navigation. As mentioned in Chapter V, the true acceleration due to motion rarely, if ever, gets above the tilt error sensed by the accelerometer. This results in an ambiguous acceleration signal and thus an inaccurate velocity signal.

This problem has two major causes; the acceleration actually experienced by the MARRS-1 as it travels across the floor is very small in magnitude and short in duration; and, there is no error compensation in the accelerometer subsystem for gravity induced (tilt) errors. Nothing can be done about the small accelerations experienced by the

MARRS-1, since it is an inherently slow moving robot. Furthermore, speeding up the movement of MARRS-1 would not solve the problem since any flexible robot navigation system must be able to perform well at all reasonable speeds. This means that to make the GYRAC a completely usable system, a method of isolating the accelerometer from gravity tilt error must be incorporated. There are several methods for overcoming this tilt error problem. Several of these are presented in the Recommendations section.

The problems encountered with the accelerometer subsystem should not overshadow the success with the remainder of the GYRAC system. The GYRAC has proven to be a very reliable and accurate source of heading data. This heading data can be used by any robot system with an RS-232 serial interface. In addition, the heading data available from the GYRAC can be with respect to any reference direction and only one alignment along this reference is necessary. Subsequently, the GYRAC need only be powered up and will automatically provide accurate heading data with respect to the aligned reference. This GYRAC heading data could be combined with a separate source of distance measurement, such as wheel optical shaft encoders, to produce a viable navigation system. This could be accomplished on the MARRS-1 through software alone and is discussed further in the Recommendations section.

The GYRAC is completely integrated onto the MARRS-1 and several tests of the integrated system have been completed verifying the compatibility of the two systems. Numerous software routines were produced allowing for communication between the MARRS-1 and the GYRAC and for data gathering and processing purposes. Complete detail of these programs can be found in Appendices G, H, and I.

The MARRS-1 is not yet capable of autonomous navigation, but a step was made toward that goal. The MARRS-1 can follow a given heading under self control of the steering stepper motor. MARRS-1 can be initially positioned at any heading and under self control it will rotate in place until it is aligned along a programmed heading, straighten out the front wheel, and begin forward movement making steering corrections as it travels in order to maintain its heading. Currently, the MARRS-1 will follow the programmed heading until manually stopped. With the addition of a distance measurement to the control algorithms, the MARRS-1 could be programmed to move autonomously about the test area.

Finally, the importance of robot world modelling and path planning to autonomous navigation should not be taken lightly. Before a robot can begin to move it must have some knowledge of its surroundings and it must be able to plan out collision free and efficient pathways through its environment. Some of the current techniques of world modelling have been presented along with a new approach.

Path planning, under this new approach consists of finding an unobstructed pathway to the goal point. Safe points are used only if a direct path does not exist. The details of this path planning have been developed through a simple example.

Recommendations

There was not time to accomplish many of the goals optimistically set forth at the beginning of this thesis effort. In addition, throughout the development of the GYRAC system and while working with the MARRS-1 robot, many problems were identified too late to correct and many new ideas were conceived too late to implement. Therefore, the following recommendations are offered as possible extensions of, or improvements to, this thesis:

1. To correct the tilt error problem with the accelerometer subsystem in the GYRAC, some type of error compensation is necessary. For example, two or more accelerometers could be added to the system. These accelerometers would be perpendicular to each other and perpendicular to the existing accelerometer. By aligning one accelerometer along the vertical and the other along the horizontal (but perpendicular to the existing accelerometer), a signal could be generated which would be proportional to the amount of tilt experienced by the platform. This signal could be subtracted from the original

accelerometer signal; thereby, nulling out the tilt error. Perpendicular accelerometer pairs are commercially available through Sundstrand Data Control Incorporated and other manufacturers. In addition, due to the small amount of acceleration actually experienced by the MARRS-1, another single-axis accelerometer should be purchased with much greater sensitivity. This accelerometer would replace the current QA-1100 in the GYRAC. A full scale range of plus or minus one "g" would be sufficient (the QA-1100 has a range of plus or minus 13 "g's") and would result in accelerometer readings which would be less succeptible to bias and scale factor errors. Also, a new integrator circuit should be designed with a much higher impedence to avoid loading the accelerometer internal servo circuit. This is necessary so the sensitivity of the accelerometer will not be effected by the integrator circuit. The new integrator circuit must also be designed with drift rate in mind, as described in Chapter V under Phase I testing.

2. Another possibility for correcting the tilt error problem would be to incorporate a displacement gyro. This gyro could be used to sense small displacement angles of the accelerometer into the vertical. This displacement, or tilt angle, could be used to generate a signal proportional to the amount of accelerometer tilt. This signal could then be used to null out the tilt error.

3. The tilt error problem could also be corrected by

using a stable platform, such as those used in inertial navigation systems (INS), to mount the accelerometer. Only a single axis platform would be required to maintain the accelerometer input axis in the horizontal plane. Taking this suggestion even further, a complete INS could be incorporated on the MARRS-1 or another robot instead of the GYRAC. An INS would provide velocity and direction information sufficient for navigation.

4. The GYRAC heading data could be combined with an external source of displacement data, such as the wheel shaft encoder data on MARRS-1, to produce data suitable for position determination.

5. Tests need to be conducted to compare the accuracy of computing the position of MARRS-1 based on pure wheel shaft data with computed position based on both GYRAC heading and wheel shaft data.

6. More work is necessary to refine the control of the MARRS-1 allowing it to follow a given heading. Reasonably accurate navigation was observed using the relatively simple approached outlined in Chapter V. However, several improvements can be made that should greatly improve performance.

First, the drive computer control programs and communication software should be rewritten in assembly language using single byte ASCII commands. This will eliminate command and communication time delays allowing faster steering response (hundreds per second as opposed to one per second). This would also allow the heading window to be narrowed which would help reduce oscillations.

This change also requires the command routines of the Nav program to be changed, but nav.a (see Appendix H) has been designed in a three level structure to make changes and additions easy. The bottom level consists of hardware and device dependent routines such as transmit a byte of data to the Drive computer. The middle level of intermediate routines uses the lower level routines to define function primitives such as turn on drive motor at high speed. The top level of commands use the function primitives to form navigation commands such as rotate until locked on the heading window. Therefore, each level is independent of the way lower levels are implemented which limits the effects of changes to within a module.

Second, more intelligent steering control routines should be developed. They should anticipate when to start straightening out the front wheel instead of trying to do it all at once as was done in this thesis. In addition, they should be able to move the steering wheel in increments proportional to the amount of correction needed as opposed to single degree increments per correction. These additions will flatten out the oscillations and provide better navigation accuracy.

7. Once an accurate method of position determination

and of controlling the MARRS-1 is verified, the next step would be to develop the math routines necessary for MARRS-1 to perform the onboard processing required for navigation. The full world model as described in Chapter IV could then be implimented to provide the MARRS-1 with path planning capability.

Bibliography

1. 006-0111-05. KCS 55/55A Pictorial Navigation System. Installation Manual, King Radio Corporation, Olathe, Kansas, November 1981.

2. 006-5111-05. KI 525 Pictorial Navigation Indicator. Maintenance/Overhaul Manual. King Radio Corporation, Olathe, Kansas, March 1984.

3. 006-5111-05. KG 102 Directional Gyro. Maintenance/ Overhaul Manual. King Radio Corporation, Olathe, Kansas, March 1984.

4. 006-5111-05. KMT 112 Magnetic Aximuth Transmitter. Maintenance/Overhaul Manual. King Radio Corporation, Olathe, Kansas, March 1984.

5. 006-5111-05. KA 51Blaving Accessory. Maintenance/ Overhaul Manual. King Radio Corporation, Olathe, Kansas, March 1984.

6. 006-8250-04 20K. King KCS 55A Silver Crown Compass System. Specification Sheet. King Radio Corporation, Olathe, Kansas, April 1979.

7. 012-0293-001. Q-Flex Accelerometers. Instruction Manual. Sundstrand Data Control Incorporated, Redmond, Washington, January 1878.

8. Bloom, Capt Roland J., Dyer, Capt Donald D. and Capt Fred Williams. "Lab #3." Laboratory Report. Wright Patterson AFB, OH: EE 6.44, School of Engineering, Air Force Institute of Technology, March 1985.

9. Bortz, A.B. "Moravec's Mobile Robots," Robotics Age, 6: 25-31 (September 1984).

10. Clifford, T. and H. Schneider. Creating a Mobile Autonomous Robot Research System (MARRS). MS thesis. Wright-Patterson AFB, OH: School of Engineering, Air Force Institute of Technology, December 1984. (AFIT/GE/ENG/84D-19)

11. Coleman, Arthur. "Robotics Research: The Next Five Years and Beyond," Robotics Age, 7: 14-19 (Februrary 1985).

12. Crowley, James 1. "Navigation for an Intelligent Mobile Robot," IEEE Journal of Robotics and Automation, 1: 31-41 (March 1985).

13. Ebisch, Robert. "Metal Warriors," The Computer & Electronics Graduate, 2: 45-48 (Spring 1985).

14. General Officer Steering Committee for Artificial Intelligence and Robotics. Draft Summary of TRADOC Rquirements for Generic Robot Vehicle Systems. 9 April 1985.

15. Lozano-Perez, T. and M. A. Wesley. "An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles," Communications of the ACM, 22 (10): 560-570 (October 1979).

16. Monaghan, Capt Glen E. Navigation for an Autonomous Mobile Robot. MS thesis. Wright-Patterson AFB, OH: School of Engineering, Air Force Institute of Technology, December 1984. (AFIT/GE/ENG/84D-47)

17. Moravec, Hans P. "The Stanford Cart and the CMU Rover," Proceedings of the IEEE, 71: 872-884. IEEE Press, New York, 1983.

18. O'Donogue, Val. Syncro & Resolver Conversion. Memory Devices Ltd., Surrey, United Kingdom, 1980.

19. Owen, R.J. Environmental Mapping by a Hero-1 Robot Using Sonar and Laser Barcode Scanner. MS thesis. Wright-Patterson AFB, OH: School of Engineering, Air Force Institute of Technology, December 1983. (AFIT/GE/EE/83D-52)

20. Q-Flex Servo Accelerometers. Brochure. Sundstrand Data Control, Incorporated, Redmond Washington, 1978.

21. Tice, Jim. "Autonomous Vehicles Ready for Intellignt Applications," Air Force Times: 18 (29 July 1985).

Vita

Captain Roland J. Bloom was born on 2 December 1958 in Grants, New Mexico. After graduating from White Pine High School, Ely, Nevada in 1977 he entered the United States Air Force Academy in Colorado Springs, Colorado.

Upon graduation from the Academy in May 1981, he received the degree of Bachelor of Science in Astronautical Engineering and was commissioned as a Regular Second Lieutenant of the United States Air Force.

He was assigned to the Peacekeeper Division of the 6595th Missile Test Group at Vandenberg AFB, California. While at Vandenberg AFB, Captain Bloom served as primary Test Controller for the assembly, check-out, and launch preparation of the first four Peacekeeper flight-test missiles.

In may 1984, he entered the Masters's program in Astronautical Engineering at the Air Force Institute of Technology.

Captain Bloom is married to the former Raylene A. Burgess of East Ely, Nevada. They have two children: Jessica and Brandon.

Address: Box 316
Jackpot, Nevada 89825

Vita

Captain William J. Ramey Jr. was born December 25, 1952 in Great Falls, Montana. He graduated as class Valedictorian in 1971 from Jefferson High School, Boulder, Montana and later that year attended Montana State University. He enlisted in the Air Force in 1974 and became an Honor Graduate of the Cryptographic Equipment Maintenance School at Lackland AFB, Texas, where he was later assigned as an Instructor. He attended San Antonio College pursuing a degree in Electrical Engineering. In 1977, he was selected for the Airman Education and Commissioning Program and attended Texas A&M University where he received a Bachelor of Science degree in Electrical Engineering in 1979. In 1980, he graduated as a Distinguished Graduate from Officer Training School. Upon graduation, he was assigned to the National Security Agency (NSA) as a Computer Engineer and was later certified by NSA as a Senior Cryptologic Engineer. He attended the University of Maryland pursuing a Masters degree in Electrical Engineering. In 1984, Captain Ramey began a Masters program in Electrical Engineering at the Air Force Institute of Technology, Wright Patterson AFB, Ohio.

Captain Ramey is married to the former Elizabeth A. West of Farmingdale, New York. They have three children: Matthew, Joshua, and Katherine.

Address: Box 371
Boulder, Montana 59632

APPENDIX A

SDC 1700 S/D Converter Data Sheet

ANALOG DEVICES

Low Profile Synchro/Resolver-to-Digital Converter

SDC1700/1702/1704 SERIES

FEATURES

- Internal Microtransformers for 60Hz, 400Hz and 2.6kHz References
- Low Profile (0.4")
- 10-, 12- or 14-Bit Resolution for 360°
- High Tracking Rates (75 revs/sec)
- Voltage Scaling with External Resistors (Unique Feature)
- DC Voltage Output Proportional to Angular Velocity
- Low Cost
- Lightweight 3oz. (85 grams)
- MIL Spec/Hi Rel Options Available

APPLICATIONS

- Servo Mechanisms
- Retransmission Systems
- Coordinate Conversion
- Antenna Monitoring
- Simulation
- Industrial Controls
- Fire Control Systems
- Machine Tool Control Systems

GENERAL DESCRIPTION

The SDC1700, SDC1702 and SDC1704 are modular, continuous tracking Synchro/Resolver-to-Digital Converters which employ a type 2 servo loop.

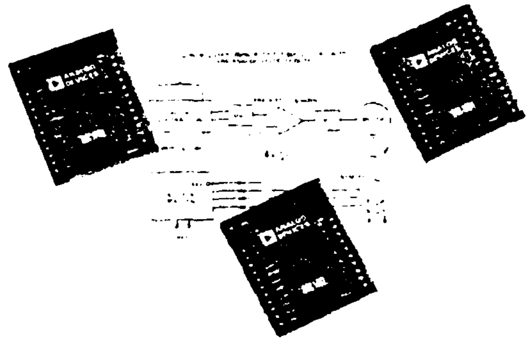
They are intended for use in both Industrial and Military applications.

The input signals can be either 3 wire synchro plus reference or 4 wire resolver plus reference, depending on the option. The outputs will be presented in TTL compatible, parallel natural binary.

One of the outstanding features of the converters is the use of precision Scott T and reference microtransformers. *This has made it possible to include the transformers within the module, even on the 60Hz option, and yet still maintain the profile height of 0.4"*

Particular attention has been paid in the design, to achieving the highest tracking rates and accelerations possible, compatible with the *resolution and carrier frequency* used, while at the same time obtaining a high overall accuracy.

When SDC's are used in control loops, it is often useful to have a voltage which is proportional to angular velocity. This voltage is available and has been brought out on all the SDC1700 converters.



Extended temperature range versions* of all the converters are available.

MODELS AVAILABLE

The three Synchro-to-Digital Converters described in this data sheet differ primarily in the areas of resolution, accuracy and dynamic performance as follows.

Model SDC1702XYZ is a 10-bit converter which has an overall accuracy of ±22 arc-minutes and a resolution of 21 arc-minutes.

Model SDC1700XYZ is a 12-bit converter with an overall accuracy of ±8.5 arc-minutes and a resolution of 5.3 arc-minutes.

Model SDC1704XYZ is a 14-bit converter with an overall accuracy of ±2.2 arc-minutes ±1LSB and a resolution of 1.3 arc-minutes.

The XYZ code defines the option thus: (X) signifies the operating temperature range, (Y) signifies the reference frequency, (Z) signifies the input voltage and range, and whether it will accept synchro or resolver format.

More information about the option code is given under the heading of "Ordering Information".

NOTE
*For all the standard options, no external transformers are needed with these converters.

SDC 1700 S/D Converter Data Sheet (continued)

SPECIFICATIONS (typical @ +25°C unless otherwise noted)

MODELS	SDC1702	SDC1700	SDC1704
ACCURACY[1] (max error)			
60Hz	±22 arc minutes	±8.5 arc minutes	±2.9 arc minutes ±1LSB
400Hz	±22 arc minutes	±8.5 arc minutes	±2.2 arc minutes ±1LSB
2.6kHz	±22 arc minutes	±8.5 arc minutes	±2.9 arc minutes ±1LSB
RESOLUTION	10 Bits (1LSB = 21 arc mins)	12 Bits (1LSB = 5.3 arc mins)	14 Bits (1LSB = 1.3 arc mins)
OUTPUT (In Parallel)	10 Bits (Natural Binary)	12 Bits (Natural Binary)	14 Bits (Natural Binary)
SIGNAL AND REFERENCE FREQUENCY	60Hz, 400Hz, 2.6kHz	•	•
SIGNAL VOLTAGE (Line-to-Line)			
Low Level	11.8V rms	•	•
High Level	90V rms	•	•
SIGNAL IMPEDANCES			
Low Level	26kΩ (Resistive)	•	•
High Level	200kΩ (Resistive)	•	•
REFERENCE VOLTAGE			
Low Level	26V (11.8V Signal)	•	•
High Level	115V (90V Signal)	•	•
REFERENCE IMPEDANCE	270kΩ (115V Signal) 56kΩ (26V Reference) (Impedance is Resistive)	• • •	• • •
TRANSFORMER ISOLATION	500V dc	•	•
TRACKING RATE (min)			
60Hz	5 Revolutions Per Second	•	500°/sec
400Hz	36 Revolutions Per Second	•	12 Revolutions Per Second
2.6kHz	75 Revolutions Per Second	•	25 Revolutions Per Second
Accel. Constant K_a			
60Hz	1880/sec²	•	520/sec²
400Hz	110,000/sec²	•	36,000/sec²
2.6kHz	518,000/sec²	•	170,000/sec²
STEP RESPONSE (179° Step) (For 1LSB Error)			
60Hz	1.5sec	•	•
400Hz	125ms	•	•
2.5kHz	50ms	•	•
POWER LINES	±15V @ 25mA, +5V @ 70mA } ±5%	• •	±15V @ 30mA, +5V @ 85mA } ±5%
POWER DISSIPATION	1.1 Watts	•	1.3 Watts
DATA LOGIC OUTPUT[2] (TTL Compatible)	2TTL Loads SDC17026YZ 4TTL Loads SDC17025YZ	2TTL Loads SDC17006YZ 4TTL Loads SDC17005YZ	2TTL Loads on All Options
BUSY LOGIC OUTPUT, POSITIVE PULSE (1 TTL Load)			
60Hz	9.0µs } ±30%	•	9.0µs } ±30%
400Hz	2.0µs	•	2.0µs
2.6kHz	2.0µs	•	1.5µs
MAX DATA TRANSFER TIME			
60Hz	40µs	•	35µs
400Hz	5.0µs	•	3.0µs
2.6kHz	1.8µs	•	0.8µs
INHIBIT INPUT (To Inhibit)	Logic "0" 1 TTL Load	•	Logic "0" 2 TTL Loads
WARM UP TIME	1 sec to Rated Accuracy	•	•
TEMPERATURE RANGE			
Operating	0 to +70°C Standard -55°C to +105°C Extended	• •	• •
Storage	-55°C to +125°C	•	•
DIMENSIONS	3.125" x 2.625" x 0.4" (79.4 x 66.7 x 10.2mm)	•	•
WEIGHT	[illegible] oz (65 grams)	•	•

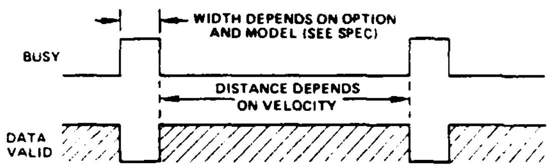
NOTES
*Specifications same as SDC1702
[1] Specified over the appropriate operating temperature range of the option and for (a) ±10% signal and reference amplitude variation, (b) 10% signal and reference harmonic distortion, (c) ±5% power supply variation, (d) ±10% variation in reference frequency.
[2] It is recommended that buffers should be used if the above converters are required to drive over a distance greater than 6"
Specifications subject to change without notice.

DATA TRANSFER (All Models)

The readiness of the converters for data transfer is indicated by the state of the BUSY pin.

The voltage appearing on the BUSY pin consists of a train of pulses, at TTL levels, of length according to the model and option (see specification table). The converter is busy when the BUSY pin is at a TTL "High" level. These pulses correspond to those delivered by the VCO to increment or decrement the up-down counter (see schematic diagram). Thus the pulses will occur for increasing and decreasing counts.

The most suitable time for transferring data is when the BUSY is at a logic "Lo" state, and the times allowable for data transfer are shown in the specification. Even at the maximum speed of the option, these times will be sufficient to transfer data before the next BUSY pulse occurs.



Data Transfer Diagram

DATA TRANSFER DIAGRAM

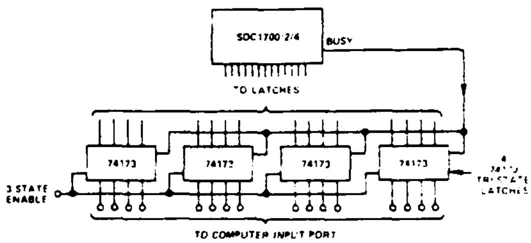
Taking the INHIBIT to a logic "Lo" state prevents the VCO (BUSY) pulses from updating the up-down counter. However, if applied during a BUSY pulse, the INHIBIT will not become effective until the end of the BUSY pulse.

The best method of transferring the data is by applying the INHIBIT (taking it to a logic "Lo" state), waiting for at least the width of a BUSY pulse, transferring the data and releasing the INHIBIT.

Note that sustained application of the INHIBIT opens the internal control loop and the converter may take on appreciable time to recover to full accuracy when the loop is restored.

INTERFACING WITH A COMPUTER

It is recommended that external latches are used to enable data to be transferred onto a computer data bus. One method is shown in the diagram. Using this method will mean that the latches are constantly updated by the BUSY signal, while at the same time enabling inputs to be made to the computer by means of normal data transfer procedures. The AC1755 mounting card contains these external components.



Suggested External Computer Interface Circuitry

THEORY OF OPERATION

If the unit is a Synchro-to-Digital Converter, then the 3 wire synchro output will be connected to S1, S2 and S3 on the module and the Scott T transformer pair will convert these signals into resolver format.

i.e., $V_1 = K\,E_O \sin \omega t \sin \theta$
$V_2 = K\,E_O \sin \omega t \cos \theta$

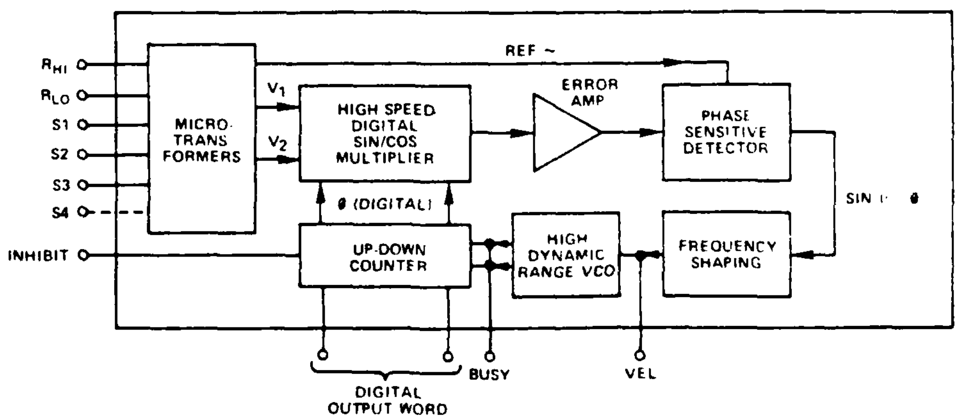
Where θ is the angle of the Synchro Shaft

If the unit is a Resolver-to-Digital Converter, then the 4 wire resolver output will be connected to S1, S2, S3 and S4 on the module and the microtransformer will act purely as an isolator.

To understand the conversion process, then assume that the current word state of the up-down counter is ϕ

The V_1 is multiplied by Cos ϕ and V_2 is multiplied by Sin ϕ to give

$K\,E_O \sin \omega t \sin \theta \cos \phi$
and $K\,E_O \sin \omega t \cos \theta \sin \phi$



Functional Diagram of the SDC1700/2/4 Converters

SDC 1700 S/D Converter Data Sheet (continued)

These signals are subtracted by the error amplifier to give:

$$K E_0 \text{ Sin } \omega t \,(\text{Sin } \theta \text{ Cos } \phi - \text{Cos } \theta \text{ Sin } \phi)$$

or $$K E_0 \text{ Sin } \omega t \text{ Sin } (\theta - \phi)$$

A phase sensitive detector, integrator and Voltage Controlled Oscillator (VCO) form a closed loop system which seeks to null Sin $(\theta - \phi)$

When this is accomplished, the word state of the up-down counter (ϕ), equals within the rated accuracy of the converter, the synchro shaft angle θ.

CONNECTING THE CONVERTER

The electrical connections to the converter are straightforward. The power lines, which must not be reversed, are ±15V and 5V. They must be connected to the "±15V" and "5V" pins with the common connection to the ground pin GND.

It is suggested that 0.1μF and 6.8μF capacitors be placed in parallel from +15V to GND, from −15V to GND and from +5V to GND.

The digital output is taken from pins:

- 1 through to 10 for the SDC1702
- 1 through to 12 for the SDC1700
- 1 through to 14 for the SDC1704

Pin 1 represents the MSB in each case. The reference connections are made to pins "R_{HI}" and "R_{LO}".

In the case of a Synchro, the signals are connected to "S1", "S2" and "S3" according to the following convention:

$$E_{S1-S3} = E_{RLO-RHI} \text{ Sin } \omega t \text{ Sin } \theta$$
$$E_{S3-S2} = E_{RLO-RHI} \text{ Sin } \omega t \text{ Sin } (\theta + 120^\circ)$$
$$E_{S2-S1} = E_{RLO-RHI} \text{ Sin } \omega t \text{ Sin } (\theta + 240^\circ)$$

For a resolver, the signals are connected to "S1", "S2", "S3" and "S4" according to the following convention:

$$E_{S1-S3} = E_{RLO-RHI} \text{ Sin } \omega t \text{ Sin } \theta$$
$$E_{S2-S4} = E_{RHI-RLO} \text{ Sin } \omega t \text{ Cos } \theta$$

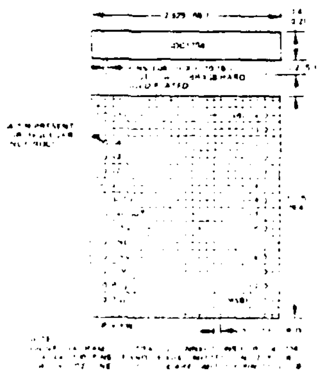
The analog voltage representing velocity is available between "VEL" and "GND".

The "BUSY" and "INHIBIT" pin (if used), should be connected as described under the heading "Data Transfer".

NOTE. If the INHIBIT pin is used (i.e., driven to 0 volts), the control loop will be opened and a finite time will be required (see spec) for the converter to recover.

OUTLINE DIMENSIONS AND PIN CONNECTION DIAGRAM

Dimensions are shown in inches and (mm).



MATING SOCKET: CAMBION 450-3388-01-03

RESISTIVE SCALING OF INPUTS

A unique feature of the SDC1700 series of converters is that the inputs can be resistively scaled to accommodate any range of input signal and reference voltages.

This means that a standard converter can be used with a personality card in systems where a wide range of input and reference voltages are encountered. In addition it should be noted that a 400Hz unit will operate from a 2.6kHz reference. It will however have the velocity and acceleration characteristics as specified for the 400Hz converter. A 60Hz converter will operate from a 400Hz reference and will have the velocity and acceleration characteristics as specified for the 60Hz converter.

To calculate the values of the external scaling resistors for a synchro converter, add 1.11kΩ in series with S1, S2 and S3 per extra volt in the case of the signal, and 2.2kΩ in the case of the reference. In the case of a resolver converter add 2.22kΩ per extra volt in series with S1 and S2 for the signal and 2.2kΩ per extra volt in series with R_{HI} for the reference.

For example, assume that we have an 11.8 volt line to line signal/26.0 volt reference converter, and we wish to use a 60 volt line to line signal with a 115 volt reference.

Thus in each signal input line, the extra voltage capability required is:

$$60 - 11.8 = 48.2 \text{ volts}$$

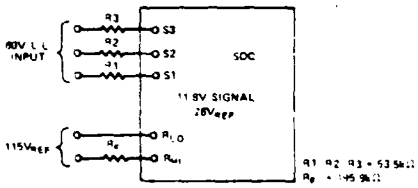
Therefore each resistor needs to have a value of 48.2 x 1.11 = 53.5kΩ. In the case of the reference, the extra voltage capability required is:

$$115 - 26.0 = 89 \text{ volts}$$

Therefore the resistor needs to have a value of:

$$89.0 \times 2.2 = 195.8\text{k}\Omega$$

Thus the inputs can be scaled as in the diagram below.



NOTE
IN THE CASE OF R1, R2 AND R3 THE RATIO ERRORS BETWEEN THE RESISTANCES IS MORE IMPORTANT THAN THE ABSOLUTE RESISTANCE VALUES

IN GENERAL A 1% RATIO ERROR WILL GIVE RISE TO AN EXTRA INACCURACY OF 17 ARC MINUTES WHILE A RATIO ERROR OF 0.1% WILL GIVE RISE TO AN EXTRA INACCURACY OF 1.7 ARC MINUTES

THE ABSOLUTE VALUE OF R_e IS NOT CRITICAL

BIT WEIGHT TABLE

Bit Number	Weight in Degrees
1 (MSB)	180.0000
2	90.0000
3	45.0000
4	22.5000
5	11.2500
6	5.6250
7	2.8125
8	1.4063
9	0.7031
10 (LSB for SDC1702)	0.3516
11	0.1758
12 (LSB for SDC1700)	0.0879
13	0.0439
14 (LSB for SDC1704)	0.0220

VELOCITY PIN

This pin provides a voltage output which is proportional to the angular velocity of the input. The voltage goes negative for an increasing digital angle and goes positive for a decreasing digital angle.

The characteristics of the velocity pin output are given in the table below.

Scaling of Output Voltage for One Fifth max. Velocity	2Volts (Nominal)
Output Voltage Temp. Coeff.	0.05% °C of Output
Output Voltage Drift (All Models)	0 to +70°C ±50μV/°C -55°C to +105°C ±100μV/°C
Linearity	0°sec to 800°/sec SDC1704 400Hz 1% 0°sec to 100°/sec SDC1704 60Hz 1% 0°sec to 800°/sec SDC1700/2 400Hz 2% 0°sec to 100°/sec SDC1700/2 60Hz 1.5%
Noise (0 to 20Hz)	@1600°/sec SDC1700/2/4 400Hz 2mV rms @200°/sec SDC1700/2/4 60Hz 2mV rms
Impedance (Output)	1Ω
max Current Available	1mA

The velocity voltage can be used in closed loop servo systems for stabilization instead of a tachometer.

The SDC1700/2/4 velocity outputs do not have the disadvantages of being inefficient at low speeds and do not need gearing required by tachometers. In addition, the output is available at no extra cost.

For other velocity output scaling and linearity consult the factory.

Two examples of the use of the velocity pin are shown in the diagram below.

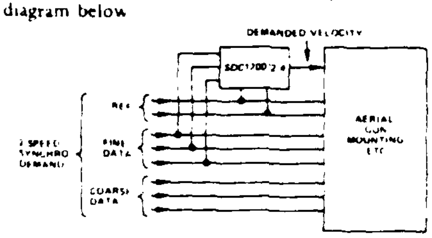


Diagram showing a velocity feed forward application. The SDC is used to produce the demanded velocity from Synchro form inputs.

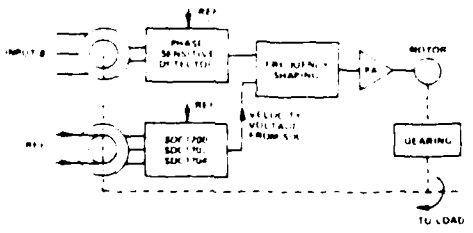
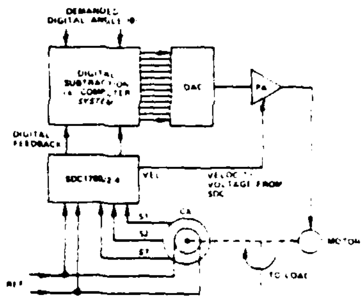


Diagram showing the velocity voltage being used to stabilize an electro-mechanical control loop.

APPLICATIONS OF SYNCHRO-TO-DIGITAL CONVERTERS

SDCs can be used in a variety of ways in control loops as well as for the conversion of angular data into a form which is readily acceptable to digital displays or computers.

The diagram below shows an SDC being used in a digitally controlled feedback loop.



An SDC Being Used in a Digitally Controlled Feedback Loop

Such loops as shown in the diagram above require the high dynamic performance of the SDC1700 series converters. It should be noted that in this application, the SDC1700 series will replace conventional tachometers and phase sensitive detectors while at the same time provide digital position feedback.

Many synchro systems employ a two speed, geared arrangement utilizing one synchro for the fine shaft and one for the coarse. An example of this type is shown below.

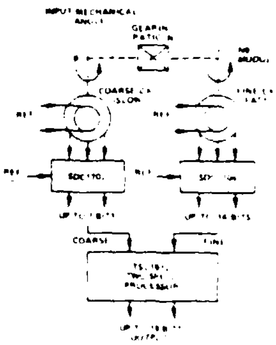


Diagram Showing Coarse/Fine Synchro Processor System

In the above example, two tracking SDC's are being used to provide data for coarse/fine (two speed) data transmission systems.

The TSL1612 is a processor which combines the outputs of two SDC's to provide one output word of up to 19 bits in length.

The TSL1612 is available for any ratio between 2:1 and 36:1 and provides automatic compensation for misalignment of the coarse synchro relative to its shaft. It also corrects for any overlap between the digits of the coarse and fine shafts.

MEAN TIME BETWEEN FAILURES (M.T.B.F.)

The estimated mean time between failures is given as follows:

SDC1700/2	174,000 Hours
SDC1704	167,000 Hours

Further information relating to M.T.B.F. and to the quality control and test procedures employed by us can be obtained from the factory on request.

TRANSFER FUNCTION

The transfer function of the SDC1700/2 and SDC1704, 400Hz versions, is given below.

For the transfer functions of the other models or for a detailed analysis of those given here, please contact us.

SDC1700/2 400Hz

$$\frac{\theta_0}{\theta_1} = \frac{8.8 \times 10^7 (1 + 6.8 \times 10^{-3} s)}{s^3 + 8.04 \times 10^2 s^2 + 6.1 \times 10^5 s + 8.8 \times 10^7}$$

SDC1704 400Hz

$$\frac{\theta_0}{\theta_1} = \frac{2.95 \times 10^7 (1 + 8.2 \times 10^{-3} s)}{s^3 + 8.05 \times 10^2 s^2 + 1.95 \times 10^5 s + 2.95 \times 10^7}$$

CARD MOUNTING

All the converters can be mounted on an AC1755 mounting card. This card contains the latches described under the "Data Transfer" heading, which are necessary to transfer the data on to a computer bus system, and sockets for the converter.

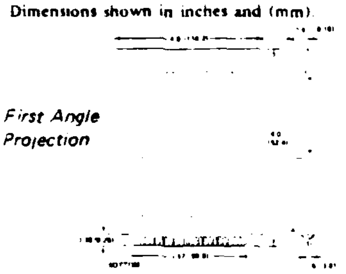
The latches have a tri-state output to facilitate ease of use.

The AC1755 also contains facilities for the inclusion of input signal and reference scaling resistors as described under the heading "Resistive Scaling of Inputs".

The card uses a 22/22 0.156" pitch edge connector. The pin out is shown below. If it is not required to use the external latches, they can be jumpered on the board.

AC1755 MOUNTING CARD

Dimensions shown in inches and (mm)



EDGE CONNECTIONS AC1755

Edge Pin Number	Function	Edge Pin Letter	Function
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	S4	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	INHIBIT	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]

[illegible]

ORDERING INFORMATION

Parts should be ordered by the appropriate part number (i.e., SDC1700, SDC1702, SDC1704) followed by the appropriate XYZ option code.

If the unit is to be a Resolver-to-Digital Converter, the SDC should be replaced by RDC in the part number.

The XYZ options are as follows:

X signifies the operating temperature range and the options are:

- X = 5 signifies 0 to +70°C (commercial) temperature.
- X = 6 signifies -55°C to +105°C (extended) temperature.

Y signifies the reference frequency and the options are:

- Y = 1 signifies 400Hz
- Y = 2 signifies 60Hz *
- Y = 4 signifies 2.6kHz

Z signifies the input signal and reference voltages and whether the converter is an SDC or an RDC. The options are:

- Z = 1 signifies synchro, signal 11.8V rms, reference 26V rms
- Z = 2 signifies synchro, signal 90V rms, reference 115V rms
- Z = 3 signifies resolver, signal 11.8V rms, reference 11.8V rms
- Z = 4 signifies resolver, signal 26V rms, reference 26V rms
- Z = 8 signifies resolver, signal 11.8V rms, reference 26V rms

Thus, for example, an SDC1704 with a commercial (0 to +70°C) operating range, using a 400Hz, 26V reference with an 11.8V signal would be ordered as an SDC1704511.

For other than these options, consult the factory.

CAUTIONS

Do not reverse the power supplies.

Do not connect signal and/or reference inputs to other than S1, S2, S3, S4, R_{HI} or R_{LO}.

Do not connect signals and/or references to a lower voltage rated converter. (Such as a 115V Synchro into a 26V Converter).

Misconnections as per the above will damage the units and void the warranty.

OTHER PRODUCTS

The SDC1700/2/4 converters are just a few of the modules and instruments concerned with Synchro and Resolver conversion manufactured by us.

Other products are listed below and technical data is available. If you have any questions about our products or require advice about the use of them for a particular application, please contact our Applications Engineering Department.

TWO SPEED PROCESSORS

Which utilize the digital outputs of two SDCs in a 2 speed coarse/fine system to produce one combined digital word of up to 19 bits in length. The TSL1612 in particular is available for any ratio between 2:1 and 36:1.

DIGITAL-TO-SYNCHRO CONVERTERS

Resolutions of between 10 and 14 bits are available.

BCD OUTPUT SYNCHRO-TO-DIGITAL CONVERTERS

The SBCD1752 and SBCD1753 are converters with a BCD instead of a binary output based upon the SDC1700. They have outputs of ±180.0 degrees and 0 to 360.0 degrees respectively.

*60Hz Operation

For 50Hz operation, a 60Hz converter can be used with no reduction in accuracy.

AD573 A/D Converter Data Sheet

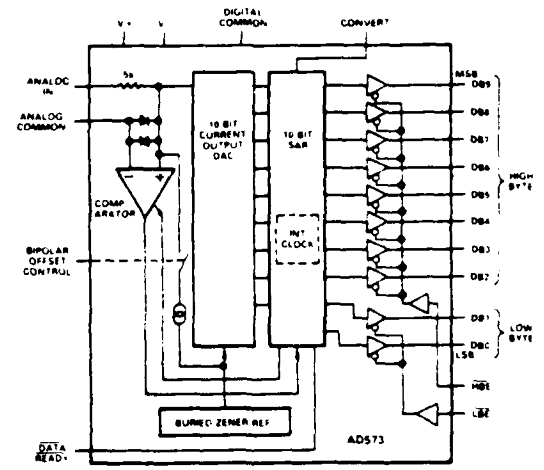
ANALOG DEVICES

Fast, Complete 10-Bit A/D Converter with Microprocessor Interface

AD573*

FEATURES
Complete 10-Bit A/D Converter with Reference, Clock and Comparator
Full 8- or 16-Bit Microprocessor Bus Interface
Fast Successive Approximation Conversion – 20µs typ
No Missing Codes Over Temperature
Operates on +5V and –12V to –15V Supplies
Low Cost Monolithic Construction

AD573 FUNCTIONAL BLOCK DIAGRAM



PRODUCT DESCRIPTION
The AD573 is a complete 10-bit successive approximation analog to digital converter consisting of a DAC, voltage reference, clock, comparator, successive approximation register (SAR) and 3 state output buffers—all fabricated on a single chip. No external components are required to perform a full accuracy 10-bit conversion in 20µs.

The AD573 incorporates the most advanced integrated circuit design and processing technology available today. The successive approximation function is implemented with I²L (integrated injection logic). Laser trimming of the high stability SiCr thin film resistor ladder network at the wafer stage (LWT) insures high accuracy, which is maintained with a temperature compensated sub-surface Zener reference.

Operating on supplies of +5V and –12V to –15V, the AD573 will accept analog inputs of 0 to +10V or –5V to +5V. The trailing edge of a positive pulse on the CONVERT line initiates the 20µs conversion cycle. $\overline{\text{DATA READY}}$ indicates completion of the conversion. $\overline{\text{HIGH BYTE ENABLE}}$ ($\overline{\text{HBE}}$) and $\overline{\text{LOW BYTE ENABLE}}$ ($\overline{\text{LBE}}$) control the 8-bit and 2-bit three state output buffers.

The AD573 is available in two versions for the 0 to +70°C temperature range, the AD573J and AD573K. The AD573S guarantees ±1LSB relative accuracy and no missing codes from –55°C to +125°C.

Two package configurations are offered. All versions are also offered in a 20-pin hermetically sealed ceramic DIP. The AD573J and AD573K are also available in a 20-pin plastic DIP.

PRODUCT HIGHLIGHTS
1. The AD573 is a complete 10-bit A/D converter. No external components are required to perform a conversion.
2. The AD573 interfaces to many popular microprocessors without external buffers or peripheral interface adapters. The 10 bits of output data can be read as a 10-bit word or as 8- and 2-bit words.
3. The device offers true 10-bit accuracy and exhibits no missing codes over its entire operating temperature range.
4. The AD573 adapts to either unipolar (0 to +10V) or bipolar (–5V to +5V) analog inputs by simply grounding or opening a single pin.
5. Performance is guaranteed with +5V and –12V or –15V supplies.

*Protected by U.S. Patent Nos. 3,940,760; 4,213,806; 4,136,349; 4,400,689; and 4,400,690.

AD573 A/D Converter Data Sheet (continued)

SPECIFICATIONS

($T_A = 25°C$, V+ = +5V, V− = −12V or −15V, all voltages measured with respect to digital common, unless otherwise indicated)

Model	AD573J Min	AD573J Typ	AD573J Max	AD573K Min	AD573K Typ	AD573K Max	AD573S Min	AD573S Typ	AD573S Max	Units
RESOLUTION		10			10			10		Bits
RELATIVE ACCURACY[1]			±1			±1/2			±1	LSB
$T_A = T_{min}$ to T_{max}			±1			±1/2			±1	LSB
FULL SCALE CALIBRATION[2]		±2			±2				±2	LSB
UNIPOLAR OFFSET			±1			±1/2			±1	LSB
BIPOLAR OFFSET			±1			±1/2			±1	LSB
DIFFERENTIAL NONLINEARITY[3]	10			10			10			Bits
$T_A = T_{min}$ to T_{max}		9			10				10	Bits
TEMPERATURE RANGE	0		+70	0		+70	−55		+125	°C
TEMPERATURE COEFFICIENTS[4]										
Unipolar Offset			±2			±1			±2	LSB
Bipolar Offset			±2			±1			±2	LSB
Full Scale Calibration[2]			±4			±2			±5	LSB
POWER SUPPLY REJECTION										
Positive Supply										
+4.5V ≤ V+ ≤ +5.5V			±2			±1			±2	LSB
Negative Supply										
−15.75V ≤ V− ≤ −14.25V			±2			±1			±2	LSB
−12.6V ≤ V− ≤ −11.4V			±2			±1			±2	LSB
ANALOG INPUT IMPEDANCE	3.0	5.0	7.0	3.0	5.0	7.0	3.0	5.0	7.0	kΩ
ANALOG INPUT RANGES										
Unipolar	0		+10	0		+10	0		+10	V
Bipolar	−5		+5	−5		+5	−5		+5	V
OUTPUT CODING										
Unipolar	Positive True Binary			Positive True Binary			Positive True Binary			
Bipolar	Positive True Offset Binary			Positive True Offset Binary			Positive True Offset Binary			
LOGIC OUTPUT										
Output Sink Current										
V_{OUT} = 0.4V max, T_{min} to T_{max}	3.2			3.2			3.2			mA
Output Source Current[5]										
V_{OUT} = 2.4V max, T_{min} to T_{max}	0.5			0.5			0.5			mA
Output Leakage			±40			±40			±40	μA
LOGIC INPUTS										
Input Current			±100			±100			±100	μA
Logic "1"	2.0			2.0			2.0			V
Logic "0"			0.8			0.8			0.8	V
CONVERSION TIME										
$T_A = T_{min}$ to T_{max}	10	20	30	10	20	30	10	20	30	μs
POWER SUPPLY										
V+	+4.5	+5.0	+7.0	+4.5	+5.0	+7.0	+4.5	+5.0	+7.0	V
V−	−11.4	−15	−16.5	−11.4	−15	−16.5	−11.4	−15	−16.5	V
OPERATING CURRENT										
V+		15	25		15	25		15	25	mA
V−		9	15		9	15		9	15	mA
PACKAGE[6]										
Ceramic DIP		D20A			D20A			D20A		
Plastic DIP		N20A			N20A					

NOTES

[1] Relative accuracy is defined as the deviation of the code transition points from the ideal transfer point on a straight line from the zero to the full scale of the device.

[2] Full scale calibration is guaranteed trimmable to zero with an external 50Ω potentiometer in place of the 15Ω fixed resistor.

Full scale is defined as 10 volts minus 1LSB, or 9.990 volts.

[3] Defined as the resolution for which no missing codes will occur.

[4] Change from +25°C value from +25°C to T_{min} or T_{max}.

[5] The data output lines have active pull-ups to source 0.5mA. The $\overline{\text{DATA READY}}$ line is open collector with a nominal 6kΩ internal pull-up resistor.

[6] See Section 19 for package outline information.

Specifications subject to change without notice.

Specifications shown in boldface are tested on all production units at final electrical test. Results from those tests are used to calculate outgoing quality levels. All min and max specifications are guaranteed, although only those shown in boldface are tested on all production units.

AD573 A/D Converter Data Sheet (continued)

FUNCTIONAL DESCRIPTION

A block diagram of the AD573 is shown in Figure 1. The positive CONVERT pulse must be at least 500ns wide. $\overline{DR}$ goes high within 1.5μs after the leading edge of the convert pulse indicating that the internal logic has been reset. The negative edge of the CONVERT pulse initiates the conversion. The internal 10-bit current output DAC is sequenced by the integrated injection logic (I^2L) successive approximation register (SAR) from its most significant bit to least significant bit to provide an output current which accurately balances the input signal current through the 5kΩ resistor. The comparator determines whether the addition of each successively weighted bit current causes the DAC current sum to be greater or less than the input current; if the sum is more, the bit is turned off. After testing all bits, the SAR contains a 10-bit binary code which accurately represents the input signal to within ±1/2LSB (0.05% of full scale).

The SAR drives $\overline{DR}$ low to indicate that the conversion is complete and that the data is available to the output buffers. $\overline{HBE}$ and $\overline{LBE}$ can then be activated to enable the upper 8-bit and lower 2 bit buffers as desired. $\overline{HBE}$ and $\overline{LBE}$ should be brought high prior to the next conversion to place the output buffers in the high impedance state.

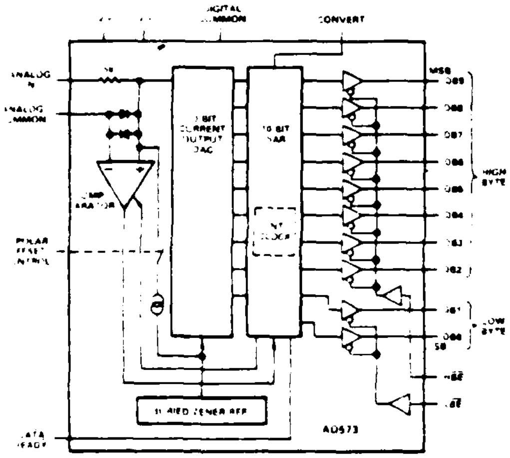


Figure 1. AD573 Functional Block Diagram

The temperature compensated buried Zener reference provides the primary voltage reference to the DAC and ensures excellent stability with both time and temperature. The bipolar offset input controls a switch which allows the positive bipolar offset current (exactly equal to the value of the MSB less 1/2LSB) to be injected into the summing (+) node of the comparator to offset the DAC output. Thus the nominal 0 to +10V unipolar input range becomes a −5V to +5V range. The 5kΩ thin film input resistor is trimmed so that with a full scale input signal, an input current will be generated which exactly matches the DAC output with all bits on.

UNIPOLAR CONNECTION

The AD573 contains all the active components required to perform a complete A/D conversion. Thus, for many applications, all that is necessary is connection of the power supplies (+5V and −12V to −15V), the analog input and the convert pulse. However, there are some features and special connections which should be considered for achieving optimum performance. The functional pin-out is shown in Figure 2.

The standard unipolar 0 to +10V range is obtained by shorting the bipolar offset control pin (pin 16) to digital common (pin 17).

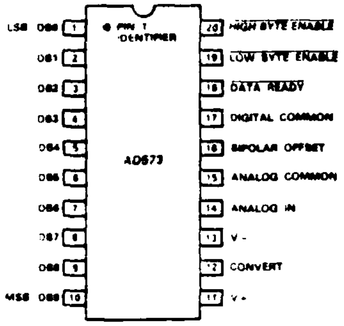


Figure 2. AD573 Pin Connections

Full Scale Calibration

The 5kΩ thin film input resistor is laser trimmed to produce a current which matches the full scale current of the internal DAC—plus about 0.3%—when an analog input voltage of 9.990 volts (10 volts − 1LSB) is applied at the input. The input resistor is trimmed in this way so that if a fine trimming potentiometer is inserted in series with the input signal, the input current at the full scale input voltage can be trimmed down to match the DAC full scale current as precisely as desired. However, for many applications the nominal 9.99 volt full scale can be achieved to sufficient accuracy by simply inserting a 15Ω resistor in series with the analog input to pin 14. Typical full scale calibration error will then be within ±2LSB or ±0.2%. If more precise calibration is desired, a 50Ω trimmer should be used instead. Set the analog input at 9.990 volts, and set the trimmer so that the output code is just at the transition between 1111111110 and 1111111111. Each LSB will then have a weight of 9.766mV. If a nominal full scale of 10.24 volts is desired (which makes the LSB have a weight of exactly 10.00mV), a 100Ω resistor and a 100Ω trimmer (or a 200Ω trimmer with good resolution) should be used. Of course, larger full scale ranges can be arranged by using a larger input resistor, but linearity and full scale temperature coefficient may be compromised if the external resistor becomes a sizeable percentage of 5kΩ. Figure 3 illustrates the connections required for full scale calibration.

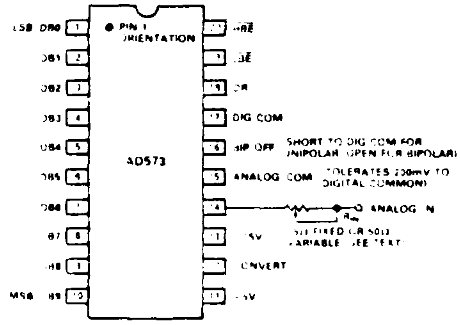


Figure 3. Standard AD573 Connections

Unipolar Offset Calibration

Since the Unipolar Offset is less than ±1LSB for all versions of the AD573, most applications will not require trimming. Figure 4 illustrates two trimming methods which can be used if greater accuracy is necessary.

Applying the AD573

Figure 4a shows how the converter zero may be offset by up to ±3 bits to correct the device initial offset and/or input signal offsets. As shown, the circuit gives approximately symmetrical adjustment in unipolar mode.

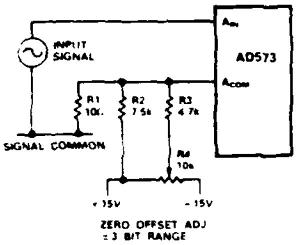


Figure 4a.

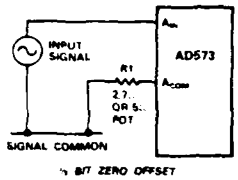


Figure 4b.

Figure 5 shows the nominal transfer curve near zero for an AD573 in unipolar mode. The code transitions are at the edges of the nominal bit weights. In some applications it will be preferable to offset the code transitions so that they fall between the nominal bit weights, as shown in the offset characteristics.

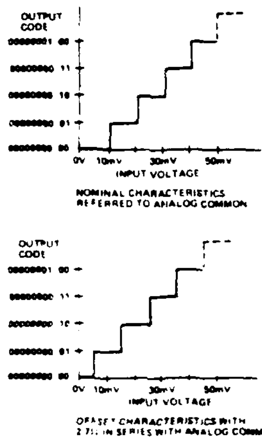


Figure 5. AD573 Transfer Curve – Unipolar Operation (Approximate Bit Weights Shown for Illustration, Nominal Bit Weights ≈ 9.766mV)

This offset can easily be accomplished as shown in Figure 4b. At balance (after a conversion) approximately 2mA flows into the Analog Common terminal. A 2.7Ω resistor in series with this terminal will result in approximately the desired ½ bit offset of the transfer characteristics. The nominal 2mA Analog Common current is not closely controlled in manufacture. If high accuracy is required, a 5Ω potentiometer (connected as a rheostat) can be used as R1. Additional negative offset range may be obtained by using larger values of R1. Of course, if the zero transition point is changed, the full scale transition point will also move. Thus, if an offset of ½LSB is introduced, full scale trimming as described on the previous page should be done with an analog input of 9.985 volts.

NOTE: During a conversion, transient currents from the Analog Common terminal will disturb the offset voltage. Capacitive decoupling should not be used around the offset network. These transients will settle appropriately during a conversion. Capacitive decoupling will "pump up" and fail to settle resulting in conversion errors. Power supply decoupling, which returns to analog signal common, should go to the signal input side of the resistive offset network.

BIPOLAR CONNECTION

To obtain the bipolar −5V to +5V range with an offset binary output code, the bipolar offset control pin is left open.

A −5.000 volt signal will give a 10-bit code of 00000000 00; an input of 0.000 volts results in an output code of 10000000 00 and +4.99 volts at the input yields the 11111111 11 code. The nominal transfer curve is shown in Figure 6.

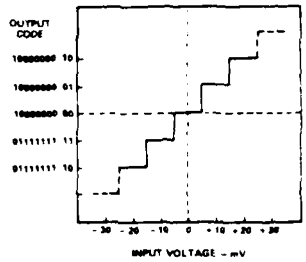


Figure 6. AD573 Transfer Curve – Bipolar Operation

Note that in the bipolar mode, the code transitions are offset ½LSB such that an input voltage of 0 volts ±5mV yields the code representing zero (10000000 00). Each output code is then *centered* on its nominal input voltage.

Full Scale Calibration

Full Scale Calibration is accomplished in the same manner as in Unipolar operation except the full scale input voltage is +4.985 volts.

Negative Full Scale Calibration

The circuit in Figure 4a can also be used in Bipolar operation to offset the input voltage (nominally −5V) which results in the 00000000 00 code. R2 should be omitted to obtain a symmetrical range.

The bipolar offset control input is not directly TTL compatible but a TTL interface for logic control can be constructed as shown in Figure 7.

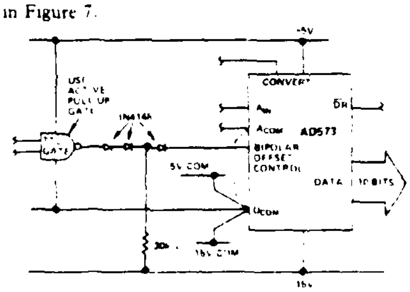


Figure 7. Bipolar Offset Controlled by Logic Gate
Gate Output = 1 Unipolar 0 – 10V Input Range
Gate Output = 0 Bipolar ±5V Input Range

SAMPLE-HOLD AMPLIFIER CONNECTION TO THE AD573

Many situations in high-speed acquisition systems or digitizing rapidly changing signals require a sample-hold amplifier (SHA) in front of the A-D converter. The SHA can acquire and hold a signal faster than the converter can perform a conversion. A SHA can also be used to accurately define the exact point in time at which the signal is sampled. For the AD573, a SHA can also serve as a high input impedance buffer.

Figure 8 shows the AD573 connected to the AD582 monolithic SHA for high speed signal acquisition. In this configuration, the AD582 will acquire a 10 volt signal in less than 10µs with a droop rate less than 100µV/ms.

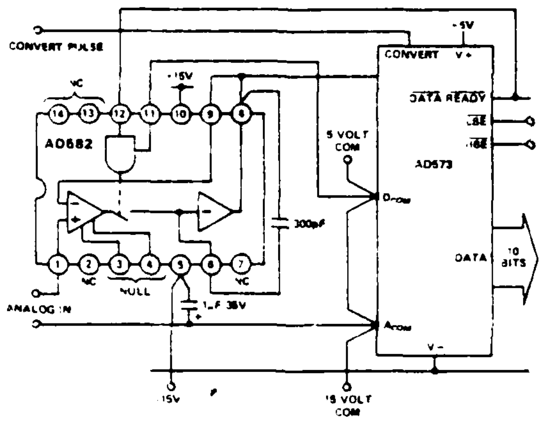


Figure 8. Sample-Hold Interface to the AD573

$\overline{DR}$ goes high after the conversion is initiated to indicate that reset of the SAR is complete. In Figure 8 it is also used to put the AD582 into the hold mode while the AD573 begins its conversion cycle. The AD582 settles to final value well in advance of the first comparator decision inside the AD573).

$\overline{DR}$ goes low when the conversion is complete placing the AD582 back in the sample mode. Configured as shown in Figure 8, the next conversion can be initiated after a 10µs delay to allow for signal acquisition by the AD582.

Observe carefully the ground, supply, and bypass capacitor connections between the two devices. This will minimize ground noise and interference during the conversion cycle.

GROUNDING CONSIDERATIONS

The AD573 provides separate Analog and Digital Common connections. The circuit will operate properly with as much as ±200mV of common mode voltage between the two commons. This permits more flexible control of system common bussing and digital and analog returns.

In normal operation, the Analog Common terminal may generate transient currents of up to 2mA during a conversion. In addition a static current of about 2mA will flow into Analog Common in the unipolar mode after a conversion is complete. The Analog Common current will be modulated by the variations in input signal.

The absolute maximum voltage rating between the two commons is ±1 volt. It is recommended that a parallel pair of back-to-back protection diodes be connected between the commons if they are not connected locally.

CONTROL AND TIMING OF THE AD573

The operation of the AD573 is controlled by three inputs: CONVERT, $\overline{HBE}$ and $\overline{LBE}$.

Starting a Conversion

The conversion cycle is initiated by a positive-going CONVERT pulse at least 500ns wide. The rising edge of this pulse resets the internal logic, clears the result of the previous conversion, and sets $\overline{DR}$ high. The falling edge of CONVERT begins the conversion cycle. When conversion is completed $\overline{DR}$ returns low. During the conversion cycle, $\overline{HBE}$ and $\overline{LBE}$ should be held high. If $\overline{HBE}$ or $\overline{LBE}$ goes low during a conversion, the data output buffers will be enabled and intermediate conversion results will be present on the data output pins. This may cause bus conflicts if other devices in a system are trying to use the bus.

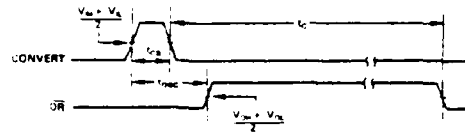


Figure 9. Convert Timing

Reading the Data

The three-state data output buffers are enabled by $\overline{HBE}$ and $\overline{LBE}$. Access time of these buffers is typically 150ns (250 maximum). The Data outputs remain valid until 50ns after the enable signal returns high, and are completely into the high-impedance state 100ns later.

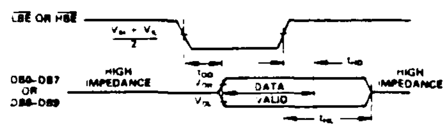


Figure 10. Read Timing

TIMING SPECIFICATIONS (All grades, $T_A = T_{min} - T_{max}$)

Parameter	Symbol	Min	Typ	Max	Units
CONVERT Pulse Width	t_{CS}	500	–	–	ns
$\overline{DR}$ Delay from CONVERT	t_{DSC}	–	1	1.5	µs
Conversion Time	t_C	10	20	30	µs
Data Access Time	t_{DD}	0	150	250	ns
Data Valid after $\overline{HBE}$ $\overline{LBE}$ High	t_{HD}	50	–	–	ns
Output Float Delay	t_{HL}	–	100	200	ns

MICROPROCESSOR INTERFACE CONSIDERATIONS – GENERAL

When an analog-to-digital converter like the AD573 is interfaced to a microprocessor, several details of the interface must be considered. First, a signal to start the converter must be generated; then an appropriate delay period must be allowed to pass before valid conversion data may be read. In most applications, the AD573 can interface to a microprocessor system with little or no external logic.

The most popular control signal configuration consists of decoding the address assigned to the AD573, then gating this signal with the system's $\overline{WR}$ signal to generate the CONVERT pulse, and gating it with RD to enable the output buffers. The use of a memory address and memory $\overline{WR}$ and RD signals denotes "memory-mapped" I/O interfacing, while the use of a separate I/O address space denotes "isolated I/O" interfacing. In 8-bit bus systems, the 10-bit AD573 will occupy two locations when data is to be read; therefore, two (usually consecutive) addresses must be decoded. One of the addresses can also be used as the address which produces the CONVERT signal during $\overline{WR}$ operations.

Figure 11 shows a generalized diagram of the control logic for

Interfacing to the AD573

an AD573 interfaced to an 8-bit data bus, where two addresses (ADC ADDR and ADC ADDR + 1) have been decoded. ADC ADDR starts the converter when written to (the actual data being written to the converter does not matter) and contains the high byte data during read operations. ADC ADDR + 1 performs no function during write operations, but contains the low byte data during read operations.

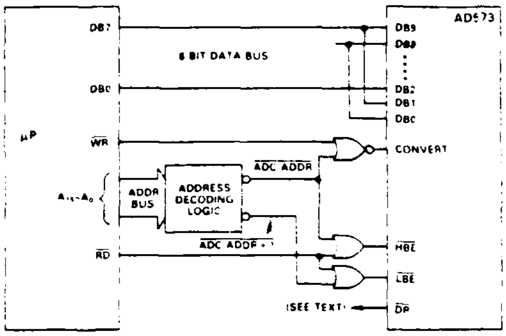


Figure 11. General AD573 Interface to 8-Bit Microprocessor

In systems where this read-write interface is used, at least 30 microseconds (the maximum conversion time) must be allowed to pass between starting a conversion and reading the results. This delay or "timeout" period can be implemented in a short software routine such as a countdown loop, enough dummy instructions to consume 30 microseconds, or enough actual useful instructions to consume the required time. In tightly-timed systems, the $\overline{DR}$ line may be read through an external three-state buffer to determine precisely when a conversion is complete. Higher-speed systems may choose to use $\overline{DR}$ to signal an interrupt to the processor at the end of a conversion.

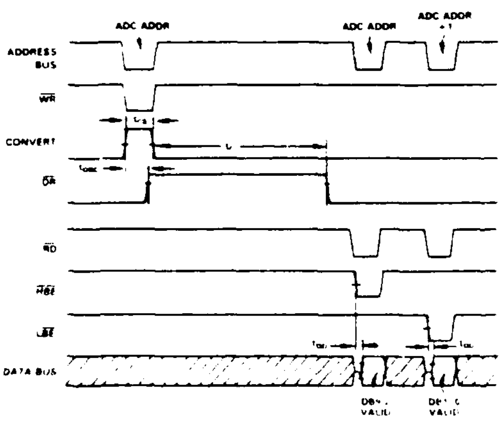


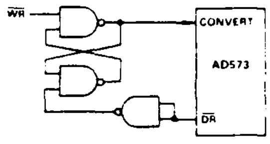
Figure 12. Typical AD573 Interface Timing Diagram

CONVERT Pulse Generation

The AD573 is tested with a CONVERT pulse width of 500ns and will typically operate with a pulse as short as 300ns. However, some microprocessors produce active WR pulses which are shorter than this. Either of the circuits shown in Figure 13 can be used to generate an adequate CONVERT pulse for the AD573.

In both circuits, the short low-going $\overline{WR}$ pulse sets the CONVERT line high through a flip-flop. The rising edge of $\overline{DR}$ (which signifies that the internal logic has been reset) resets the flip-flop and brings CONVERT low, which starts the conversion.

Note that t_{DSC} is slightly longer when the result of the previous conversion contains a logic 1 on the LSB. This means that the actual CONVERT pulse generated by the circuits in Figure 13 will vary slightly in width.



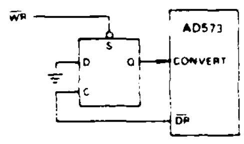


Figure 13a. Using 74LS00 *Figure 13b. Using 1/2 74LS74*

Output Data Format

The AD573 output data is presented in a left-justified format. The 8 MSBs (DB9–DB2, pins 10 through 3) are enabled by $\overline{HBE}$ (pin 20) and the 2 LSBs (DB1, DB0 – pins 2 and 1) are enabled by $\overline{LBE}$ (pin 19). This allows simple interface to 8-bit system buses by overlapping the 2 MSBs and the 2 LSBs. The organization of the data is shown in Figure 14.

When the least significant bits are read ($\overline{LBE}$ brought low), the six remaining bits of the byte will contain meaningless data. These unwanted bits can be masked by logically ANDing the byte with 11000000 (C0 hex), which forces the 6 lower bits to logic 0 while preserving the two most significant bits of the byte.

Note that it is not possible to reconfigure the AD573 for right justified data.

$\overline{HBE}$	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2
$\overline{LBE}$	DB1	DB0	X	X	X	X	X	X

Figure 14. AD573 Output Data Format

In systems where all 10 bits are desired at the same time, $\overline{HBE}$ and $\overline{LBE}$ may be tied together. This is useful in interfacing to 16-bit bus systems. The resulting 10-bit word can then be placed at the high end of the 16-bit bus for left justification or at the low end for right justification.

It is also possible to use the AD573 in a "stand-alone" mode, where the output data buffers are automatically enabled at the end of a conversion cycle. In this mode, the $\overline{DR}$ output is wired to the $\overline{HBE}$ and $\overline{LBE}$ inputs. The outputs thus are forced into the high-impedance state during the conversion period, and valid data becomes available approximately 500ns after the $\overline{DR}$ signal goes low at the end of the conversion. The 500ns delay allows propagation of the least significant bit through the internal logic.

This mode is particularly useful for bench-testing of the AD573, and in applications where dedicated I/O ports of peripheral interface adapter chips are available.

AD573 A/D Converter Data Sheet (continued)

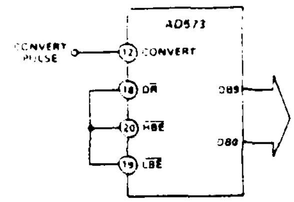


Figure 15. AD573 in "Stand-Alone" Mode (Output Data Valid 500ns After $\overline{DR}$ Goes Low)

Apple II Microcomputer Interface

The AD573 can provide a flexible, low-cost analog interface for the popular Apple II microcomputer. The Apple II, based on a 1MHz 6502 microprocessor, meets all timing requirements for the AD573. Only a few TTL gates are required to decode the signals available on the Apple II's peripheral connector. The recommended connections are shown in Figure 16.

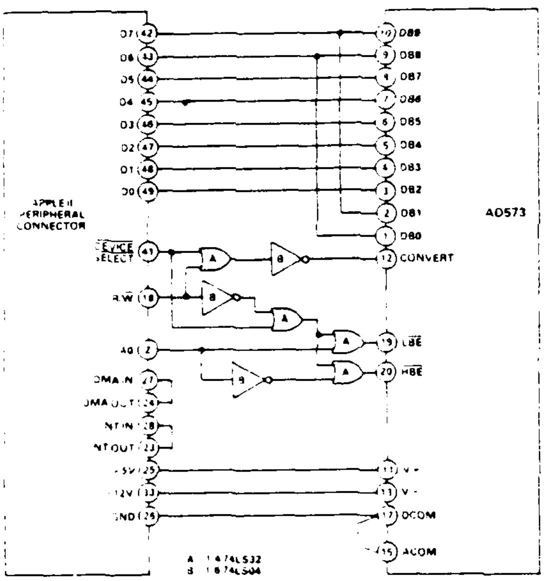


Figure 16. AD573 Interface to Apple II

The BASIC routine listed here will operate the AD573 circuit shown in Figure 16. The conversion is started by POKEing to the location which contains the AD573. The relatively slow execution speed of BASIC eliminates the need for a delay routine between starting and reading the converter. This routine assumes that the AD573 is connected for a ±5 volt input range. Variable I represents the integer value (from 0 to 1023) read from the AD573. Variable V represents the actual value of the input signal (in volts).

```
100 PRINT "WHICH SLOT IS THE A/D IN";: INPUT S
110 A = 49280 + 16*S
120 POKE A,0
130 L = PEEK(A): H = PEEK(A+1)
140 I = 4*H + INT(L/64)
150 V = I/1024*10-5
160 PRINT "THE INPUT SIGNAL IS ";V;" VOLTS."
```

It is also possible to write a faster-executing assembly-language routine to control the AD573. Such a routine will require a delay between starting and reading the converter. This can be easily implemented by calling the Apple's WAIT subroutine (which resides at location $FCA8) after loading the accumulator with a number greater than or equal to two.

8085-Series Microprocessor Interface

The AD573 can also be used with 8085-series microprocessors. These processors use separate control signals for RD and WR, as opposed to the single R/$\overline{W}$ control signal used in the 6800/6500 series processors.

There are two constraints related to operation of the AD573 with 8085-series processors. The first problem is the width of the CONVERT pulse. The circuit shown in Figure 17 (essentially the same as that shown in Figure 13) will produce a wide enough CONVERT pulse when the 8085 is running at 5MHz. For 8085 systems running at slower clock rates (3MHz), the flip-flop-based circuit can be eliminated since the $\overline{WR}$ pulse will be approximately 500ns wide.

The other consideration is the access time of the AD573's three-state output data buffers, which is 250ns maximum. It may be necessary to insert wait states during RD operations from the AD573. This will not be a problem in systems using memories with comparable access times, since wait states will have already been provided in the basic system design.

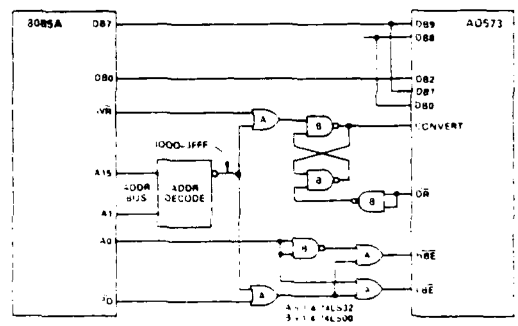


Figure 17 AD573–8085A Interface Connections

The following assembly-language subroutine can be used to control an AD573 residing at memory locations 3000_H and 3001_H. The 10 bits of data are returned (left-justified) in the DE register pair.

```
ADC.   LXI  H,3000   ;LOAD HL WITH AD573 ADDRESS
       MOV  M,A      ;START CONVERSION
       MVI  B,06     ;LOAD DELAY PERIOD
LOOP:  DCR  B        ;DELAY LOOP
       JNZ  LOOP     ;
       MOV  A,M      ;READ LOW BYTE
       ANI  C0       ;MASK LOWER 6 BITS
       MOV  E,A      ;STORE CLEAN LOW BYTE IN E
       INR  L        ;LOAD HIGH BYTE ADDRESS
       MOV  D,M      ;MOVE HIGH BYTE TO D
       RET           ;EXIT
```

AD544 Operational Amplifier Data Sheet

ANALOG DEVICES

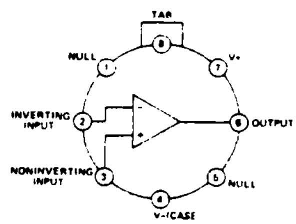
High Speed Implanted FET-Input Op Amp

AD544

FEATURES

Low Bias Current: 25pA max, warmed-up
Low Offset Voltage: 500μV max
Low Offset Voltage Drift: 5μV/°C max
Low Input Voltage Noise: 2μV p-p
Low Quiescent Current: 2.5mA max
High Slew Rate: 13V/μs
Fast Settling to ±0.01%: 3μs
Low Total Harmonic Distortion: 0.0015% at 1kHz

AD544 FUNCTIONAL BLOCK DIAGRAM



TO-99
TOP VIEW

PRODUCT DESCRIPTION

The AD544 is a high speed monolithic FET-input operational amplifier fabricated with the most advanced bipolar, JFET and laser trimming technologies. The AD544 offers bias currents significantly lower than currently available monolithic FET-input devices: 25pA max, warmed-up for the AD544K and L, 50pA max for the AD544J. In addition, the offset voltage is *laser trimmed to less than* 0.5mV *on the* AD544L and 1.0mV on the AD544K utilizing Analog's laser-wafer-trimming (LWT) process. When combined with the AD544's low offset voltage drift (5μV/°C max for "L", 10μV/°C max for "K"), these features offer the user IC performance truly superior to existing FET-input op amps—and at low, monolithic pricing.

The key technology required for monolithic JFET-input op amps is the ion-implanted JFET. Ion-implantation (as opposed to diffusion) permits the fabrication of precision, matched JFET's on a monolithic bipolar chip. Analog Devices optimizes the process to produce bias currents lower than other popular FET-input op amps and specifies each device for the maximum value at either input in the fully warmed-up condition. Additional benefits of this optimization include low voltage noise (2μV p-p, 0.1–10Hz), and low quiescent current.

The AD544 is recommended for any operational amplifier application requiring excellent ac and dc performance at low cost. The 2MHz bandwidth and low offset of the AD544 make it an excellent choice as an output amplifier for current output D/A Converters such as the AD7541, 12-Bit CMOS DAC. High common mode rejection (80dB, min on the "K" and "L" versions) and open-loop gain ensures better than "12-bit" linearity in high impedance buffer applications.

The AD544 is available in four versions: the "J", "K" and "L" are specified over the 0 to +70°C temperature range and the "S" over the -55°C to +125°C operating temperature range. All devices are packaged in the hermetically-sealed, TO-99 metal can.

PRODUCT HIGHLIGHTS

1. Improved bipolar and JFET processing on the AD544 results in the lowest bias current available in a high speed monolithic FET op amp.
2. Analog Devices, unlike some manufacturers, specifies each device for the maximum bias current at either input in the warmed-up condition, thus assuring the user that the AD544 *will* meet its published specifications *in actual use*.
3. Laser-wafer-trimming reduces offset voltage to as low as 0.5mV max (AD544L), thus eliminating the need for external nulling in many situations.
4. If offset nulling is required, the additional offset voltage drift induced will be minimal. (In some devices, offset voltage drift can increase an additional 3μV/°C per mV of offset nulled.)
5. Low voltage noise (2μV, p-p), and low offset voltage drift (5μV/°C) enhance the AD544's performance as a precision op amp.
6. The high slew rate (13.0V/μs) and fast settling time to 0.01% (3.0μs) make the AD544 ideal for D/A, A/D, sample-hold circuits and high speed integrators.
7. Low harmonic distortion (0.0015%) makes the AD544 an ideal choice for audio applications.

AD544 Operational Amplifier Data Sheet (continued)

SPECIFICATIONS (@ +25°C and $V_S = \pm 15V$ dc)

Model	AD544J			AD544K			AD544L			AD544S			
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Units
OPEN LOOP GAIN[1]													
$V_O = \pm 10V$, $R_L \geq 2k\Omega$	**30,000**			**50,000**			**50,000**			**50,000**			V/V
T_{min} to T_{max}, $R_L = 2k\Omega$	20,000			40,000			40,000			**20,000**			V/V
OUTPUT CHARACTERISTICS													
Voltage @ $R_L = 2k\Omega$, T_{min} to T_{max}	**±10**	±12		**±10**	±12		**±10**	±12		**±10**	±12		V
Voltage @ $R_L = 10k\Omega$, T_{min} to T_{max}	**±12**	±13		**±12**	±13		**±12**	±13		**±12**	±13		V
Short Circuit Current		25			25			25			25		mA
FREQUENCY RESPONSE													
Unity Gain Small Signal		2.0			2.0			2.0			[illegible]		MHz
Full Power Response		200			200			200			200		kHz
Slew Rate, Unity Gain	**8.0**	13.0		**8.0**	13.0		**8.0**	13.0		8.0	13.0		V/μs
Settling Time to 0.01%		3.0			3.0			3.0			3.0		μs
Total Harmonic Distortion		0.0025			0.0025			0.0025			0.0025		%
INPUT OFFSET VOLTAGE[2]													
Initial Offset			**2.0**			**1.0**			**0.5**			**1.0**	mV
Input Offset Voltage vs. Temp. or T_{min} to T_{max}			**20**			**10**			**5**			**15**	μV/°C
Input Offset Voltage vs. Supply, T_{min} to T_{max}			**200**			**100**			**100**			**100**	μV/V
INPUT BIAS CURRENT[3]													
Either Input		10	**50**		10	**25**		10	**25**		10	**25**	pA
Offset Current		5			2			2			2		pA
INPUT IMPEDANCE													
Differential[4]		$10^{12}\|6$			$10^{12}\|6$			$10^{12}\|6$			$10^{12}\|6$		MΩ‖pF
Common Mode		$10^{12}\|3$			$10^{12}\|3$			$10^{12}\|3$			$10^{12}\|3$		MΩ‖pF
INPUT VOLTAGE RANGE													
Differential		±20			±20			±20			±20		V
Common Mode	**±10**	±12		**±10**	±12		**±10**	±12		**±10**	±12		V
Common Mode Rejection	**74**			**80**			**80**			**80**			dB
INPUT NOISE													
Voltage 0.1Hz to 10Hz		2			2			2			2		μV p-p
f = 10Hz		35			35			35			35		nV/√Hz
f = 100Hz		22			22			22			22		nV/√Hz
f = 1kHz		18			18			18			18		nV/√Hz
f = 10kHz		16			16			16			16		nV/√Hz
POWER SUPPLY													
Rated Performance		±15			±15			±15			±15		V
Operating	±5		±18	±5		±18	±5		±18	±5		±18	V
Quiescent Current		1.8	2.5		1.8	2.5		1.8	2.5		1.8	2.5	mA
TEMPERATURE RANGE													
Operating, Rated Performance	0		+70	0		+70	0		+70	−55		+125	°C
Storage	−65		+150	−65		+150	−65		+150	−65		+150	°C
PACKAGE[5]													
TO-99 Style (H08B)	AD544JH			AD544KH			AD544LH			AD544SH			

NOTES

[1]Open Loop Gain is specified with V_{OS} both nulled and unnulled

[2]Input Offset Voltage specifications are guaranteed after 5 minutes of operation at $T_A = +25°C$

[3]Bias Current specifications are guaranteed at maximum at either input after 5 minutes of operation at $T_A = +25°C$. For higher temperatures, the current doubles every 10°C

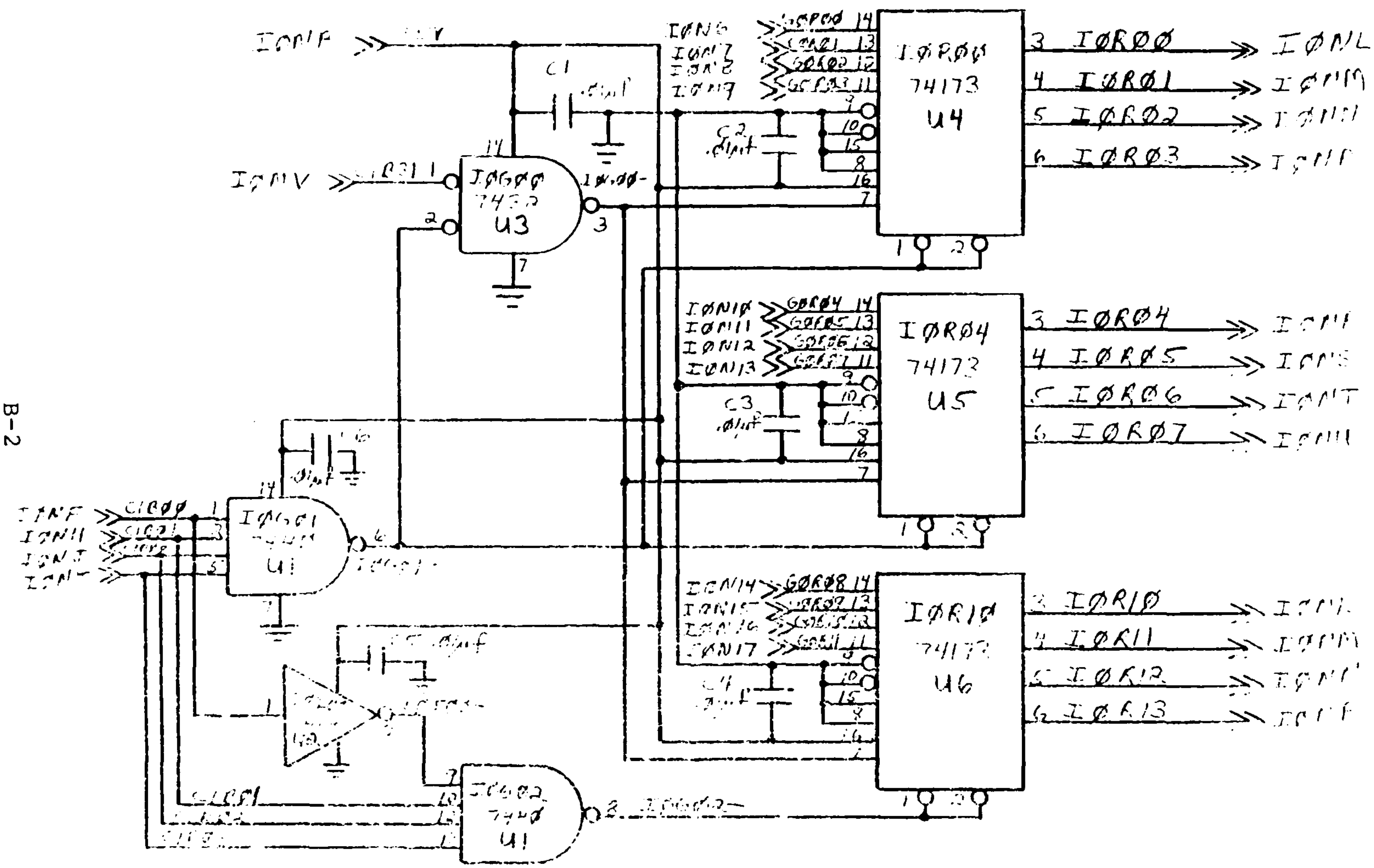
[4]Defined as voltage between inputs, such that neither exceeds ±10V from ground

[5]See Section 19 for package outline information

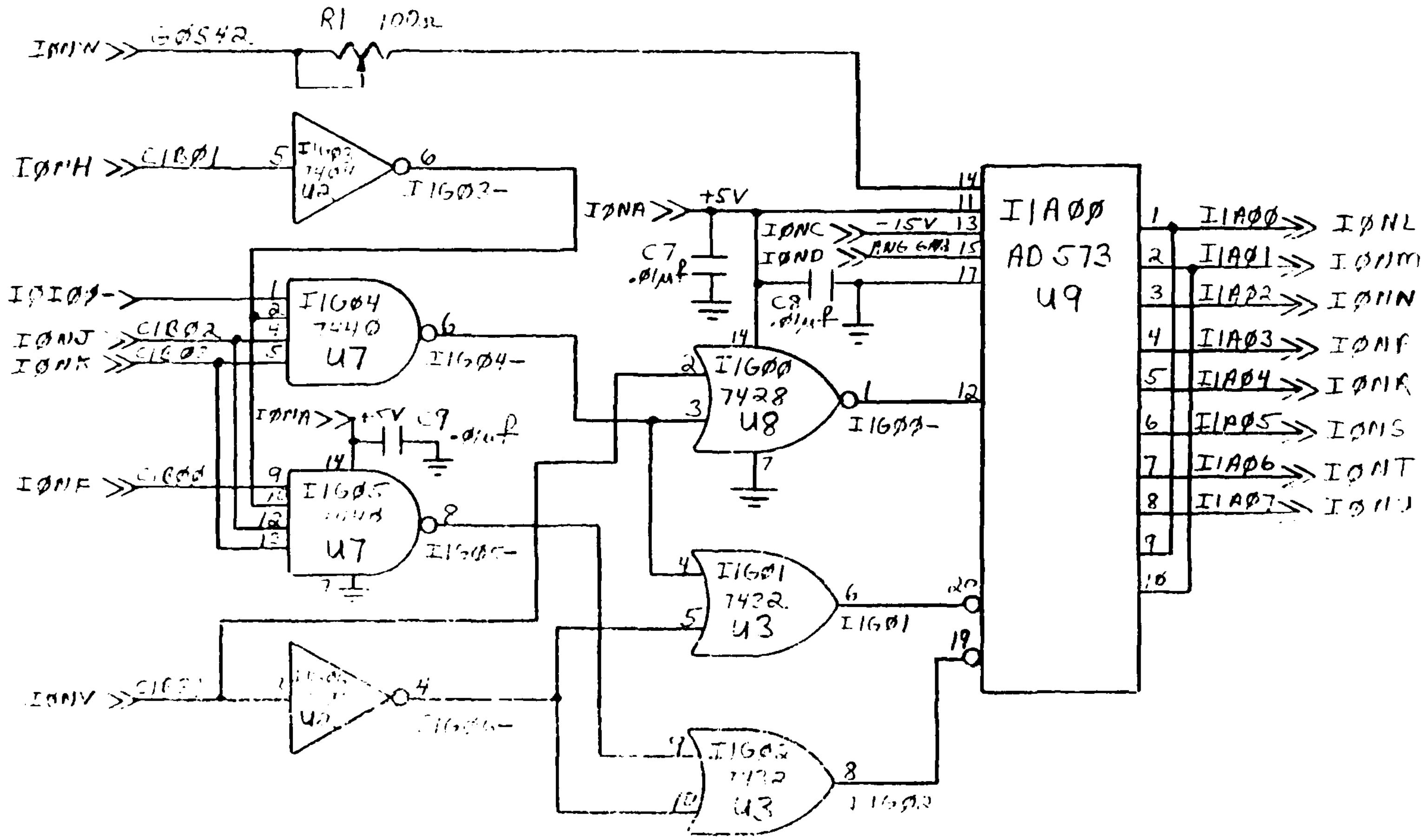
Specifications subject to change without notice

Specifications shown in boldface are tested on all production units at final electrical test. Results from those tests are used to calculate outgoing quality levels. All min and max specifications are guaranteed, although only those shown in boldface are tested on all production units

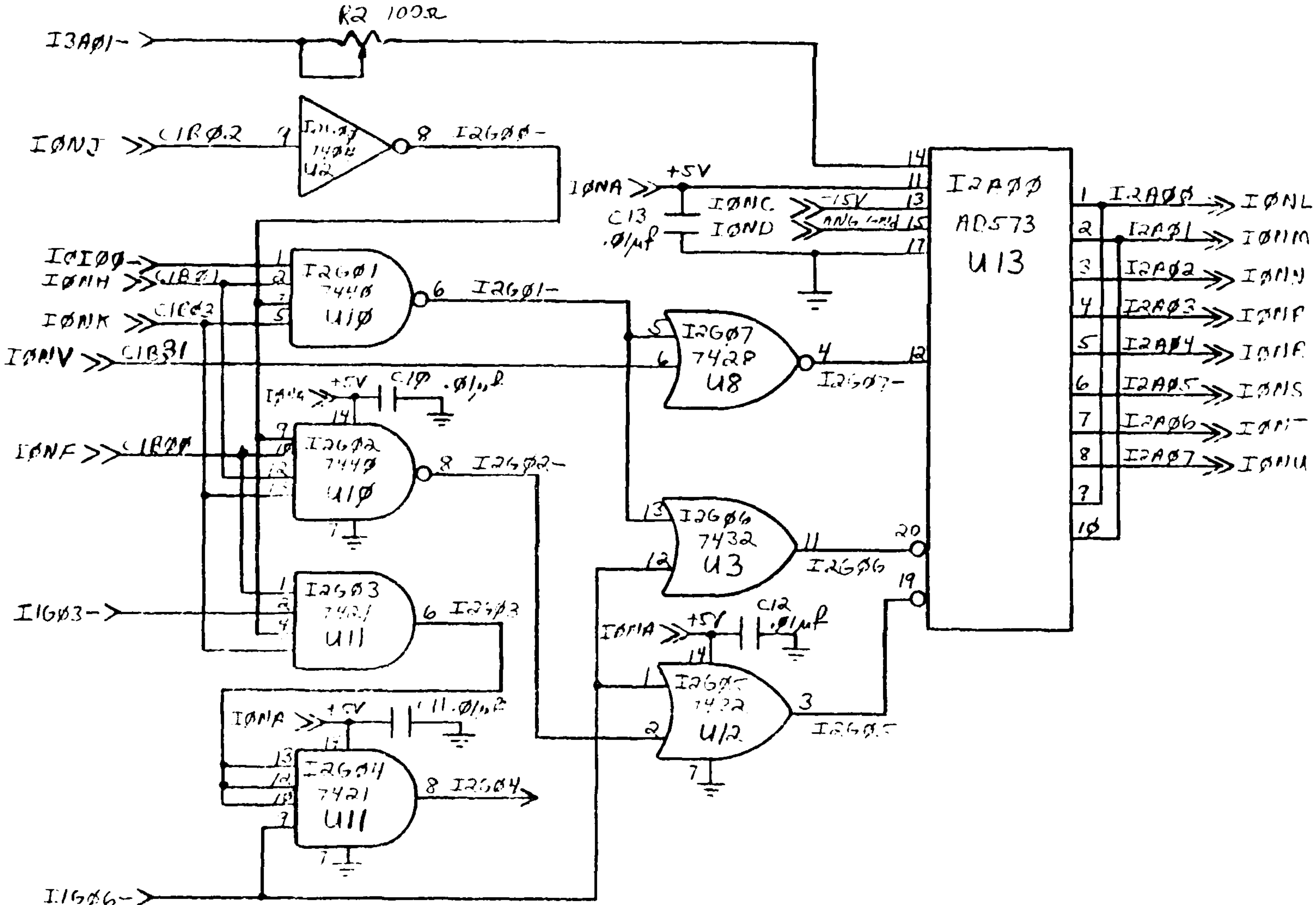
APPENDIX B



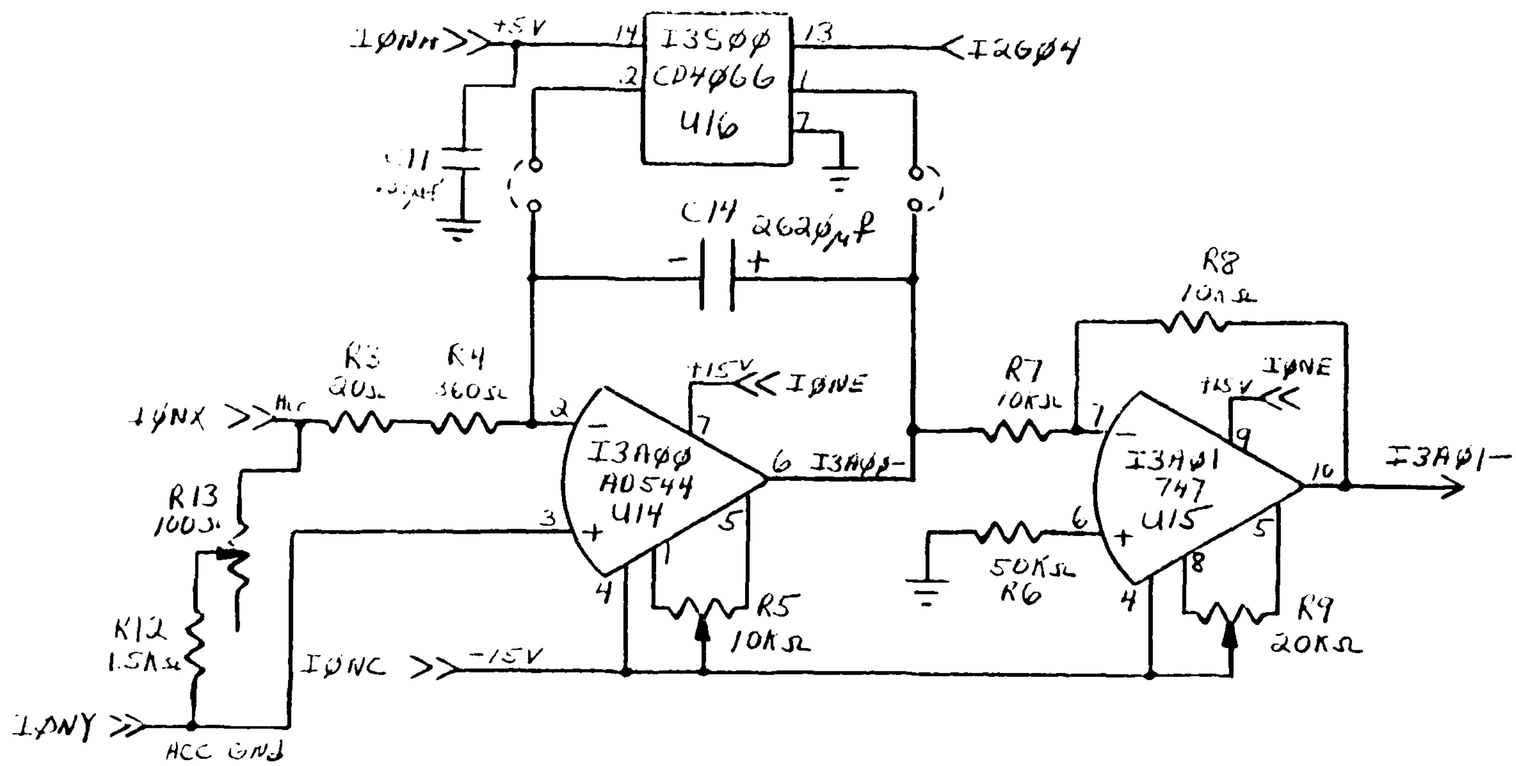
DIGITAL INTERFACE SCHEMATIC DIAGRAMS



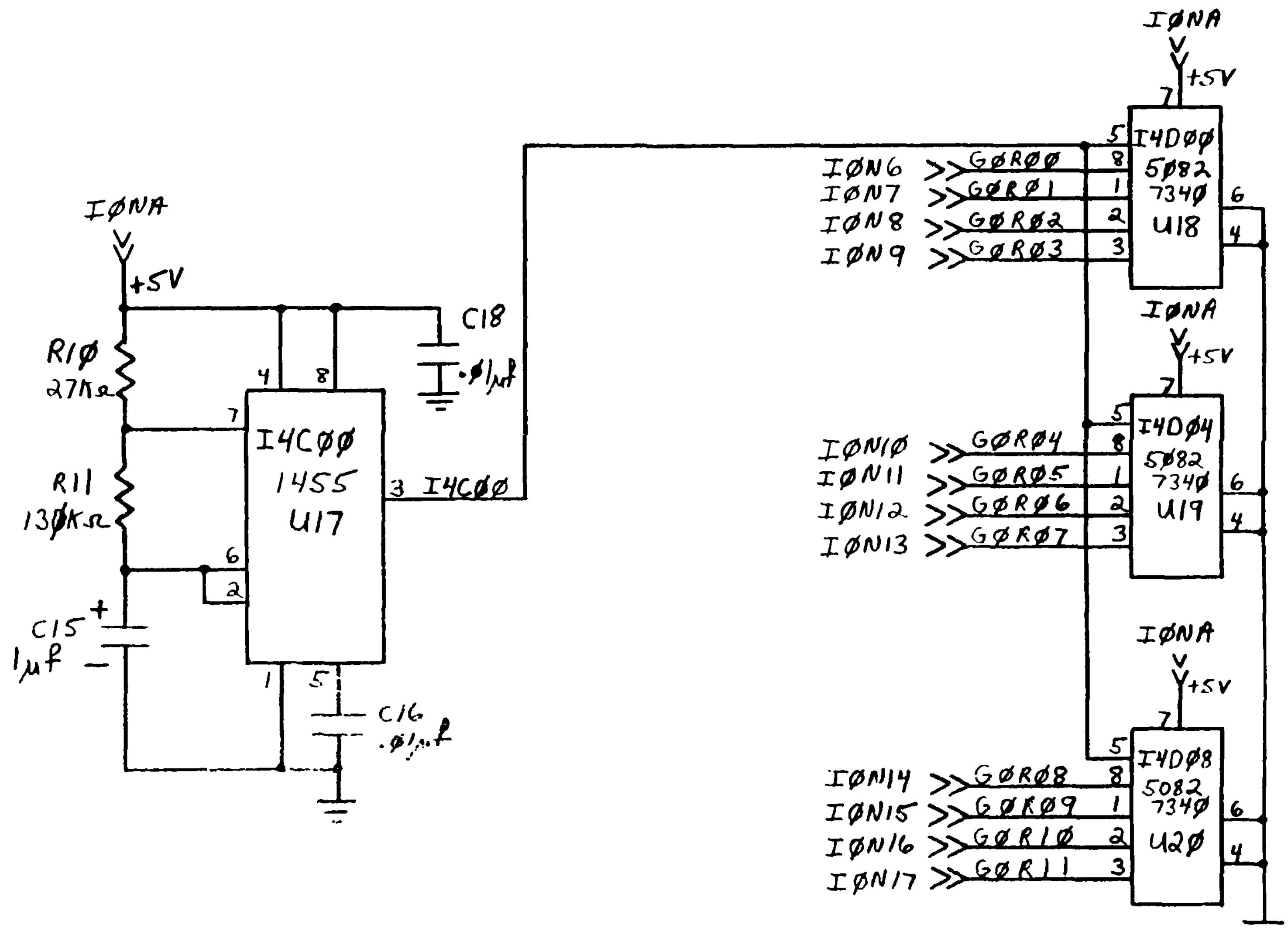
DIGITAL INTERFACE SCHEMATIC DIAGRAMS (continued)



DIGITAL INTERFACE SCHEMATIC DIAGRAMS (continued)



DIGITAL INTERFACE SCHEMATIC DIAGRAMS (continued)



DIGITAL INTERFACE SCHEMATIC DIAGRAMS (continued)

Circuit Card I: Digital Interface
Parts List

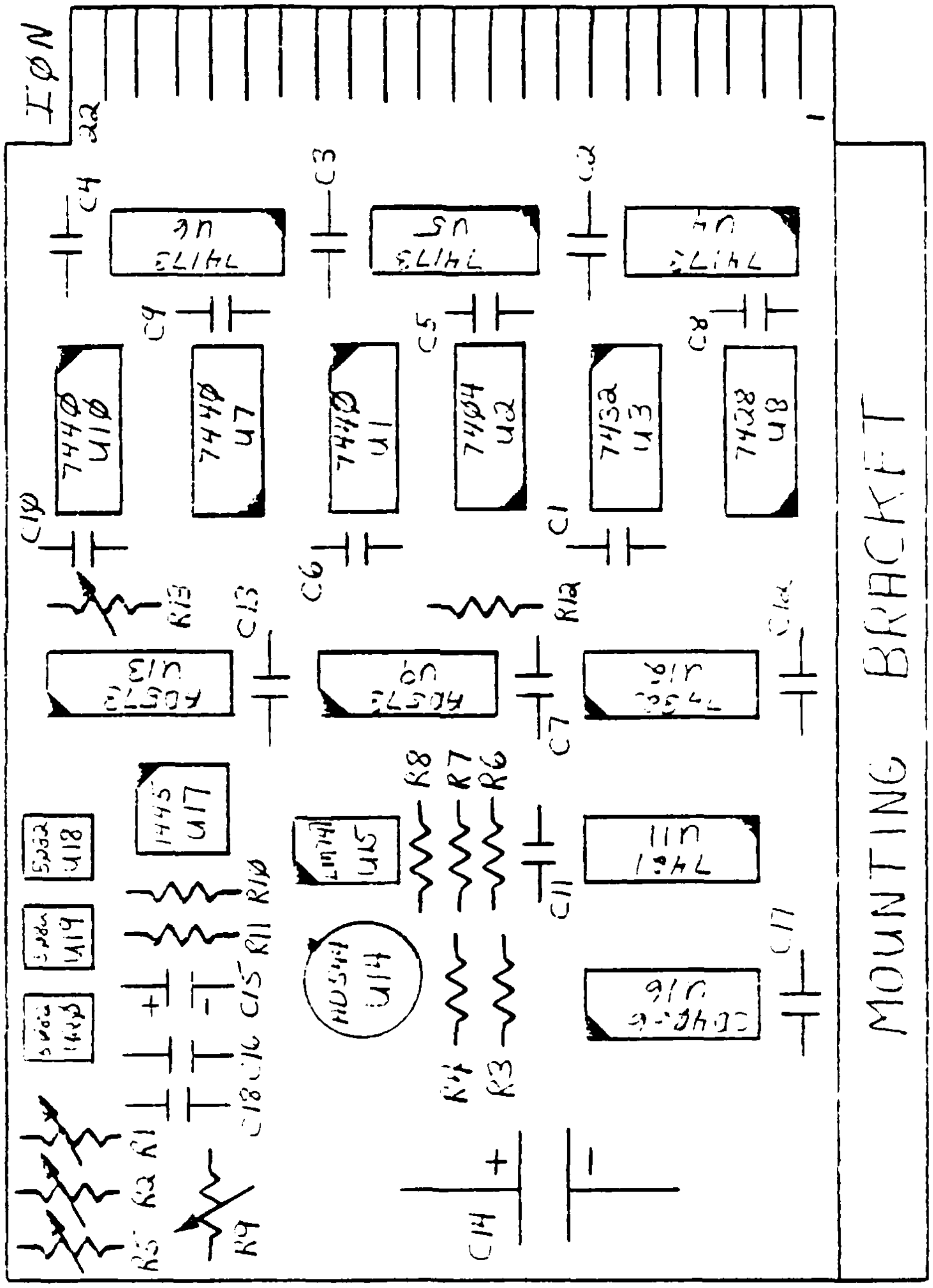
Part Number	Type	Schematic Reference #
7440	Dual 4-Input Positive NAND Buffers	U1 U7 U10
7404	Hex Inverters	U2
7432	Quadruple 2-Input Positve-OR Gates	U3 U12
74173	4-Bit D-Type Register With 3-State Outputs	U4 U5 U6
7428	Quadruple 2-Input Positive-NOR Gates	U8
AD 573	10-Bit Analog to Digital Converter	U9 U13
7421	Dual 4-Input Positive AND Gates	U11
AD 544	Precision Operational Amplifier	U14
LM 747	Dual Operational Amplifiers	U15
CD 4066	Quad Bilateral Switch	U16
1445	Dual Monostable Multivibrator	U17

Circuit Card I: Digital Interface
Parts List (Continued)

Part Number	Type	Schematic Reference #
5082/7340	Single Digit HEX LED Display with Latches and Driver	U18 U19 U20
100	Variable Resistor (Ohms)	R1 R2 R13
20	Resistor (Ohms)	R3
360	Resistor (Ohms)	R4
10K	Variable Resistor (Ohms)	R5
50K	Resistor (Ohms)	R6
10K	Resistor (Ohms)	R7 R8
20K	Variable Resistor (Ohms)	R9
27K	Resistor (Ohms)	R10
130K	Resistor (Ohms)	R11
1.5K	Resistor (Ohms)	R12

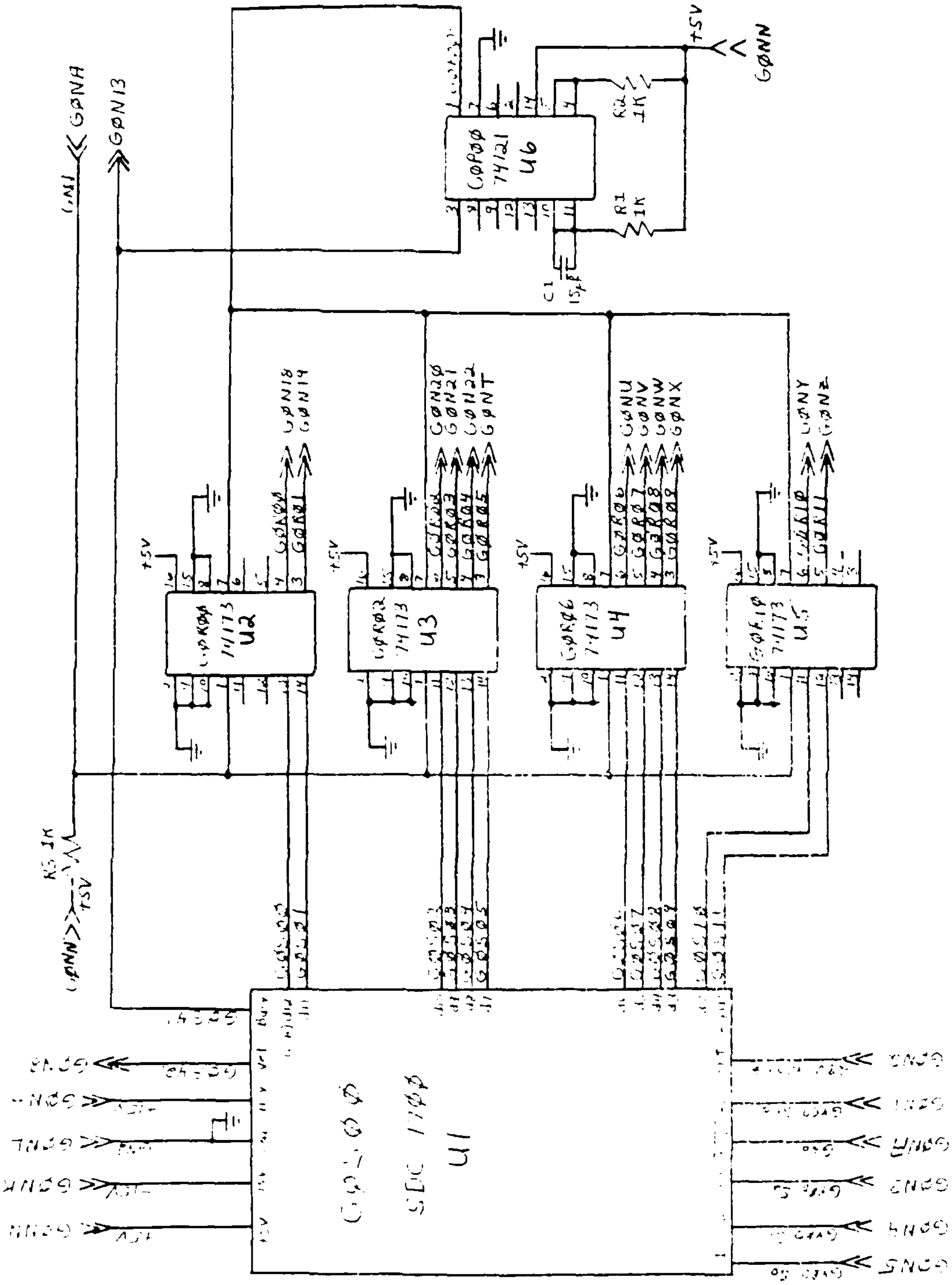
Circuit Card I: Digital Interface
Parts List (Continued)

Part Number	Type	Schematic Reference #
0.01 Micro	Capacitor (Farads)	C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C16 C17 C18
2220 Micro	Capacitor (Farads)	C14
1.0	Capacitor (Farads)	C15



DIGITAL INTERFACE DEVICE LAYOUT

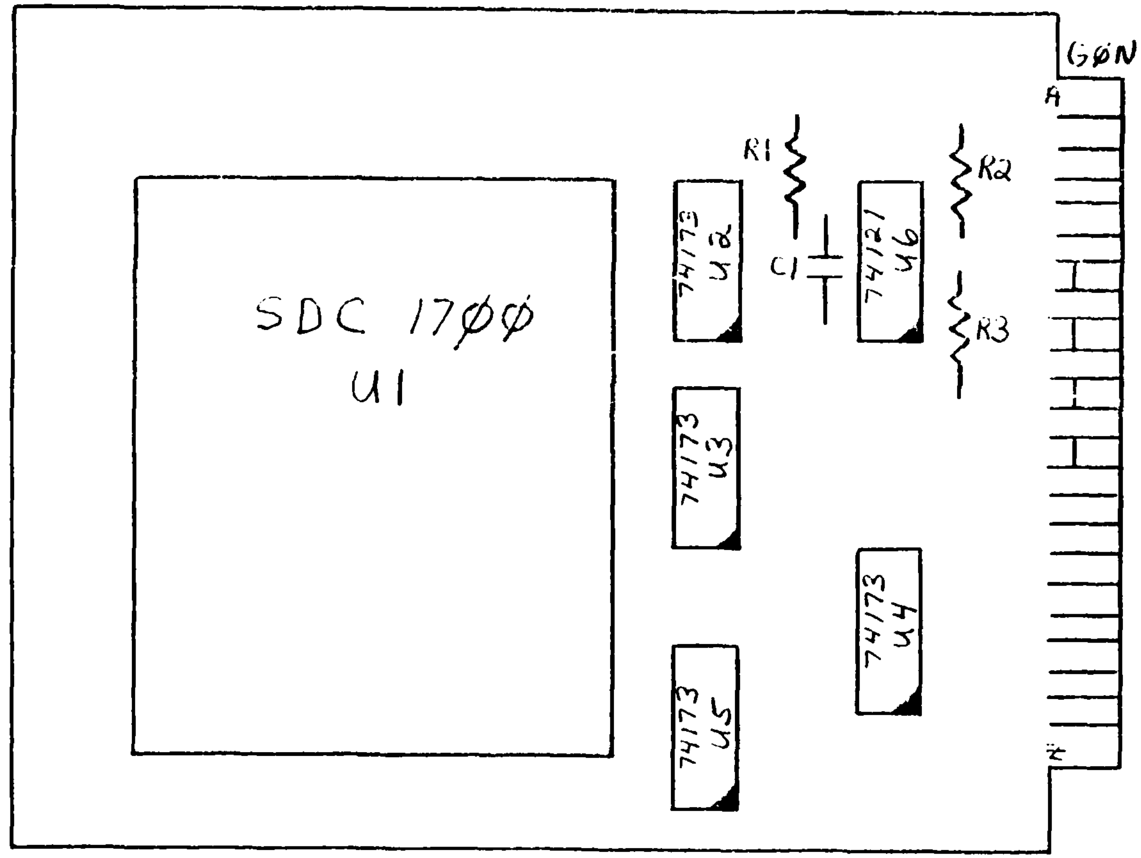
APPENDIX C



SYNCRO TO DIGITAL SCHEMATIC DIAGRAM

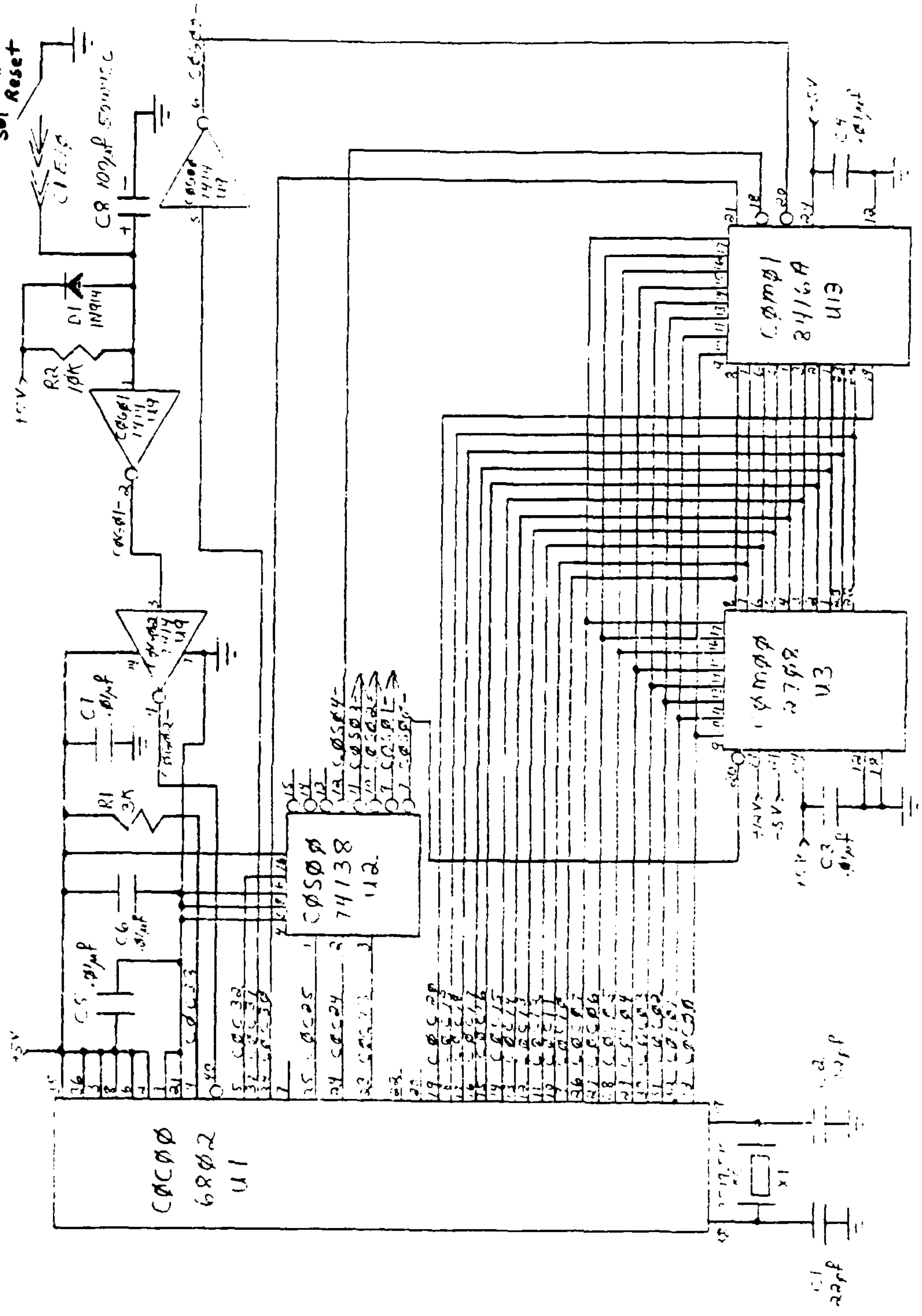
Circuit Card G: Syncro to Digital
Parts List

Part Number	Type	Schematic Reference #
SDC 1700	Syncro to Digital Converter	U1
74173	4-Bit D-Type Register With 3-State Outputs	U2 U3 U4 U5
74121	Monostable Multivibrator	U6
1K	Resistor (Ohms)	R1 R2 R3
15 pico	Capacitor (Farads)	C1

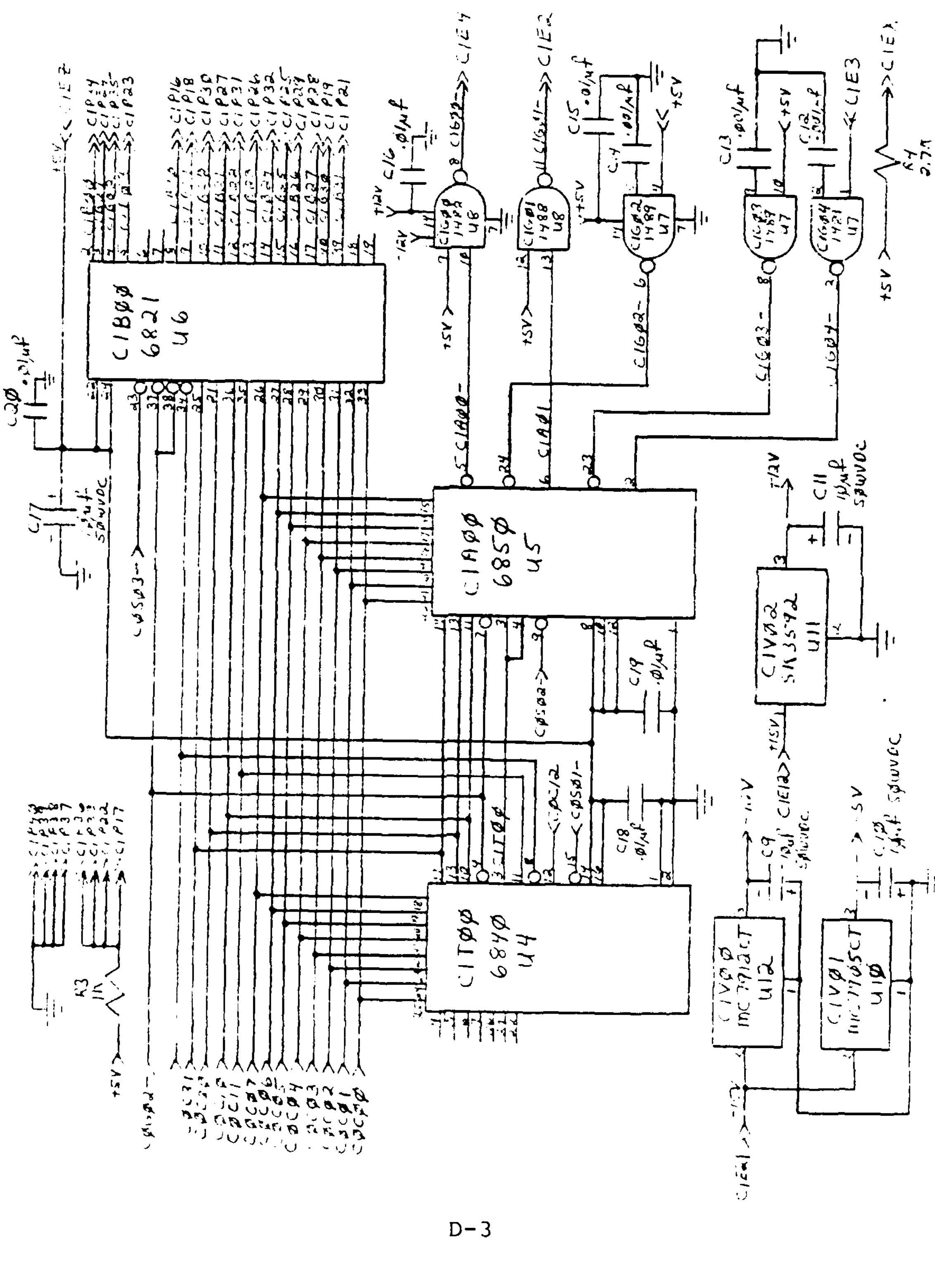


SYNCRO TO DIGITAL DEVICE LAYOUT

APPENDIX D



GYRAC COMPUTER SCHEMATIC DIAGRAMS



GYRAC COMPUTER SCHEMATIC DIAGRAMS (continued)

Circuit Card C: Computer Controller Parts List

Part Number	Type	Schematic Reference #
MC6802	Microprocessor with Clock and 128 Bytes RAM (CPU)	U1
74138	3-8 Line Decoders-Multiplexers	U2
2708	1024 x 8 bit U.V. Erasable PROM	U3
MC6840	Programmable Timer Module (PTM)	U4
MC6850	Asynchronous Communications Interface Adapter (ACIA)	U5
MC6821	Peripheral Interface Adapter (PIA)	U6
1489	Quad Line Receiver	U7
1488	Quad Line Driver	U8
7414	Hex Schmitt-Trigger Inverters	U9
MC7905CT	Negative 5 Volt Voltage Regulator	U10
SK3592	Positive 12 Volt Voltage Regulator	U11

Circuit Card C: Computer Controller
Parts List (Continued)

Part Number	Type	Schematic Reference #
MC7912CT	Negative 12 Volt Voltage Regulator	U12
MB8416A	CMOS 2048 x 8 Byte Static RAM	U13
1N914	Signal Diode	D1
3579.545 KC	Crystal Oscillator	X1
3K	Resistor (Ohms)	R1
10K	Resistor (Ohms)	R2
1K	Resistor (Ohms)	R3
2.7K	Resistor (Ohms)	R4
22 Pico	Capacitor (Farads)	C1 C2
0.01 Micro	Capacitor (Farads)	C3 C4 C5 C6 C7 C15 C16 C18 C19 C20
100 Micro	Capacitor (Farads) 50 WVDC	C8

Circuit Card C: Computer Controller
Parts List (Continued)

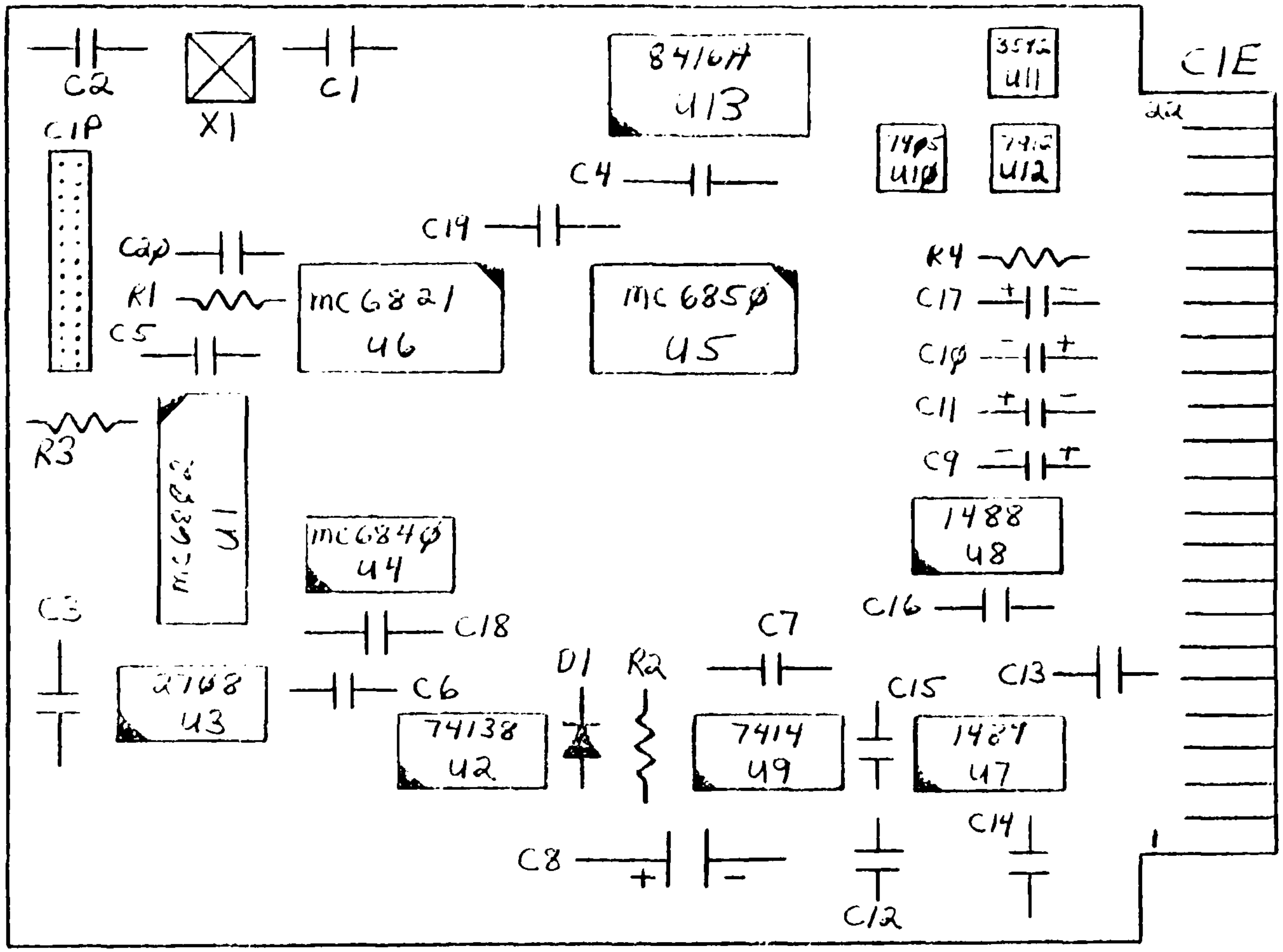
Part Number	Type	Schematic Reference #
10 Micro	Capacitor (Farads) 50 WVDC	C9 C10 C11 C17
0.001 Micro	Capacitor (Farads)	C12 C13 C14

GYRAC COMPUTER MEMORY MAP

ADDRESS (HEX)	DEVICE
0000 to 007F	0128 Bytes of on Processor Scratchpad RAM
6000 to 67FF	2048 Bytes of RAM (Alternate Addresses 7000-77FF)
8000	PIA Data Direction/Peripheral Register A (Alternate Address 9000)
8001	PIA Control Register A (Alternate Address 9001)
8002	PIA Data Direction/Peripheral Register B (Alternate Address 9002)
8003	PIA Control Register B (Alternate Address 9003)
A000	ACIA Control/Status Register (Alternate Address B000)
A001	ACIA TX Data/RX Data Register (Alternate Address B001)
C000	PTM Write Control Register #3/#1 (Alternate Address D000)
C001	PTM Write Control #2/Status Registers (Alternate Address D001)
C002	PTM MSB Buffer #1 Register/Timer #1 Counter (Alternate Address D002)
C003	PTM Timer #1 Latches/LSB Buffer #1 Register (Alternate Address D003)
C004	PTM MSB Buffer #2 Register/Timer #2 Counter (Alternate Address D004)
C005	PTM timer #2 Latches/LSB Buffer #2 Register (Alternate Address D005)

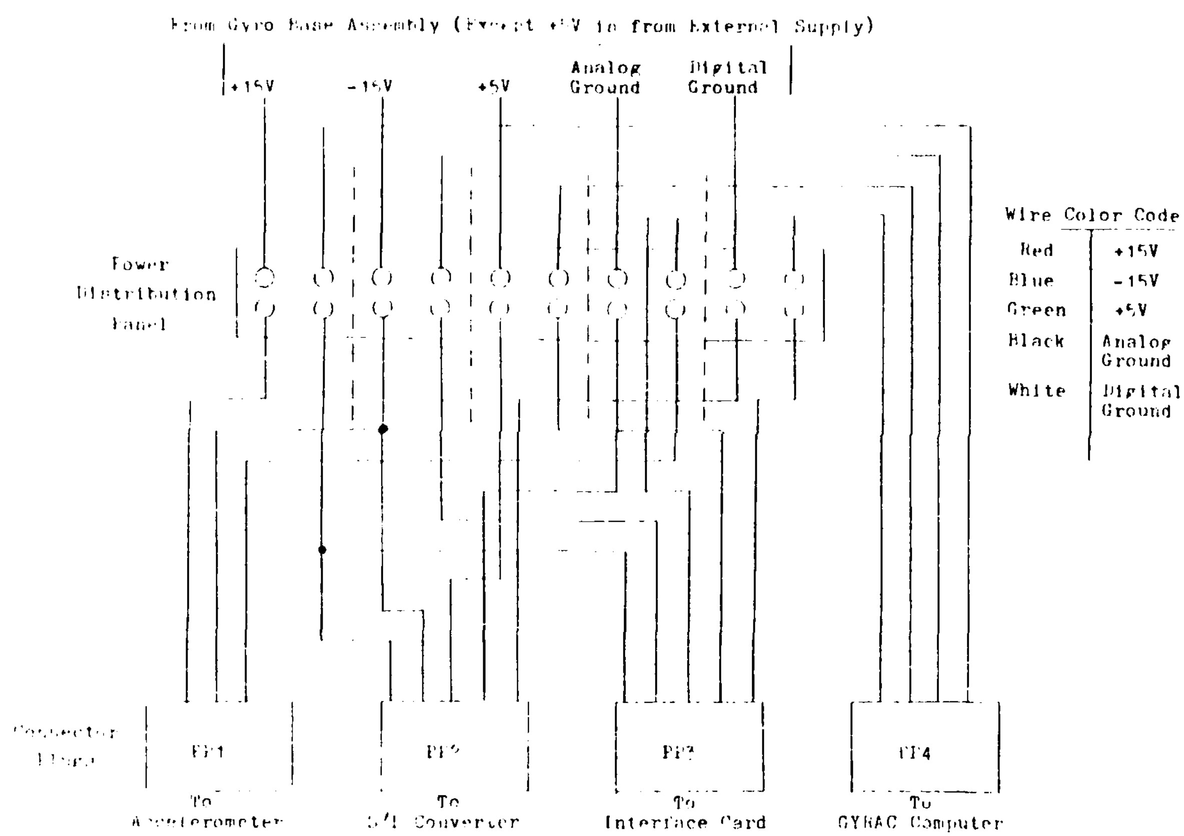
GYRAC COMPUTER MEMORY MAP (continued)

ADDRESS (HEX)	DEVICE
C0060	PTM MSB Buffer #3 Register/Timer #3 Counter (Alternate Address D006)
C007	PTM Timer #3 Latches/LSB Buffer #3 Register (Alternate Address D007)
E000 to E3FF	1024 Bytes of GYRAC Control Program in EPROM (Alternate Addresses F000-F3FF)
E3FE to E3FF	Address of Reset Vector
E3FC to E3FD	Address of Non-maskable Interrupt Vector
E3FA to E3FB	Address of Software Interrupt Vector
E3F8 to E3F9	Address of Interrupt Vector



GYRAC COMPUTER DEVICE LAYOUT

APPENDIX E



GYRAC POWER PANEL

GON: Syncro to Digital Edge Connector

Pin Number	Signal Name	Description
GON1	None	GYRO Base Ground
GON2	None	GYRO Base 28 Volts 400 Hertz
GON3	None	GYRO Indicator S2
GON4	None	GYRO Indicator S1
GON5	None	GYRO Indicator S0
GON6		Unused
GON8	GOS42	Angular Velocity Analog Signal
GON13	GOS41	S to D Busy Signal (unused)
GON15		Unused
GON16		Unused
GON17		Unused
GON18	GOR00	D0 Data Bit
GON19	GOR01	D1 Data Bit
GON20	GOR02	D2 Data Bit
GON21	GOR03	D3 Data Bit
GON22	GOR04	D4 Data Bit
GONA	None	Digital Ground
GONF	None	+15 Volts DC
GONH	None	+15 Volts DC
GONJ	None	-15 Volts DC
GONK	None	-15 Volts DC
GONL	None	Analog Ground
GONM	None	Analog Ground
GONN	None	+5 Volts DC
GONP	None	+5 Volts DC
GONT	GOR05	D5 Data Bit
GONU	GOR06	D6 Data Bit
GONV	GOR07	D7 Data Bit
GONW	GOR08	D8 Data Bit
GONX	GOR09	D9 Data Bit
GONY	GOR10	D10 Data Bit
GONZ	GOR11	D11 Data Bit

C1P: GYRAC Computer Sensor Bus Connector

Pin Number	Signal Name	Description
C1P1		Unused
C1P2		Unused
C1P3		Unused
C1P4		Unused
C1P5		Unused
C1P6		Unused
C1P7		Unused
C1P8		Unused
C1P9		Unused
C1P10		Unused
C1P11		Unused
C1P12		Unused
C1P13		Unused
C1P14		Unused
C1P15		Unused
C1P16	C1B10	Reserved (unused)
C1P17	None	+5 Volts (thru pullup resistor)
C1P18	C1B11	Reserved (unused)
C1P19	C1B30	Interrupt In (unused)
C1P20		Unused
C1P21	C1B31	Read/Write Out
C1P22	None	+5 Volts (thru pullup resistor)
C1P23	C1B03	A3 Sensor Address Bit
C1P24	C1B01	A1 Sensor Address Bit
C1P25	C1B25	D5 Sensor Data Bus
C1P26	C1B23	D3 Sensor Data Bus
C1P27	C1B21	D1 Sensor Data Bus
C1P28	C1B27	D7 Sensor Data Bus
C1P29	C1B26	D6 Sensor Data Bus
C1P30	C1B20	D0 Sensor Data Bus
C1P31	C1B22	D2 Sensor Data Bus
C1P32	C1B24	D4 Sensor Data Bus
C1P33	None	+5 Volts (thru pullup resistor)
C1P34	C1B00	A0 Sensor Address Bit
C1P35	C1B02	A2 Sensor Address Bit
C1P36	None	+5 Volts (thru pullup resistor)
C1P37	None	Digital Ground
C1P38	None	Digital Ground
C1P39	None	Digital Ground
C1P40	None	Digital Ground

C1E: GYRAC Computer Edge Connector

Pin Number	Signal Name	Description
C1E1	None	Digital Ground
C1E2	C1G01-	Tx Data Out (active low)
C1E3	None	Rx Data In (active low)
C1E4	C1G00-	Ready to Send
C1E5	None	+5 Volts DC Out
C1E6		Unused
C1E7		Unused
C1E8	None	+5 Volts DC Out
C1E9		Unused
C1E10	None	Power On Reset (active low)
C1E11		Unused
C1E12	None	+15 Volts DC In
C1E13		Unused
C1E14	None	-12 Volts DC Out
C1E15		Unused
C1E16	None	+12 Volts DC Out
C1E17		Unused
C1E18	None	-5 Volts DC Out
C1E19		Unused
C1E20		Unused
C1E21	None	-15 Volts DC In
C1E22		Unused
C1EA	None	Digital Ground
C1EB		Unused
C1EC		Unused
C1ED		Unused
C1EE		Unused
C1EF		Unused
C1EH	None	Unused
C1EJ		Unused
C1EK		Unused
C1EL		Unused
C1EM		Unused
C1EN		Unused
C1EP		Unused
C1ER		Unused
C1ES		Unused
C1ET		Unused
C1EU		Unused
C1EV		Unused
C1EW		Unused
C1EX	None	+5 Volts (thru pullup resistor)
C1EY		Unused
C1EZ	None	+5 Volts DC In

ION: Digital Interface Edge Connector

Pin Number	Signal Name	Description
ION1		Unused
ION2		Unused
ION3		Unused
ION4		Unused
ION5		Unused
ION6	GOR00	S/D D0 Data Bit
ION7	GOR01	S/D D1 Data Bit
ION8	GOR02	S/D D2 Data Bit
ION9	GOR03	S/D D3 Data Bit
ION10	GOR04	S/D D4 Data Bit
ION11	GOR05	S/D D5 Data Bit
ION12	GOR06	S/D D6 Data Bit
ION13	GOR07	S/D D7 Data Bit
ION14	GOR08	S/D D8 Data Bit
ION15	GOR09	S/D D9 Data Bit
ION16	GOR10	S/D D10 Data Bit
ION17	GOR11	S/D D11 Data Bit
ION18		Reserved
ION19		Reserved
ION20		Reserved
ION21		Reserved
ION22		Reserved
IONA	None	+5 Volts DC
IONB	None	Digital Ground
IONC	None	-15 Volts DC
IOND	None	Analog Ground
IONE	None	+15 Volts DC
IONF	C1B00	A0 Address Bit
IONH	C1B01	A1 Address Bit
IONJ	C1B02	A2 Address Bit
IONK	C1B03	A3 Address Bit
IONL	Bus	D0 Data Bit
IONM	Bus	D1 Data Bit
IONN	Bus	D2 Data Bit
IONP	Bus	D3 Data Bit
IONR	Bus	D4 Data Bit
IONS	Bus	D5 Data Bit
IONT	Bus	D6 Data Bit
IONU	Bus	D7 Data Bit
IONV	C1B31	Read/Write Control Line
IONW	GOS42	Angular Velocity Analog Signal
IONX	None	Acceleration Analog Signal
IONY	None	Acceleration Ground
IONZ		Unused

H89 to GYRAC RS-232 Cable

89 Connector Pin Number	Signal Description	GYRAC Connector Pin Number
2	Tx Data	2
3	Rx Data	3
7	Ground	7

H89 to Nav T or Drive Computer RS-232 Cable

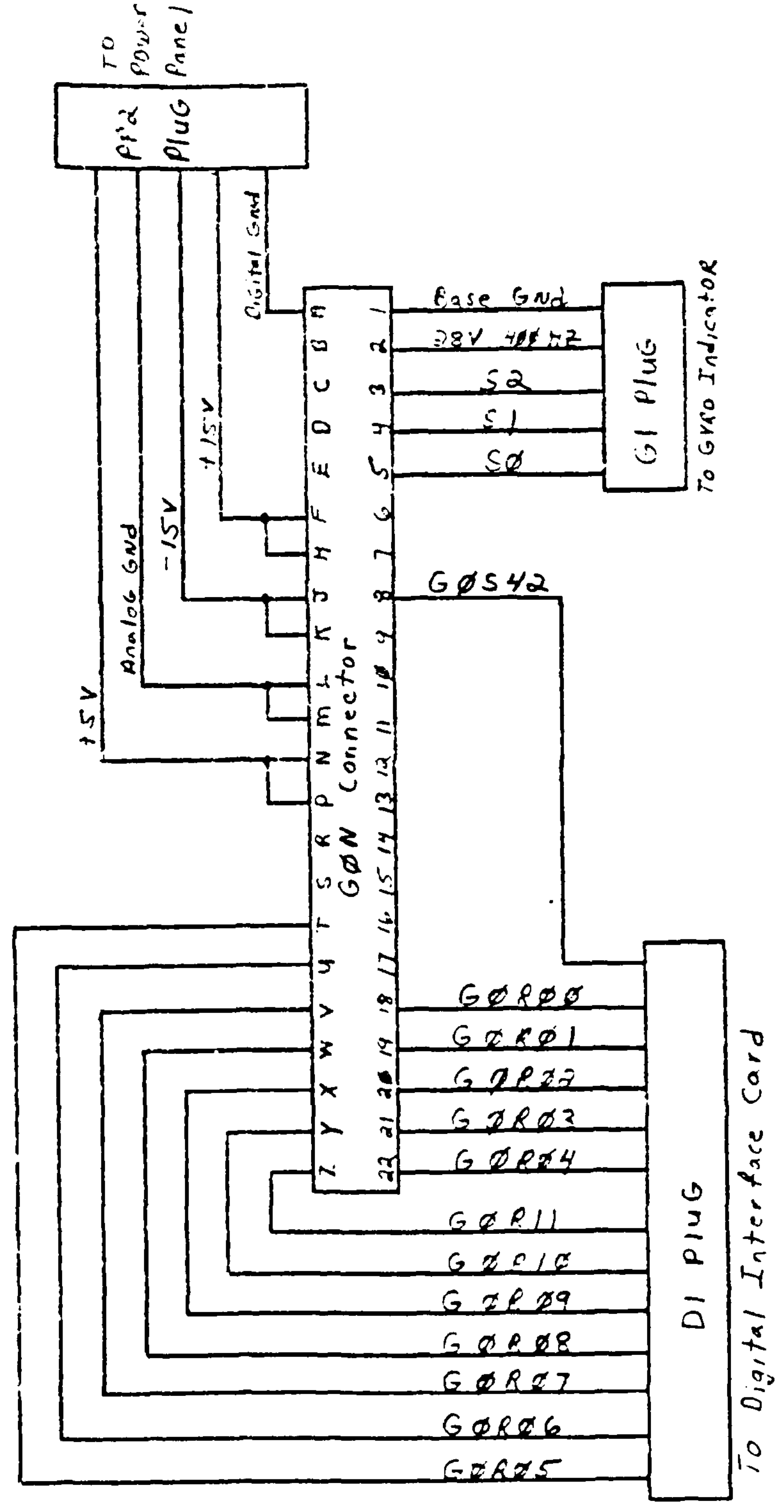
H89 Connector Pin Number	Signal Description	Nav T/Drive Computer Connector Pin Number
2	Tx Data	3
3	Rx Data	2
7	Ground	7

GYRAC to Nav L Computer RS-232 Cable

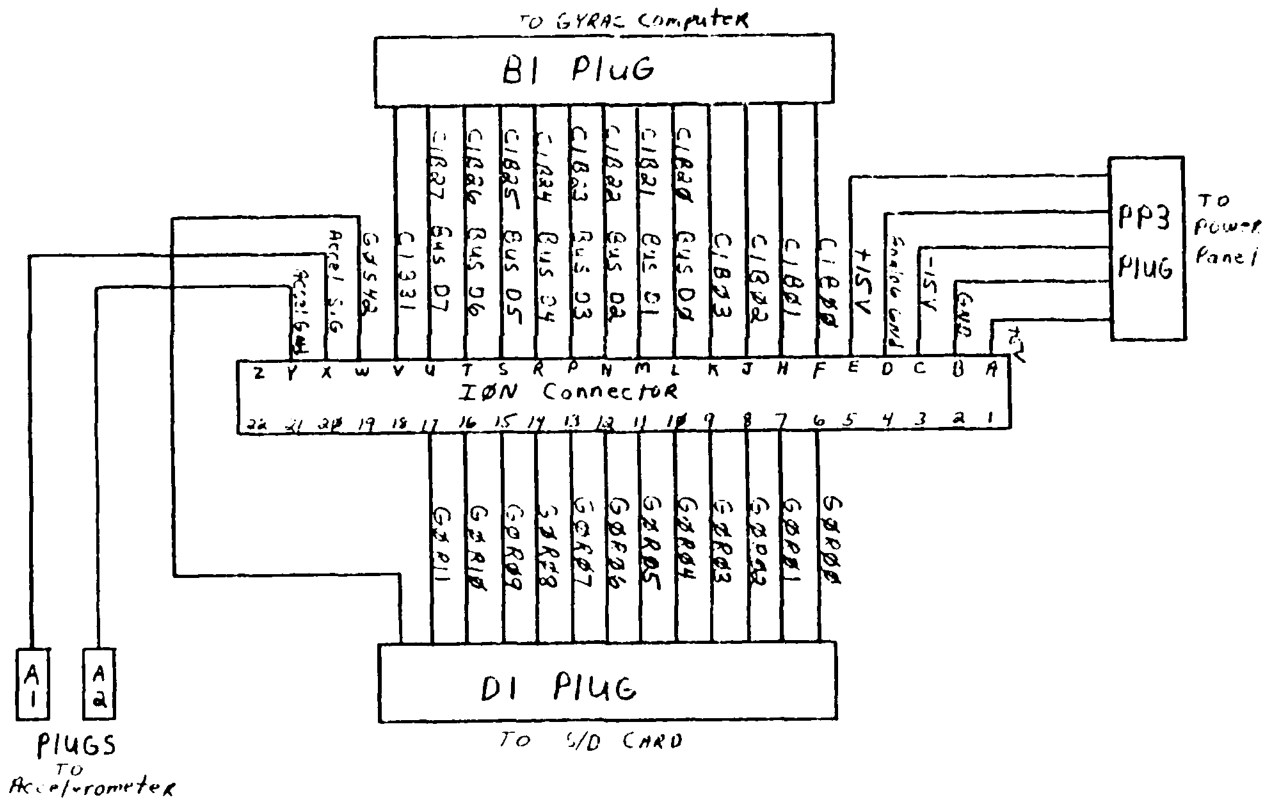
GYRAC Connector Pin Number	Signal Description	Nav L Connector Pin Number
2	Tx Data	3
3	Rx Data	2
7	Ground	7

Drive Computer to Nav X RS-232 Cable

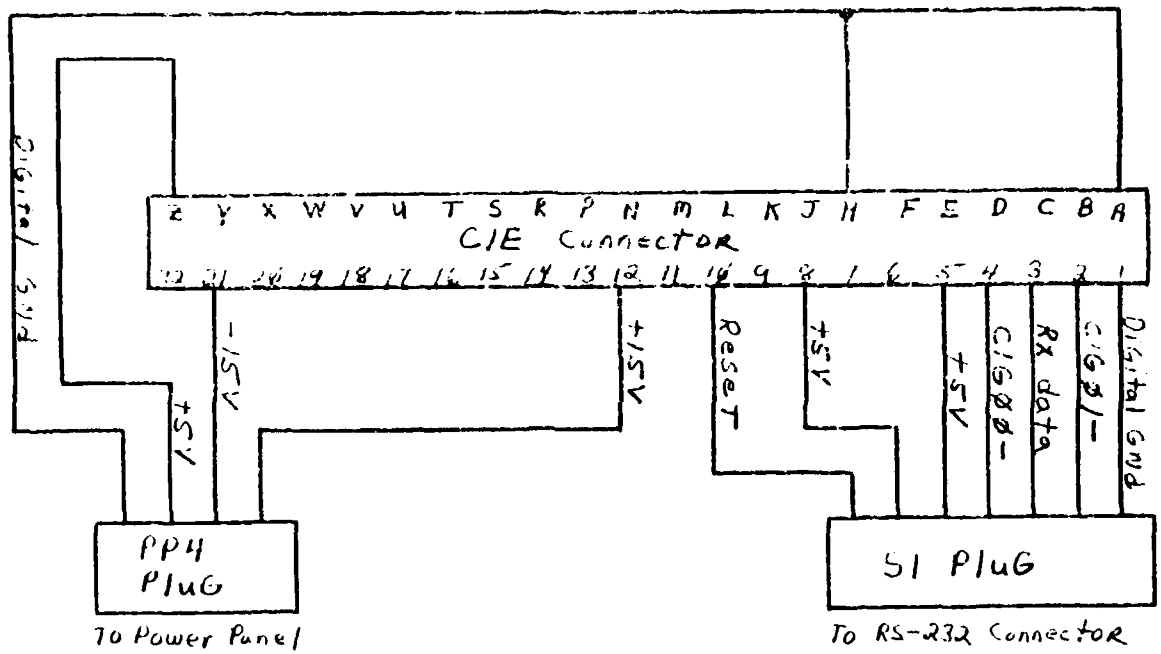
Drive Computer Connector Pin Number	Signal Description	Nav X Connector Pin Number
2	Tx Data	3
3	Rx Data	2
7	Ground	7



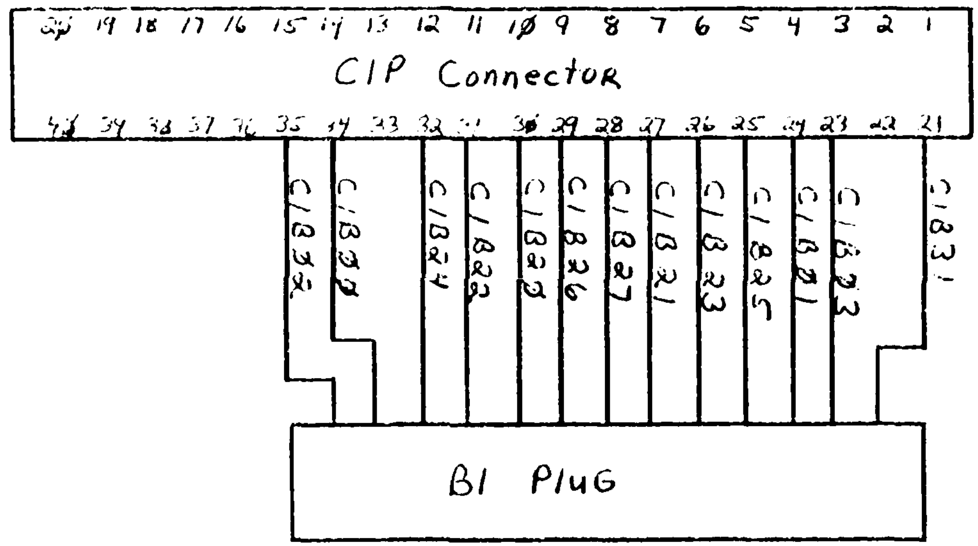
GYRAC WIRING HARNESS



GYRAC WIRING HARNESS (continued)

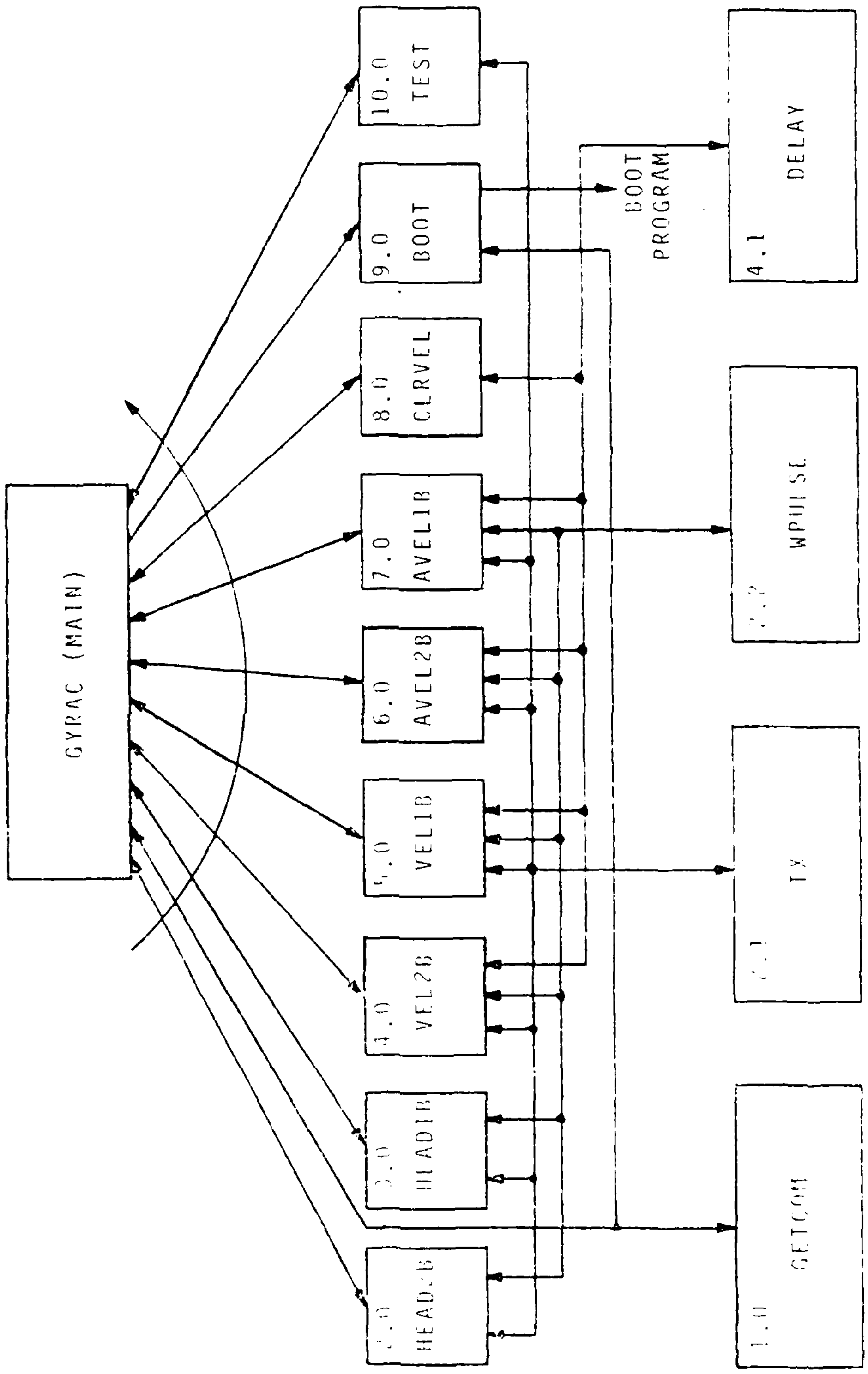


GYRAC WIRING HARNESS (continued)



GYRAC WIRING HARNESS (continued)

APPENDIX F



GYRAC.A STRUCTURE CHART

GYRAC.A PROGRAM LISTING

```
;*****************************************************************************
;
;       DATE:     [illegible]
;       [illegible]  1.1
;       TITLE:    [illegible] CONTROL PROGRAM
;       FILENAME:  [illegible]
;       [illegible]: CAPT WILLIAM [illegible] JR.
;       PROJECT:  [illegible] AND ACCELEROMETER BASED NAVIGATION SYSTEM FOR A
;                 MOBILE AUTONOMOUS ROBOT (THESIS)
;       OPERATING SYSTEM: [illegible] MAGNOLIA MICROSYSTEM [illegible]
;       LANGUAGE: [illegible] ASSEMBLER [illegible] VIRTUAL DEVICES 1984
;       USE: This program is assembled and then burned into an EPROM
;            for operation and control of the [illegible] subsystem.
;       CONTENTS:  main
;                  getin
;                  head2b
;                  head1b
;                  [illegible]
;                  vel1b
;                  avel2b
;                  avel1b
;                  clrvel
;                  boot
;                  test
;                  [illegible]
;                  pulse
;                  delay
;       FUNCTION:  This program provides an RS-232 test, a RAM boot
;                  loader, the capability to read 1 or 2 bytes of
;                  heading, angular velocity, and velocity and transmit
;                  the data over an RS-232 interface, and can clear the
;                  velocity [illegible] start in the accelerometer [illegible].
;*****************************************************************************

[illegible]   equ   [illegible]   ;Rom program start address
[illegible]   equ   41H   ;[illegible] command-- [illegible] 2 bytes
[illegible]   equ   42H   ;[illegible] command-- [illegible] 1 byte
[illegible]   equ   43H   ;[illegible] command-- heading 2 bytes
[illegible]   equ   44H   ;[illegible] command-- heading 1 byte
[illegible]   equ   45H   ;[illegible] command-- velocity 2 bytes
[illegible]   equ   [illegible]   ;[illegible] command-- velocity 1 byte
[illegible]   equ   47H   ;[illegible] command-- clear velocity
[illegible]
```

GYRAC.A PROGRAM LISTING (continued)

```
[illegible]         [illegible]     equ     [illegible]   ;Timer module LSB buffer & timer latch
[illegible]         stack           equ     [illegible]H  ;Initial stack pointer
[illegible]         read            equ     [illegible]H  ;PIA read port command
[illegible]4        write           equ     [illegible]   ;PIA write port command
[illegible]         [illegible]     equ     [illegible]H  ;CR 1 may be written, use enable clock,
                                                         ;normal CT mode, continuous operating
                                                         ;mode, output enabled
[illegible]         [illegible]     equ     [illegible]H  ;Timer divide count to 1 =(2*(count+1))
                                                         ;i.e. ACIA TX/RX clk to 147,[illegible] Hz
[illegible]         aciareset       equ     [illegible]H  ;Reset ACIA command
[illegible]         aciacmd         equu    15H           ;Divide by 16 mode, 8 bits data plus
                                                         ;1 stop bit, all interrupts disabled
[illegible]         tdre            equ     [illegible]H  ;ACIA transmit data register empty
[illegible]         rdrf            equ     [illegible]H  ;ACIA receive data register full
[illegible]         headaddl        equ     [illegible]H  ;Address of GYRAC heading low byte
[illegible]         headaddh        equ     [illegible]H  ;Address of GYRAC heading high byte
[illegible]         veladdl         equ     [illegible]H  ;Address of GYRAC velocity low byte
[illegible]A        veladdh         equ     [illegible]H  ;Address of GYRAC velocity high byte
[illegible]         avel addl       equ     [illegible]H  ;Address of GYRAC ang vel low byte
[illegible]         aveladdh        equ     [illegible]H  ;Address of GYRAC ang vel high byte
[illegible]         [illegible]sadd equ     [illegible]H  ;Address of GYRAC vel cons clear device
[illegible]         temp            equ     [illegible]H  ;Address of RAM scratch pad
[illegible]         rambot          equ     [illegible]H  ;Address of bottom of [illegible] ram block
[illegible]         blocks          equ     [illegible]H  ;Address of number of blocks of data
[illegible]         bytes           equ     [illegible]H  ;Address of number of bytes in a block
[illegible]         char            equ     [illegible]H  ;Address of current ASCII character
[illegible]4        charcnt         equ     [illegible]4H ;Address of current character count
[illegible]5        tdel            equ     [illegible]   ;Timing delay loop variable


E[illegible]                          org     begin     ;Beginning of main program
E[illegible]  0F              init    sei               ;Disable interrupts
E[illegible]1 8E [illegible]7 FF      lds     #stack    ;Initialize stack pointer to top of RAM

E[illegible]4 4F              pia     clra              ;Put zero in A reg
E[illegible]5 B7 [illegible]          staa    piactrlA  ;Clear PIA control register A
E[illegible]  B7 80 [illegible]       staa    piactrlB  ;Clear PIA control register B
E[illegible]  B7 80 [illegible]       staa    piadataB  ;Make PIA port B an input port
E[illegible]E 43                      coma
E[illegible]F B7 [illegible]          staa    piadataA  ;Make PIA port A an output port
E[illegible]2 [illegible]             ldab    #read
E[illegible]4 F7 [illegible]          stab    piactrlA  ;Prepare port A for data out & read enable
E[illegible]7 F7 [illegible]          stab    piactrlB  ;Prepare port B for data input

E[illegible]A [illegible]     ptm     ldaa    #[illegible]cmd   ;Get PTM timer 1 command
E[illegible]  B7 [illegible]          staa    [illegible]       ;Write PTM timer 1 command
E[illegible]F CE [illegible]          ldx     #[illegible]cnt   ;Get timer 1 count
E[illegible]  FF [illegible]4         stx     [illegible]       ;Set timer to divide by [illegible]
E[illegible]  4F                      clra
E[illegible]  B7 [illegible]          staa    [illegible]       ;Start timer

E[illegible]  [illegible]     acia    ldaa    #aciareset
E[illegible]  B7 [illegible]          staa    aciastat  ;Reset ACIA
E[illegible]  [illegible]             ldaa    #aciacmd
E[illegible]  [illegible]             staa    aciastat  ;Set [illegible] ACIA [illegible] data, [illegible]

[illegible]   [illegible]                                [illegible]
[illegible]   [illegible] FF [illegible]        [illegible]   ;[illegible] ACIA

E[illegible]  [illegible] A1          ldab    #[illegible]A     ;[illegible] B register
[illegible]   11                      cba               ;[illegible] A [illegible]
[illegible]   27 [illegible]          [illegible]       ;If [illegible]
```

```
[illegible]   [illegible] 42        ldab   #[illegible]   ;Put [illegible] in B register
E[illegible]  11                    cba                   ;Is command [illegible] in A register?
E[illegible]  27 4F                 beq    cmd[illegible] ;If yes go execute command [illegible]

E[illegible]  C[illegible] 43       ldab   #[illegible]   ;Put command [illegible] in B register
E[illegible]  11                    cba                   ;Is command [illegible] in A register?
E[illegible]  27 5[illegible]       beq    cmd[illegible] ;If yes go execute command [illegible]

E[illegible]  C[illegible] 44       ldab   #comD          ;Put command D in B register
E[illegible]  11                    cba                   ;Is command D in A register?
E[illegible]  27 57                 beq    cmdD           ;If yes go execute command D

E[illegible]  C[illegible] 45       ldab   #comE          ;Put command E in B register
E[illegible]  11                    cba                   ;Is command E in A register?
E[illegible]  27 5[illegible]       beq    cmdE           ;If yes go execute command E

E[illegible]  C[illegible] 46       ldab   #comF          ;Put command F in B register
E[illegible]  11                    cba                   ;Is command F in A register?
E[illegible]  27 5[illegible]       beq    cmdF           ;If yes go execute command F

E[illegible]  C[illegible] 47       ldab   #comG          ;Put command G in B register
E[illegible]  11                    cba                   ;Is command G in A register?
E[illegible]  27 5[illegible]       beq    cmdG           ;If yes go execute command G

E[illegible]  C[illegible] 48       ldab   #comH          ;Put command H in B register
E[illegible]  11                    cba                   ;Is command H in A register?
E[illegible]  27 5[illegible]       beq    cmdH           ;If yes go execute command H

E[illegible]  C[illegible] 49       ldab   #comI          ;Put command I in B register
E[illegible]  11                    cba                   ;Is command I in A register?
E[illegible]4 27 5[illegible]       beq    cmdI           ;If yes go execute command I

E[illegible]  C[illegible] 4A       ldab   #comJ          ;Put command J in B register
E[illegible]  11                    cba                   ;Is command J in A register?
E[illegible]  27 5[illegible]       beq    cmdJ           ;If yes go execute command J

E[illegible]  C[illegible] 4[illegible]  ldab   #comK     ;Put command K in B register
E[illegible]  11                    cba                   ;Is command K in A register?
E[illegible]E 27 [illegible]        beq    cmdK           ;If yes go execute command K

E[illegible]  C[illegible] 4C       ldab   #comL          ;Put command L in B register
E[illegible]  11                    cba                   ;Is command L in A register?
E[illegible]  27 [illegible]        beq    cmdL           ;If yes go execute command L

E[illegible]  C[illegible] 4D       ldab   #comM          ;Put command M in B register
E[illegible]  11                    cba                   ;Is command M in A register?
E[illegible]  27 [illegible]        beq    cmdM           ;If yes go execute command M

E[illegible]  C[illegible] 4E       ldab   #comN          ;Put command N in B register
E[illegible]  11                    cba                   ;Is command N in A register?
E[illegible]  27 [illegible]        beq    cmdN           ;If yes go execute command N

E[illegible]  C[illegible] 4F       ldab   #com[illegible]  ;Put command [illegible] in B register
E[illegible]  11                    cba                   ;Is command [illegible] in A register?
E[illegible]  27 [illegible]        beq    cmd[illegible] ;If yes go execute command [illegible]

[illegible]                         bra    [illegible]    ;[illegible]

[illegible]
[illegible]
[illegible]
[illegible]

[illegible]
[illegible]
```

GYRAC.A PROGRAM LISTING (continued)

```
E[illegible]   [illegible] E1 [illegible]            jsr    avel1b   ;Read and Tx 1 byte of ang vel
E[illegible]   7E E[illegible] [illegible]            jmp    input    ;Get next command

E[illegible]   BD E[illegible] FA cmdC        jsr    head2b   ;Read and Tx 2 bytes of heading
E[illegible]   7E E[illegible] [illegible]            jmp    input    ;Get next command

E[illegible]   BD E1 1[illegible] cmdD        jsr    head1b   ;Read and Tx 1 byte of heading
E[illegible]   7E E[illegible] [illegible]            jmp    input    ;Get next command

E[illegible]A  BD E1 41 cmdE        jsr    vel2b    ;Read and Tx 2 bytes of velocity
E[illegible]D  7E E[illegible] [illegible]            jmp    input    ;Get next command

E[illegible]   BD E1 74 cmdF        jsr    vel1b    ;Read and Tx 1 byte of velocity
E[illegible]   7E E[illegible] [illegible]            jmp    input    ;Get next command

E[illegible]   BD E1 [illegible] cmdG        jsr    clrvel   ;Clear velocity constant
E[illegible]   7E E[illegible] [illegible]            jmp    input    ;Get next command

E[illegible]   BD E1 [illegible] cmdH        jsr    avel2b   ;Read and Tx 2 bytes of ang vel
E[illegible]   7E E0 [illegible]            jmp    input    ;Get next command

E[illegible]   BD E1 B9 cmdI        jsr    avel1b   ;Read and Tx 1 byte of ang vel
E[illegible]5  7E E0 16            jmp    input    ;Get next command

E[illegible]   BD E0 FA cmdJ        jsr    head2b   ;Read and Tx 2 bytes of heading
E[illegible]   BD E1 41            jsr    vel2b    ;Read and Tx 2 bytes of velocity
E[illegible]   7E E0 [illegible]            jmp    input    ;Get next command

E[illegible]1  BD E1 19 cmdK        jsr    head1b   ;Read and Tx 1 byte of heading
E[illegible]4  BD E1 74            jsr    vel1b    ;Read and Tx 1 byte of velocity
E[illegible]   7E E0 [illegible]            jmp    input    ;Get next command

E[illegible]A  7E E0 [illegible] cmdL        jmp    init     ;Reset and re-initialize GYRAC

E[illegible]   7E E1 E[illegible] cmdM        jmp    boot     ;Load RAM memory from RS-232 & execute

E[illegible]   BD E[illegible] [illegible] cmdN        jsr    test     ;Tx printable char to RS-232
E[illegible]   7E E0 [illegible]            jmp    input    ;Get next command

E[illegible]   BD E1 1[illegible] cmdO        jsr    head1b   ;Read and Tx 1 byte of heading
E[illegible]   BD E1 41            jsr    vel2b    ;Read and Tx 2 bytes of velocity
E[illegible]   7E E0 [illegible]            jmp    input    ;Get next command
```

```
;**********************************************************************;
;                                                                      ;
;       DATE:   5/10/85                                                ;
;       VERSION:  1.0                                                  ;
;       NAME:  getcom                                                  ;
;       MODULE NUMBER:  1.6                                            ;
;       DESCRIPTION:  This module reads an ascii byte from the ACIA    ;
;                     and places it in the A register.  It will loop   ;
;                     indefinitely until a byte is read.               ;
;       PASSED VARIABLES:  None                                        ;
;       RETURNS:  Ascii command byte in the A register                 ;
;       GLOBAL VARIABLES USED:  None                                   ;
;       GLOBAL VARIABLES CHANGED:  None                                ;
;       REGISTERS USED:  A                                             ;
;       FILES READ:  None                                              ;
;       FILES WRITTEN:  None                                           ;
;       MODULES CALLED:  None                                          ;
;       CALLING MODULES:  main                                         ;
;       AUTHOR: CAPT WILLIAM J. RAME  JR.                              ;
;       HISTORY: NONE                                                  ;
;**********************************************************************;

E[illegible]EF   B6 A0 00 getcom  ldaa   aciastat ;Get ACIA (RS-232) status
E[illegible]F2   84 01            anda   #rdrf    ;Is receive data register full
E[illegible]F4   27 F9            beq    getcom   ;If no loop to getcom
E[illegible]F6   B6 A0 01         ldaa   aciadata ;Else read command byte from ACIA
E[illegible]F9   39               rts
```

```
;*****************************************************************;
;                                                                 ;
;       DATE: [illegible]                                         ;
;       VERSION: 1.1                                              ;
;       NAME: head2b                                              ;
;       MODULE NUMBER: [illegible]                                ;
;       DESCRIPTION: This module sends a write pulse to the gyro  ;
;                    heading data register to latch the data, reads ;
;                    two bytes of heading data, and sends the data, ;
;                    high byte first, to the ACIA (RS-232) output. ;
;                    Only the least significant 12 bits of the data is ;
;                    significant.                                 ;
;       PASSED VARIABLES: None                                    ;
;       RETURNS: Two bytes of heading data written to ACIA (RS-232) ;
;       GLOBAL VARIABLES USED: None                               ;
;       GLOBAL VARIABLES CHANGED: None                            ;
;       REGISTERS USED: A, B                                      ;
;       FILES READ: None                                          ;
;       FILES WRITTEN: None                                       ;
;       MODULES CALLED: [illegible]                               ;
;                       wpulse                                    ;
;       CALLING MODULES: main                                     ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                         ;
;       HISTORY: Version 1.0 original                             ;
;                Version 1.1 change heading data format           ;
;*****************************************************************;

E0FA    86 0F     head2b  ldaa  #headaddl ;Get address of LSB of heading
E0FC    B7 90 00          staa  piadataA  ;Write LSB heading address to GYRAC
E0FF    BD E2 23          jsr   wpulse    ;Send GYRAC write pulse to latch data
E102    86 0E             ldaa  #headaddh ;Get address of MSB of heading
E104    B7 90 00          staa  piadataA  ;Write MSB heading address to GYRAC
E107    F6 90 02          ldab  piadataB  ;Read MSB of heading from GYRAC
E10A    BD E2 13          jsr   [illegible] ;Send MSB of heading to ACIA
E10D    86 0F             ldaa  #headaddl ;Get address of LSB of heading
E10F    B7 90 00          staa  piadataA  ;Write LSB heading address to GYRAC
E112    F6 90 02          ldab  piadataB  ;Read LSB of heading from GYRAC
E115    BD E2 13          jsr   [illegible] ;Send LSB of heading to ACIA
E118    39                rts
```

```
;************************************************************************;
;                                                                        ;
;       DATE:  05/1[illegible]/85                                        ;
;       VERSION:  1.0                                                    ;
;       NAME:  tx                                                        ;
;       MODULE NUMBER:  2.1                                              ;
;       DESCRIPTION:  This module tests the status of the ACIA for an    ;
;                     empty transmit data register in an indefinite      ;
;                     loop until found and then the data in the B        ;
;                     register is sent to the ACIA (RS-232 output).      ;
;       PASSED VARIABLES:  Output data byte in the B register            ;
;       RETURNS:  Data byte output to ACIA                               ;
;       GLOBAL VARIABLES USED:  None                                     ;
;       GLOBAL VARIABLES CHANGED:  None                                  ;
;       REGISTERS USED:  A, B                                            ;
;       FILES READ:  None                                                ;
;       FILES WRITTEN:  None                                             ;
;       MODULES CALLED:  None                                            ;
;       CALLING MODULES:  read2t                                         ;
;                         read1b                                         ;
;                         vel2b                                          ;
;                         vel1b                                          ;
;                         ave12t                                         ;
;                         ave11t                                         ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                                ;
;       HISTORY: NONE                                                    ;
;************************************************************************;

E218   B6 A0 00 tx      ldaa  aciastat  ;Read ACIA (RS-232) status
E21B   84 02            anda  #tdre     ;Is ACIA transmit data register empty?
E21D   27 F9            beq   tx        ;If no, loop to tx and recheck status
E21F   F7 A0 01         stab  aciadata  ;Else transmit data in B reg to ACIA
E221   39               rts
```

```
;*****************************************************************************;
;                                                                             ;
;        DATE: [illegible]                                                    ;
;        VERSION: 1.0                                                         ;
;        NAME: wpulse                                                         ;
;        MODULE NUMBER: 2.2                                                   ;
;        DESCRIPTION: This module sends a write pulse to the GYRAC.           ;
;        PASSED VARIABLES: None                                               ;
;        RETURNS: None                                                        ;
;        GLOBAL VARIABLES USED: None                                          ;
;        GLOBAL VARIABLES CHANGED: None                                       ;
;        REGISTERS USED: B                                                    ;
;        FILES READ: None                                                     ;
;        FILES WRITTEN: None                                                  ;
;        MODULES CALLED: None                                                 ;
;        CALLING MODULES: head2b                                              ;
;                         head1b                                              ;
;                         vel2b                                               ;
;                         vel1b                                               ;
;                         avel2b                                              ;
;                         avel1b                                              ;
;        AUTHOR: CAPT WILLIAM J. RAMEY JR.                                    ;
;        HISTORY: NONE                                                        ;
;*****************************************************************************;

E123   C6 14     wpulse  ldab  #write     ;Get GYRAC write command
E125   F7 30 01          stab  piactrlA   ;Send GYRAC write level to latch data
E128   C6 [illegible]    ldab  #read      ;Get GYRAC read command
E12A   F7 30 01          stab  piactrlA   ;Send GYRAC read level to form a pulse
E12D   39                rts
```

```
;**************************************************************************;
;                                                                          ;
;       DATE:  05/26/85                                                    ;
;       VERSION:  1.1                                                      ;
;       NAME:  headlb                                                      ;
;       MODULE NUMBER:  2.0                                                ;
;       DESCRIPTION:  This module sends a write pulse to the gyro          ;
;                     heading data register to latch the data, reads      ;
;                     two bytes of heading data, adjusts the data to      ;
;                     form a byte of the 8 most significant bits, and     ;
;                     sends this byte of heading data to the ACIA         ;
;                     (RS-232) output.                                     ;
;       PASSED VARIABLES:  None                                            ;
;       RETURNS:  One byte of heading data written to ACIA (RS-232)        ;
;       GLOBAL VARIABLES USED:  temp                                       ;
;       GLOBAL VARIABLES CHANGED:  temp                                    ;
;       REGISTERS USED:  A, B                                              ;
;       FILES READ:  None                                                  ;
;       FILES WRITTEN:  None                                               ;
;       MODULES CALLED:  tx                                                ;
;                        wpulse                                            ;
;       CALLING MODULES:  main                                             ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                                  ;
;       HISTORY: Version 1.0 original                                      ;
;                Version 1.1 change heading data format                    ;
;**************************************************************************;

E119    86 0F     headlb  ldaa  #headaddl ;Get address of LSB of heading
E11B    B7 80 00          staa  piadataA  ;Write address of LSB of head to GYRAC
E11E    BD E1 23          jsr   wpulse    ;Send GYRAC write pulse to latch data
E121    86 0E             ldaa  #headaddh ;Get address of MSB of heading
E123    B7 80 00          staa  piadataA  ;Write address of MSB of head to GYRAC
E126    F6 80 02          ldab  piadataB  ;Read MSB of heading data
E129    58                aslb            ;Move lower 4 bits to upper 4 bits
E12A    58                aslb            ;and shift 4 zeroes into the lower
E12B    58                aslb            ;4 bits--i.e. xxxx0000  This will
E12C    58                aslb            ;form the MS nibble of the heading byte
E12D    86 0F             ldaa  #headaddl ;Get address of LSB of heading
E12F    B7 80 00          staa  piadataA  ;Write address of LSB of head to GYRAC
E132    B6 80 02          ldaa  piadataB  ;Read LSB of heading data
E135    44                lsra            ;Move upper 4 bits to lower 4 bits
E136    44                lsra            ;and shift 4 zeroes into the upper
E137    44                lsra            ;4 bits--i.e. 0000xxxx  This will
E138    44                lsra            ;form the LS nibble of the heading byte
E139    97 00             staa  temp      ;Save the LS nibble of the heading
E13B    DA 00             orab  temp      ;Form 8 bits of heading (1 byte)
E13D    BD E1 13          jsr   tx        ;Output 1 byte of heading to ACIA
E140    39                rts
```

```
;*****************************************************************;
;                                                                 ;
;       DATE: [illegible]                                         ;
;       VERSION: 1.1                                              ;
;       NAME: vel16                                               ;
;       MODULE NUMBER: 4.0                                        ;
;       DESCRIPTION: This module sends a write pulse to the gyro  ;
;                    velocity data register to latch the data, reads ;
;                    two bytes of velocity data, reformats and sends ;
;                    the data, MSB first, to the ACIA (RS-232) output. ;
;                    Only the least significant 10 bits of the data is ;
;                    significant.                                 ;
;       PASSED VARIABLES: None                                    ;
;       RETURNS: Two bytes of velocity data written to ACIA (RS-232) ;
;       GLOBAL VARIABLES USED: None                               ;
;       GLOBAL VARIABLES CHANGED: None                            ;
;       REGISTERS USED: A, B                                      ;
;       FILES READ: None                                          ;
;       FILES WRITTEN: None                                       ;
;       MODULES CALLED: tx                                        ;
;                       wpulse                                    ;
;                       delay                                     ;
;       CALLING MODULES: main                                     ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                         ;
;       HISTORY: Version 1.0 original                             ;
;                Version 1.1 change velocity data format          ;
;*****************************************************************;

E141    86 0A     vel16   ldaa  #veladdh  ;Get address of MSB of velocity
E143    B7 80 00          staa  piadataA  ;Write MSB velocity address to GYRAC
E146    BD E2 2[illegible]          jsr   wpulse    ;Send GYRAC write pulse to latch data
E149    BD E2 2E          jsr   delay     ;Allow for A to D conversion time
E14C    F6 80 02          ldab  piadataB  ;Read MSB of heading from GYRAC
E14F    D7 00             stab  temp      ;Save raw MSB of data
E151    54                lsrb            ;Form correct MSB
E152    54                lsrb
E153    54                lsrb
E154    54                lsrb
E155    54                lsrb
E156    54                lsrb
E157    BD E2 13          jsr   tx        ;Send MSB of heading to ACIA
E15A    86 0B             ldaa  #veladdl  ;Get address of LSB of velocity
E15C    B7 80 00          staa  piadataA  ;Write LSB velocity address to GYRAC
E15F    F6 80 02          ldab  piadataB  ;Read LSB of velocity from GYRAC
E162    54                lsrb            ;Adjust raw LSB to correct format
E163    54                lsrb
E164    54                lsrb
E165    54                lsrb
E166    54                lsrb
E167    54                lsrb
E168    96 00             ldaa  temp      ;Get raw MSB of data
E16A    48                asla            ;Adjust raw MSB to correct format
E16B    48                asla
E16C    97 00             staa  temp
E16E    DA 00             orab  temp      ;Form correct MSB
E170    BD E2 13          jsr   tx        ;Send LSB of velocity to ACIA
E173    39                rts
```

```
;*****************************************************************;
;                                                                 ;
;       DATE:   [illegible]                                       ;
;       VERSION:  1.[illegible]                                   ;
;       NAME:  delay                                              ;
;       MODULE NUMBER:  4.1                                       ;
;       DESCRIPTION:  This module provides a time delay of [illegible] ;
;                   [illegible] microseconds.                     ;
;       PASSED VARIABLES:  tdel                                   ;
;       RETURNS:  None                                            ;
;       GLOBAL VARIABLES USED:  None                              ;
;       GLOBAL VARIABLES CHANGED:  None                           ;
;       REGISTERS USED:  A                                        ;
;       FILES READ:  None                                         ;
;       FILES WRITTEN:  None                                      ;
;       MODULES CALLED:  None                                     ;
;       CALLING MODULES:  [illegible]                             ;
;                         [illegible]                             ;
;                         [illegible]                             ;
;                         [illegible]                             ;
;                         [illegible]                             ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                         ;
;       HISTORY: NONE                                             ;
;*****************************************************************;


E[illegible]E  [illegible]      delay  ldaa  #tdel      ;Get time delay variable
E[illegible]   [illegible]      next   nop
E[illegible]   4A                      deca
E[illegible]   26 FC                   bne   next       ;If not done, do another loop delay
E[illegible]4  [illegible]             rts


[illegible]                            org   vector
[illegible]    E[illegible] E[illegible]   fdb   begin, begin, begin, begin   ;Set up vector table
[illegible]    E[illegible] E[illegible]


E4[illegible]                          END
```

```
;****************************************************************;
;                                                                ;
;       DATE:  01/26/83                                          ;
;       VERSION:  1.1                                            ;
;       NAME:  vel1b                                             ;
;       MODULE NUMBER:  5.0                                      ;
;       DESCRIPTION:  This module sends a write pulse to the gyro ;
;                     velocity data register to latch the data, reads ;
;                     the MSB of heading data, and sends this byte of ;
;                     velocity data to the ACIA (RS-232) output.  ;
;      PASSED VARIABLES:  None                                   ;
;       RETURNS:  One byte of velocity data written to ACIA (RS-232) ;
;       GLOBAL VARIABLES USED:  None                             ;
;       GLOBAL VARIABLES CHANGED:  None                          ;
;       REGISTERS USED:  A, B                                    ;
;       FILES READ:  None                                        ;
;       FILES WRITTEN:  None                                     ;
;       MODULES CALLED:  tx                                      ;
;                        wpulse                                  ;
;                        delay                                   ;
;       CALLING MODULES:  main                                   ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                        ;
;       HISTORY: Version 1.0 original                            ;
;                Version 1.1 change velocity data format         ;
;****************************************************************;

E174   86 0A     vel1b   ldaa  #veladdh  ;Get address of MSB of velocity
E176   B7 80 00          staa  piadataA  ;Write address of MSB of vel to GYRAC
E179   BD E2 21          jsr   wpulse    ;Send GYRAC write pulse to latch data
E17C   BD E2 1E          jsr   delay     ;Allow for A to D conversion time
E17F   F6 80 02          ldab  piadataB  ;Read MSB of velocity data
E182   BD E2 13          jsr   tx        ;Output 1 byte of velocity to ACIA
E185   39                rts
```

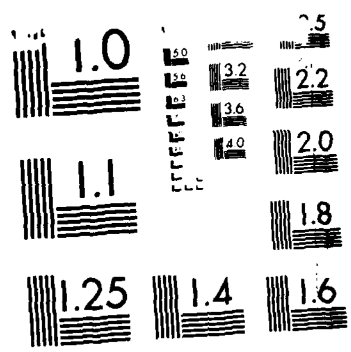
```
;*************************************************************************;
;                                                                         ;
;       DATE: ?/21/85                                                     ;
;       VERSION: 1.1                                                      ;
;       NAME: avelOt                                                      ;
;       MODULE NUMBER: 1.0                                                ;
;       DESCRIPTION:  This module sends a write pulse to the gyro         ;
;                     angular velocity data register to latch the data,   ;
;                     reads two bytes of ang vel data, reformats and      ;
;                     sends the data, MSB first, to the ACIA (RS-232)     ;
;                     output.  Only the least significant 10 bits of      ;
;                     the data is significant.                            ;
;       PASSED VARIABLES:  None                                           ;
;       RETURNS:  Two bytes of angular velocity data written to ACIA      ;
;       GLOBAL VARIABLES USED:  None                                      ;
;       GLOBAL VARIABLES CHANGED:  None                                   ;
;       REGISTERS USED:  A, B                                             ;
;       FILES READ:  None                                                 ;
;       FILES WRITTEN:  None                                              ;
;       MODULES CALLED:  tx                                               ;
;                        wpulse                                           ;
;                        delay                                            ;
;       CALLING MODULES:  main                                            ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                                 ;
;       HISTORY: Version 1.0 original                                     ;
;                version 1.1 change angular velocity data format          ;
;*************************************************************************;

E186    86 0C      avelOt   ldaa   #aveladdh ;Get address of MSB of ang vel
E188    B7 10 00            staa   piadataA  ;Write MSB ang vel address to GYRAC
E18B    BD E2 ??            jsr    wpulse    ;Send GYRAC write pulse to latch data
E18E    BD E2 ??            jsr    delay     ;Allow for A to D conversion time
E191    F6 10 02            ldab   piadataB  ;Read MSB of ang vel from GYRAC
E194    D7 00               stab   temp      ;Save raw MSB of data
E196    54                  lsrb             ;Form correct MSB of data
E197    54                  lsrb
E198    54                  lsrb
E199    54                  lsrb
E19A    54                  lsrb
E19B    54                  lsrb
E19C    BD E2 ??            jsr    tx        ;Send MSB of ang vel to ACIA
E19F    86 0D               ldaa   #aveladdl ;Get address of LSB of ang vel
E1A1    B7 10 00            staa   piadataA  ;Write LSB ang vel address to GYRAC
E1A4    F6 10 02            ldab   piadataB  ;Read LSB of ang vel from GYRAC
E1A7    54                  lsrb             ;[illegible]
E1A8    54                  lsrb
E1A9    54                  lsrb
E1AA    54                  lsrb
E1AB    54                  lsrb
E1AC    54                  lsrb
E1AD    96 00               ldaa   temp      ;Get raw MSB of data
E1AF    48                  asla             ;Adjust raw MSB to correct LSB format
E1B0    48                  asla
E1B1    [illegible]         staa   temp
E1B3    DA [illegible]      [illegible] temp ;[illegible] LSB of data
E1B5    BD E2 ??            jsr    tx        ;Send LSB of ang vel to ACIA
E1B8    39                  rts
```

```
;**********************************************************************;
;                                                                      ;
;       DATE:    [illegible]                                           ;
;       VERSION:  1.1                                                  ;
;       NAME:  avellb                                                  ;
;       MODULE NUMBER:  7.[illegible]                                  ;
;       DESCRIPTION:  This module sends a write pulse to the gyro      ;
;                     angular velocity data register to latch the data, ;
;                     reads the MSB of ang vel data, and sends this    ;
;                     byte of ang vel data to the ACIA (RS-232) output. ;
;       PASSED VARIABLES:  None                                        ;
;       RETURNS:  One byte of ang vel data written to ACIA (RS-232)    ;
;       GLOBAL VARIABLES USED:  None                                   ;
;       GLOBAL VARIABLES CHANGED:  None                                ;
;       REGISTERS USED:  A, B                                          ;
;       FILES READ:  None                                              ;
;       FILES WRITTEN:  None                                           ;
;       MODULES CALLED:  tx                                            ;
;                        wpulse                                        ;
;                        delay                                         ;
;       CALLING MODULES:  main                                         ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                              ;
;       HISTORY: Version 1.0 original                                  ;
;                Version 1.1 change angular velocity data format       ;
;**********************************************************************;

E1[illegible]  [illegible]     avellb  ldaa  #aveladdr ;Get address of MSB of ang vel
E1[illegible]  B7 [illegible]          staa  piadataA  ;Write address of MSB of avel to GYRAC
E1[illegible]  BD [illegible]          jsr   wpulse    ;Send GYRAC write pulse to latch data
E1[illegible]  BD E1 [illegible]       jsr   delay     ;Allow for A to D conversion time
E1[illegible]  F[illegible]            ldab  piadataB  ;Read MSB of angular velocity data
E1[illegible]  BD E1 [illegible]       jsr   tx        ;Output 1 byte of ang vel to ACIA
E1[illegible]  39                      rts
```

```
;*********************************************************************;
;                                                                     ;
;       DATE:  05/26/85                                               ;
;       VERSION:  1.1                                                 ;
;       NAME:  clrvel                                                 ;
;       MODULE NUMBER:  3.0                                           ;
;       DESCRIPTION:  This module clears (sets = 0) the integration   ;
;                     constant in the acceleration integrator.  This is ;
;                     equivalent to zero velocity initial condition.  ;
;                     PLACING THE ADDRESS on the GYRAC address bus    ;
;                     and sending a write pulse, clears the velocity  ;
;                     constant.  The address and write level must be  ;
;                     REMOVED to stop clearing the constant.          ;
;       PASSED VARIABLES:  None                                       ;
;       RETURNS:  None                                                ;
;       GLOBAL VARIABLES USED:  None                                  ;
;       GLOBAL VARIABLES CHANGED:  None                               ;
;       REGISTERS USED:  A, B                                         ;
;       FILES READ:  None                                             ;
;       FILES WRITTEN:  None                                          ;
;       MODULES CALLED:  delay                                        ;
;       CALLING MODULES:  main                                        ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                             ;
;       HISTORY: Version 1.0 original                                 ;
;                   Version 1.1 add delay subroutine                  ;
;*********************************************************************;

E1CB   86 09      clrvel  ldaa  #vconsadd ;Get address of vel const clear device
E1CD   B7 90 00           staa  piadataA  ;Start clearing velocity constant
E1D0   C6 04              ldab  #write    ;Get write command
E1D2   F7 90 01           stab  piactrlA  ;Send GYRAC write level to start clear
E1D5   BD E2 2E           jsr   delay     ;Allow time for capacitor to discharge
E1D8   C6 [illegible]     ldab  #read     ;Get read command
E1DA   F7 90 01           stab  piactrlA  ;Send GYRAC read level to stop clear
E1DD   4F                 clra
E1DE   B7 90 00           staa  piadataA  ;Stop clearing velocity constant
E1E1   39                 rts
```

AD-A164 036   GYRO AND ACCELEROMETER BASED NAVIGATION SYSTEM FOR A MOBILE AUTONOMOUS RO...(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI...   3/3

UNCLASSIFIED   R J BLOOM ET AL. 02 DEC 85   F/G 17/7   NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS

GYRAC.A PROGRAM LISTING (continued)

```
;*****************************************************************;
;                                                                 ;
;       DATE:  05/10/85                                           ;
;       VERSION:  1.0                                             ;
;       NAME:  boot                                               ;
;       MODULE NUMBER:  9.0                                       ;
;       DESCRIPTION:  This module will read a program from the ACIA ;
;                     (RS-232) serial port, load it into RAM memory, ;
;                     and then execute the program just loaded.  This ;
;                     program does an unconditional jump to the loaded ;
;                     program.                                    ;
;       PASSED VARIABLES:  None                                   ;
;       RETURNS:  None                                            ;
;       GLOBAL VARIABLES USED:  None                              ;
;       GLOBAL VARIABLES CHANGED:  None                           ;
;       REGISTERS USED:  A, B                                     ;
;       FILES READ:  None                                         ;
;       FILES WRITTEN:  None                                      ;
;       MODULES CALLED:  getcom                                   ;
;       CALLING MODULES:  main                                    ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                         ;
;       HISTORY: NONE                                             ;
;*****************************************************************;

E1E2   BD E0 EF boot    jsr   getcom    ;Get the number of blocks to load
E1E5   97 01            staa  blocks    ;Save # of blocks in memory
E1E7   BD E0 EF         jsr   getcom    ;Get the number of bytes in a block
E1EA   97 02            staa  bytes     ;Save # of bytes in a block in memory
E1EC   FE 60 00         ldx   rambot    ;Load pointer to RAM memory
E1EF   D6 02    blkloop ldab  bytes     ;Get number of bytes
E1F1   BD E0 EF bytloop jsr   getcom    ;Get byte of data
E1F4   A7 00            staa  0,x       ;Load data byte into RAM
E1F6   08               inx             ;Increment to next RAM pointer
E1F7   5A               decb            ;Decrement block byte count
E1F8   26 F7            bne   bytloop   ;If byte count not 0, get next byte
E1FA   7A 00 01         dec   blocks    ;Else block done, decrement blk count
E1FD   26 F0            bne   blkloop   ;If all block are not done do another
E1FF   7E 60 00         jmp   rambot    ;Else go execute loaded program
```

```
;************************************************************************;
;                                                                        ;
;        DATE:  05/10/85                                                 ;
;        VERSION:  1.0                                                   ;
;        NAME:  test                                                     ;
;        MODULE NUMBER:  10.0                                            ;
;        DESCRIPTION:  This module will on command send to the ACIA      ;
;                      (RS-232) output 93 printable ASCII characters.    ;
;                      It will then return control to the calling        ;
;                      program.                                          ;
;        PASSED VARIABLES:  None                                         ;
;        RETURNS:  93 characters to ACIA                                 ;
;        GLOBAL VARIABLES USED:  None                                    ;
;        GLOBAL VARIABLES CHANGED:  None                                 ;
;        REGISTERS USED:  A, B                                           ;
;        FILES READ:  None                                               ;
;        FILES WRITTEN:  None                                            ;
;        MODULES CALLED:  tx                                             ;
;        CALLING MODULES:  main                                          ;
;        AUTHOR: CAPT WILLIAM J. RAMEY JR.                               ;
;        HISTORY: NONE                                                   ;
;************************************************************************;

E202   C6 21      test      ldab   #21H      ;Get first ASCII character
E204   D7 03                stab   char      ;Save character in memory
E206   86 5D                ldaa   #93       ;Get number of characters to be sent
E208   97 04                staa   charcnt   ;Save character count
E20A   D6 03      charloop  ldab   char      ;Get next ASCII character
E20C   BD E2 18             jsr    tx        ;Send ASCII character
E20F   7C 00 03             inc    char      ;Increment to next ASCII character
E212   7A 00 04             dec    charcnt   ;Decrement character count
E215   26 F3                bne    charloop  ;If all characters have not been sent
                                             ;go send the next one
E217   39                   rts              ;Else return to calling program
```

acia	E0[illegible]	aciacmd	[illegible]	aciadata	A0[illegible]1	aciarese	[illegible]
aciastat	A[illegible]	avel1b	E1[illegible]	avel2b	E1[illegible]	aveladdh	00[illegible]C
aveladdl	[illegible]D	begin	E0[illegible]	bitloop	E1EF	blocks	[illegible]1
bout	E1E1	bytes	[illegible]	butloop	E1F1	char	0003
charcnt	[illegible]4	charloop	E2[illegible]	clrvel	E1CC	cmdA	E[illegible]
cmdB	E[illegible]	cmdC	E1[illegible]E	cmdD	E144	cmdE	E04A
cmdF	E1C[illegible]	cmdG	E1D[illegible]	cmdH	E0DC	cmdI	E0C2
cmdJ	E0C[illegible]	cmdK	E0C1	cmdL	E0E4	cmdM	E0DD
cmdN	E1E[illegible]	cmdO	E0E[illegible]	comA	0041	comB	0042
comC	0043	comD	0044	comE	0045	comF	0046
comG	0047	comH	0048	comI	0049	comJ	004A
comK	004B	comL	004C	comM	004D	comN	004E
comO	004F	delay	E22E	getcom	E0EF	head1b	E119
head2b	E0FA	headaddh	000E	headaddl	000F	init	E000
input	E0[illegible]	next	E2[illegible]	pia	E004	piactr1A	8001
piactr1B	8003	piadataA	8000	piadataB	2002	ptm	E01A
ptmsc	[illegible]4	ptmcmd	0033	ptmcnt	0032	ptmctrl1	C000
ptmctrl2	C001	ptmlt	C005	ranbot	6000	rdrf	0001
read	003C	stack	67FF	tdel	0005	tdre	0002
temp	0000	test	E2[illegible]2	tx	E218	vectsadd	000[illegible]
vector	E1F[illegible]	vel1b	E174	vel2b	E141	veladdh	000A
veladdl	000B	vpulse	E223	write	0034		

GYRAC.A OPERATING INSTRUCTIONS

STEP 1: Power up the H89 computer system. Place the System disk in Drive A and the gyro program disk in Drive B. Boot the system (type "B 29") and change the mode of Drive B to single sided double density (type "mode B:ss,dd"). Change the working drive to Drive B (type "B:").

STEP 2: Connect the H89 to GYRAC RS-232 cable between the H89 DCE connector and the MARRS-1 GYRAC Computer connector. Connect the external power cable to the GYRAC and turn on the power supplies. Flip the GYRAC power switch to the on position and press the GYRAC computer reset button. Flip the gyro control switch to the slaved mode for automatic tracking or the free mode for manual heading changes.

STEP 3: Load and run the M72 modem program on the H89 by typing "M72". When this program is running type "T" to enter the M72 Terminal mode. Set the H89 keyboard caps lock on.

STEP 4: Commands may now be issued to the GYRAC. They consist of single character capital letters from A thru O inclusive. The resulting hexidecimal data, as detailed below, is displayed (if it is printable) on the CRT. NOTE: Not all data returned is printable, so some results may not be displayed. Repeat STEP 4 as often as desired.

COMMAND	RESULT
A	Two bytes of heading data (12 valid bits) Two bytes of velocity data (10 valid bits) Two bytes of angular velocity data (10 valid bits)
B	One byte of heading data (8 valid bits) One byte of velocity data (8 valid bits) One byte of angular velocity data (8 valid bits)
C	Two bytes of heading data (12 valid bits)
D	One byte of heading data (8 valid bits)
E	Two bytes of velocity data (10 valid bits)
F	One byte of velocity data (8 valid bits)
G	Clear velocity constant (no data returned)

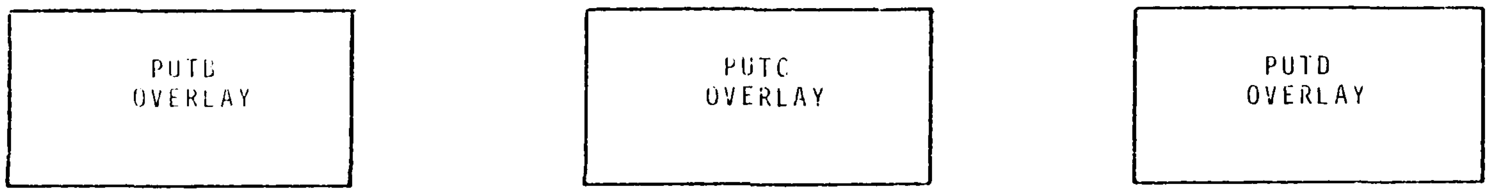
GYRAC.A OPERATING INSTRUCTIONS (continued)

COMMAND	RESULT
H	Two bytes of angular velocity data (10 valid bits)
I	One byte of angular velocity data (8 valid bits)
J	Two bytes of heading data (12 valid bits) Two bytes of velocity data (10 valid bits)
K	One byte of heading data (8 valid bits) One byte of velocity data (8 valid bits)
L	Soft reset of GYRAC computer (no data returned)
M	Boot load a program from the H89 into RAM and then execute to loaded program. No data is returned unless the user program sends it. Control is not automatically returned to the GYRAC control program, but is at the mercy of the boot program.
N	Returns 93 printable ASCII characters This tests the communications link
O	One byte of heading data (8 valid bits) Two bytes of velocity data (10 valid bits)

STEP 6: Shutdown all systems. Type "control shift " followed by "control E" to exit Terminal mode. To exit M72, type "CPM". Remove both disks from the drives and turn of the power to the H89 system. Turn off power to the GYRAC and external power supplies.

NOTE: All references to "control" and "shift" in the H89 command lines refer to the control and shift keys and not the words control and shift. It is assumed that the robot has been "pointed" to the desired initial heading before testing commences or that the gyro free mode will be used to vary the heading.

APPENDIX G



GTEST.A STRUCTURE CHART

GTEST.A TEST PROGRAM OVERLAY

```
;*****************************************************************************;
;                                                                             ;
;       DATE:    7-23-85                                                      ;
;       VERSION:  1.1                                                         ;
;       TITLE:  GYRAC DATA COLLECTION PROGRAM                                 ;
;       FILENAME:  GTEST.A                                                    ;
;       COORDINATOR:  CAPT WILLIAM J. RAME[illegible] F.                       ;
;       PROJECT:  GYRO AND ACCELEROMETER BASED NAVIGATION SYSTEM FOR A        ;
;                 MOBILE AUTONOMOUS ROBOT (THESIS)                            ;
;       OPERATING SYSTEM:  [illegible] CP/M V2.2A2 MAGNOLIA MICROSYSTEM 1[illegible];
;       LANGUAGE:  RC80-A ASSEMBLER V1.1 VIRTUAL DEVICES 1984                 ;
;       USE:  This program is assembled and then transferred to the          ;
;             robot's navigation computer via an RS-232 link.  It must       ;
;             be loaded into the nav computer using H72 in terminal (T)      ;
;             mode with the following command: L,01FA,03E9 followed by       ;
;             transmitting the file gtest.hex.  All of the original          ;
;             commands still work: C to begin transmitting data to HER,      ;
;             E to pause data Tx,  C to resume data Tx, and  C to            ;
;             warmboot the nav computer.  Each time the nav computer is      ;
;             rebooted (warm or cold) the gtest.hex file must be             ;
;             reloaded in order to send gyro data.                           ;
;       CONTENTS:  putb                                                       ;
;                  putc                                                       ;
;                  putd                                                       ;
;       FUNCTION:  This program overlays three existing subroutines in       ;
;                  the nav computer's RAM memory to allow the nav            ;
;                  computer to request data from the GYRAC (over an          ;
;                  RS-232 link) and then transmit the GYRAC's data           ;
;                  to an external computer over an RS-232 link.  It          ;
;                  replaces the subroutines that send sonar B, C, and        ;
;                  D data with subroutines that send velocity, heading,      ;
;                  and angular velocity respectively.  The data is           ;
;                  represented as four hexadecimal digits.                   ;
;*****************************************************************************;

01F4            beginb      equ   01F4H    ;Start address of the vel subr
0[illegible]4A  beginh      equ   [illegible]4AH  ;Start address of the head subr
[illegible]7A   begina      equ   01[illegible]AH ;Start address of the avel subr
0041            cmdA        equ   41H      ;GYRAC command-- tx all [illegible] bytes
0042            cmdB        equ   42H      ;GYRAC command-- tx all [illegible]
0043            cmdC        equ   43H      ;GYRAC command-- tx heading [illegible] bytes
0044            cmdD        equ   44H      ;GYRAC command-- tx heading [illegible] byte
0045            cmdE        equ   45H      ;GYRAC command-- [illegible]
0046            cmdF        equ   46H      ;GYRAC command-- [illegible]
0047            [illegible] equ   47H      ;GYRAC command-- [illegible]
0048            [illegible] equ   48H      ;GYRAC command-- tx ang vel [illegible] bytes
0049            [illegible] equ   49H      ;GYRAC command-- tx ang vel [illegible]
004A            [illegible] equ   4AH      ;GYRAC command-- tx head [illegible] bytes
004B            [illegible] equ   4BH      ;GYRAC command-- tx head [illegible]
004C            [illegible] equ   4CH      ;GYRAC command-- reset [illegible]
004D            [illegible] equ   4DH      ;GYRAC command-- [illegible] RAM
004E            [illegible] equ   4EH      ;GYRAC command-- RS-232 [illegible]
004F            [illegible] equ   4FH      ;GYRAC [illegible]
[illegible]     [illegible]stat  equ  [illegible] ;GYRAC [illegible] stat [illegible]
[illegible]     [illegible]     equ   [illegible] ;GYRAC [illegible] data [illegible]
[illegible]     [illegible]     equ   [illegible]H ;[illegible] data [illegible]
[illegible]     [illegible]           1H   ;[illegible] data [illegible]
[illegible]     [illegible]           [illegible] ;[illegible] data
[illegible]     [illegible]           [illegible] ;[illegible]
[illegible]     [illegible]     [illegible] ;[illegible] data
[illegible]     [illegible]           [illegible] ;[illegible]
[illegible]     [illegible]           [illegible] ;[illegible] data
```

GTEST.A TEST PROGRAM OVERLAY (continued)

```
;**************************************************************************;
;                                                                          ;
;       DATE: [illegible]                                                  ;
;       VERSION: 1.0                                                       ;
;       NAME: putb                                                         ;
;       MODULE NUMBER: 1.0                                                 ;
;       DESCRIPTION:  This module reads two bytes of velocity data from    ;
;                     the GYRAC, converts it to four hexadecimal           ;
;                     characters and places the characters in the output   ;
;                     buffer.                                              ;
;       PASSED VARIABLES:  None                                            ;
;       RETURNS:  Four hex velocity characters in the data buffer.         ;
;       GLOBAL VARIABLES USED:  None                                       ;
;       GLOBAL VARIABLES CHANGED:  None                                    ;
;       REGISTERS USED:  A, B, X                                           ;
;       FILES READ:  None                                                  ;
;       FILES WRITTEN:  None                                               ;
;       MODULES CALLED:  putChex                                           ;
;       CALLING MODULES:  xtend                                            ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                                  ;
;       HISTORY: NONE                                                      ;
;**************************************************************************;

02FA                    org  beginv      ;Start of putb subroutine
02FA   CE 05 4C putb    ldx  #intreading ;Point to velocity data buffer
02FD   C6 45            ldab #cmnE       ;Get GYRAC 2 byte vel command
02FF   B6 C0 10 txb     ldaa acia1stat   ;Get GYRAC ACIA status
0302   84 02            anda #tdre       ;Is transmit data reg empty?
0304   27 F9            beq  txb         ;If no loop to txb and recheck status
0306   F7 C0 11         stab acia1data   ;Else send GYRAC command

0309   B6 C0 10 rxbm    ldaa acia1stat   ;Get GYRAC ACIA status
030C   84 01            anda #rdrf       ;Is receive data register full?
030E   27 F9            beq  rxbm        ;If no loop to rxbm & recheck status
0310   B6 C0 11         ldaa acia1data   ;Else get the MSB of velocity data
0313   BD 02 5B         jsr  putChex     ;Convert data to hex & save in buffer

0316   B6 C0 10 rxbl    ldaa acia1stat   ;Get GYRAC ACIA status
0319   84 01            anda #rdrf       ;Is receive data register full?
031B   27 F9            beq  rxbl        ;If no, loop to rxbl & recheck status
031D   B6 C0 11         ldaa acia1data   ;Else get the LSB of velocity data
0320   BD 02 5B         jsr  putChex     ;Convert data to hex & save in buffer
0323   39               rts              ;Blank out remainder of old program
0324   01               nop
0325   01               nop
0326   01               nop
0327   01               nop
0328   01               nop
0329   01               nop
032A   01               nop
032B   01               nop
032C   01               nop

[illegible]
```

GTEST.A TEST PROGRAM OVERLAY (continued)

[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	1	[illegible]
[illegible]	01	[illegible]
[illegible]	[illegible]	[illegible]
034	01	nop
0341	01	nop
[illegible]	1	nop
[illegible]	1	nop
0344	01	nop
0345	01	nop
0346	01	nop
0347	01	nop
[illegible]	[illegible]	nop
[illegible]	[illegible]	nop

```
;*****************************************************************************;
;                                                                             ;
;       DATE:  [illegible]                                                    ;
;       VERSION:  1.1                                                         ;
;       NAME:  puth                                                           ;
;       MODULE NUMBER:  [illegible]                                           ;
;       DESCRIPTION:  This module reads two bytes of heading data from        ;
;                     the GYRAC, converts it to four hexadecimal              ;
;                     characters and places the characters in the output;
;                     buffer.  The most significant nibble is masked          ;
;                     out.                                                    ;
;       PASSED VARIABLES:  None                                               ;
;       RETURNS:  Four hex heading characters in the data buffer.             ;
;       GLOBAL VARIABLES USED:  None                                          ;
;       GLOBAL VARIABLES CHANGED:  None                                       ;
;       REGISTERS USED:  A, B, X                                              ;
;       FILES READ:  None                                                     ;
;       FILES WRITTEN:  None                                                  ;
;       MODULES CALLED:  put2hex                                              ;
;       CALLING MODULES:  [illegible]                                         ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                                     ;
;       HISTORY: Version 1.0 original program                                 ;
;                Version 1.1 add mask to MS nibble of the heading             ;
;*****************************************************************************;

024A   CE 45 53 puth    ldx  #increading ;Point to heading data buffer
024D   C6 43            ldab #com2       ;Get GYRAC 2 byte heading command
024F   B6 C0 10 trc     ldaa acialstat   ;Get GYRAC ACIA status
0252   84 02            anda #tdre       ;Is transmit data reg empty?
0254   27 F9            beq  trc         ;If no loop to trc and recheck status
0256   F7 C0 11         stab acialdata   ;Else send GYRAC command

0259   B6 C0 10 rcvh    ldaa acialstat   ;Get GYRAC ACIA status
025C   84 01            anda #rdrf       ;Is receive data register full?
025E   27 F9            beq  rcvh        ;If no loop to rcvh & recheck status
0260   B6 C0 11         ldaa acialdata   ;Else get the MSB of heading data
0263   84 0F            anda #hmask      ;Mask out most significant nibble
0265   BD 02 5B         jsr  put2hex     ;Convert data to hex & save in buffer

0268   B6 C0 10 rcvl    ldaa acialstat   ;Get GYRAC ACIA status
026B   84 01            anda #rdrf       ;Is receive data register full?
026D   27 F9            beq  rcvl        ;If no, loop to rcvl & recheck status
026F   B6 C0 11         ldaa acialdata   ;Else get the LSB of heading data
0272   BD 02 5B         jsr  put2hex     ;Convert data to hex & save in buffer
0275   39               rts
                                         ;Blank out remainder of program
0276   01               nop
0277   01               nop
0278   01               nop
0279   01               nop
027A   01               nop
027B   01               nop
027C   01               nop
[illegible]
```

GTEST.A TEST PROGRAM OVERLAY (continued)

```
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop

[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
[illegible]   01    nop
```

GTEST.A TEST PROGRAM OVERLAY (continued)

```
;**********************************************************************;
;                                                                      ;
;        DATE:  07 17 85                                               ;
;        VERSION:  1.0                                                 ;
;        NAME:  putd                                                   ;
;        MODULE NUMBER:  1.0                                           ;
;        DESCRIPTION:  This module reads two bytes of ang velocity data ;
;                      from the GYRAC, converts it to four hexadecimal  ;
;                      characters and places the characters in the output;
;                      buffer.                                         ;
;        PASSED VARIABLES:  None                                       ;
;        RETURNS:  Four hex ang velocity characters in the data buffer. ;
;        GLOBAL VARIABLES USED:  None                                  ;
;        GLOBAL VARIABLES CHANGED:  None                               ;
;        REGISTERS USED:  A, B, X                                      ;
;        FILES READ:  None                                             ;
;        FILES WRITTEN:  None                                          ;
;        MODULES CALLED:  put2hex                                      ;
;        CALLING MODULES:   send                                       ;
;        AUTHOR: CAPT WILLIAM J. RAMEY JR.                             ;
;        HISTORY: NONE                                                 ;
;**********************************************************************;

039A   CE 03 5A puts    ldx  #indreading ;Point to ang velocity data buffer
039D   C6 48            ldab #comH       ;Get GYRAC 2 byte ang vel command
039F   B6 C0 10 txd     ldaa acialstat   ;Get GYRAC ACIA status
03A2   84 02            anda #tdre       ;Is transmit data reg empty?
03A4   27 F9            beq  txd         ;If no loop to txd and recheck status
03A6   F7 C0 11         stab acialdata   ;Else send GYRAC command

03A9   B6 C0 10 rxdm    ldaa acialstat   ;Get GYRAC ACIA status
03AC   84 01            anda #rdrf       ;Is receive data register full?
03AE   27 F9            beq  rxdm        ;If no loop to rxdm & recheck status
03B0   B6 C0 11         ldaa acialdata   ;Else get the MSB of ang velocity data
03B3   BD 02 53         jsr  put2he      ;Convert data to hex & save in buffer

03B6   B6 C0 10 rxdl    ldaa acialstat   ;Get GYRAC ACIA status
03B9   84 01            anda #rdrf       ;Is receive data register full?
03BB   27 F9            beq  rxdl        ;If no, loop to rxdl & recheck status
03BD   B6 C0 11         ldaa acialdata   ;Else get the LSB of ang velocity data
03C0   BD 02 53         jsr  put2hex     ;Convert data to hex & save in buffer
03C3   39               rts              ;Blank out remainder of old program
03C4   01               nop
03C5   01               nop
03C6   01               nop
03C7   01               nop
03C8   01               nop
03C9   01               nop
03CA   01               nop
03CB   01               nop
03CC   01               nop

03CD   01               nop
03CE   01               nop
03CF   01               nop
03D0   01               nop
03D1   01               nop
03D2   01               nop
03D3   01               nop
03D4   01               nop
03D5   01               nop
03D6   01               nop

03D7   01               nop
```

GTEST.A TEST PROGRAM OVERLAY (continued)

```
03D8    01        nop
03D9    01        nop
03DA    01        nop
03DB    01        nop
03DC    01        nop
03DD    01        nop
03DE    01        nop
03DF    01        nop
03E0    01        nop

03E1    01        nop
03E2    01        nop
03E3    01        nop
03E4    01        nop
03E5    01        nop
03E6    01        nop
03E7    01        nop
03E8    01        nop
03E9    01        nop


03EA              END
```

GTEST.A TEST PROGRAM OVERLAY (continued)

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	conA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	@044	conE	[illegible]	conF	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	conI	[illegible]	[illegible]	@04A	[illegible]	[illegible]
comL	[illegible]	conM	[illegible]	[illegible]	[illegible]	[illegible]	@04F
mask	[illegible]	lnbreadi	[illegible]	lnbreadi	[illegible]	lnbreadi	[illegible]
out2hex	@25[illegible]	putb	[illegible]	putc	@14A	[illegible]	@37A
[illegible]	@001	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	@357	[illegible]	@3B[illegible]	[illegible]	@3A7	tdre	[illegible]
[illegible]	@2FF	txc	@34F	txd	[illegible]		

GTEST.A OPERATING INSTRUCTIONS

STEP 1: Power up the H89 computer system. Place the System disk in Drive A and the gyro program disk in Drive B. Boot the system (type "B 29") and change the mode of Drive B to single sided double density (type "mode B:ss,dd"). Change the working drive to Drive B (type "B:").

STEP 2: Power up the MARRS-1 Robot. Make sure the batteries are fully charged and the charger power line is connected and turned on. Press both the system reset key on the keypad and the Nav computer reset button.

STEP 3: Load and run the M72 modem program on the H89 by typing "M72". When this program is running type "SPD" to change the transmission time delays. When prompted for time delays reply with a "1" for both character and line delay times. Set the H89 keyboard caps lock on.

STEP 4: Connect the H89 to Robot RS-232 cable between the H89 DCE connector and the MARRS-1 Nav T connector. Connect the GYRAC to Nav Computer RS-232 cable between the GYRAC connector and the Nav L connector on MARRS-1. Connect the teaching pendant cable to MARRS-1. Connect the external power cable to he GYRAC and turn on the power supplies. Flip the GYRAC power switch to the on position and press the GYRAC computer reset button. Flip the gyro control switch to the slaved mode.

STEP 5: Load and transmit the GTEST.HEX file to the robot's navigation computer. This is done by typing "L,02FA,03E9" to load the file at Nav computer memory address 02FA (HEX). Next type "T filename". This will place the CRT in terminal mode and create an input buffer to store incoming data in disk file filename. Follow this by typing "control shift " then "control T" and "GTEST.HEX" to transmit thg program file to the Nav computer. Reply with Yes when asked for time delays. The program data will be displayed as it is transmitted. If an error is made in STEP 5, the navigation reset button must be pressed and the entire step done over.

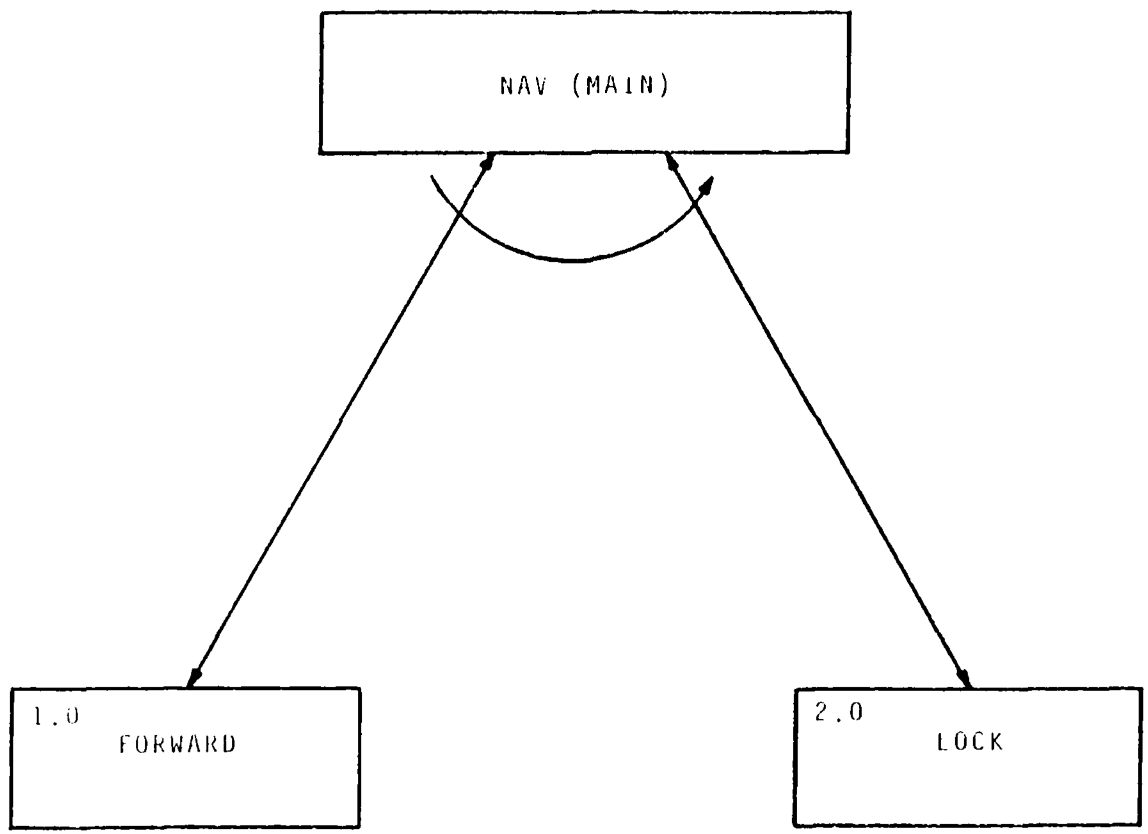
STEP 6: Begin program execution. First type "control shift " and then "control Y" to open the input data buffer. Now type "C" to begin program execution. Next steer the robot using the teaching pendant (MARRS-1 manual mode, 4) along the desired course. During program execution the robot will send optical shaft encoder data, heading data, angular velocity data, and velocity data at 0.1 second time intervals to the H89 computer. This data will be displayed on the CRT and stored in the input buffer.

STEP 7: When the robot run is completed (i.e. you have manually stopped it with the MARRS-1 system reset button) the data stored in the input buffer may be written to disk. To do this press the Nav Computer reset button on MARRS-1 and/or type "control C" on the H89. Next type "control shift " followed by "control E". Now type "WRT" to save the data to disk ("del" may also be typed to dump buffered data). If additional runs are required continue with STEP 5 and press all three MARRS-1 reset buttons.

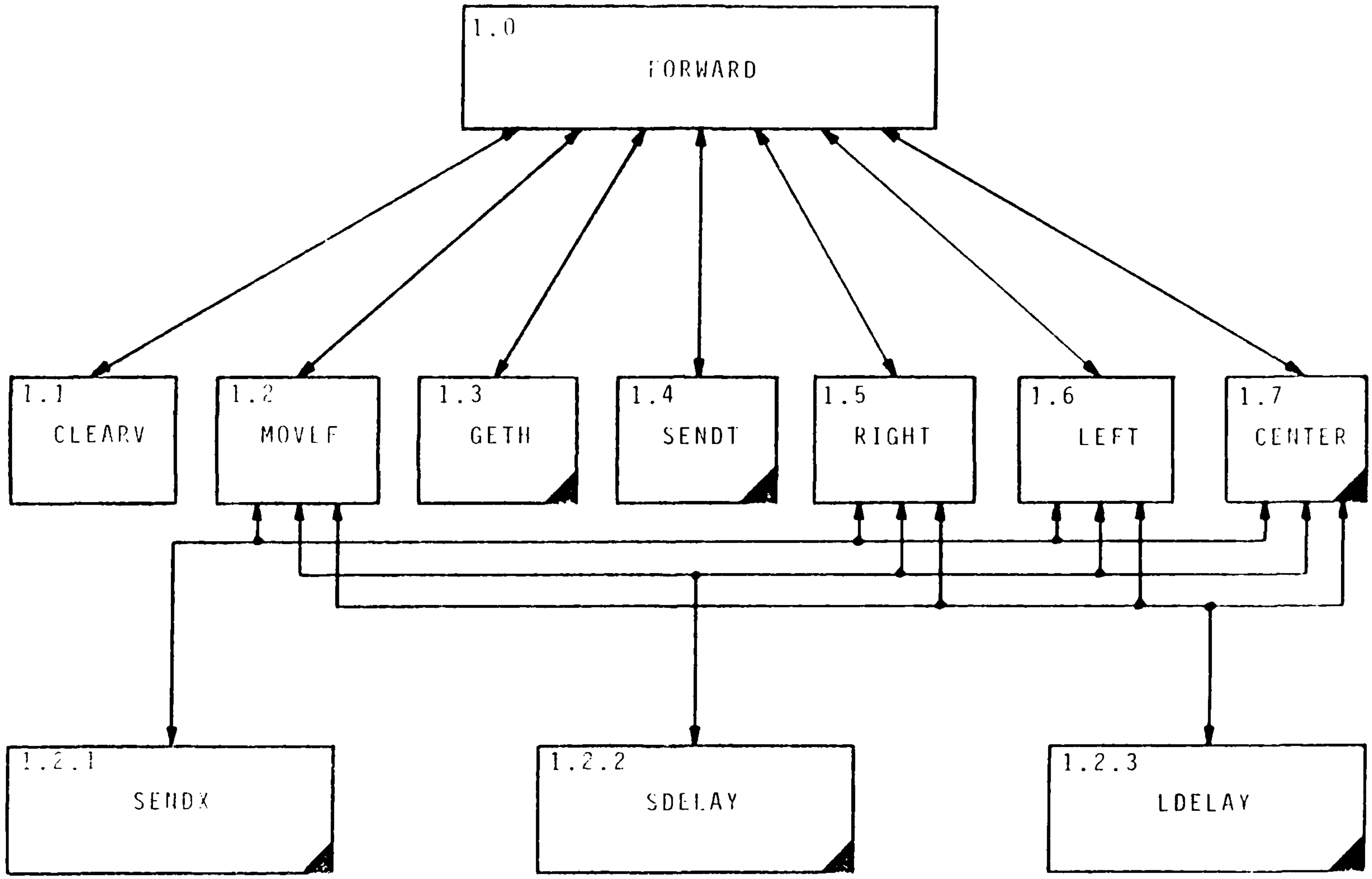
STEP 8: Shutdown all systems. To exit M72, type "CPM". Remove both disks from the drives and turn of the power to the H89 system. Turn off power to the robot, GYRAC, and external power supplies.

NOTE: All references to "control" and "shift" in the H89 command lines refer to the control and shift keys and not the words control and shift. Care must be taken to ensure the various cables to MARRS-1 do not become tangled during movement. In addition, it is assumed that the robot has been "pointed" to the desired initial heading before movement commences. The actual direction of travel is human controlled from the teaching pendant. Also, the GTEST.HEX program must be loaded each time a run is attempted, since the program is cleared on navigation computer reset or by a "control C" in M72 Terminal mode.

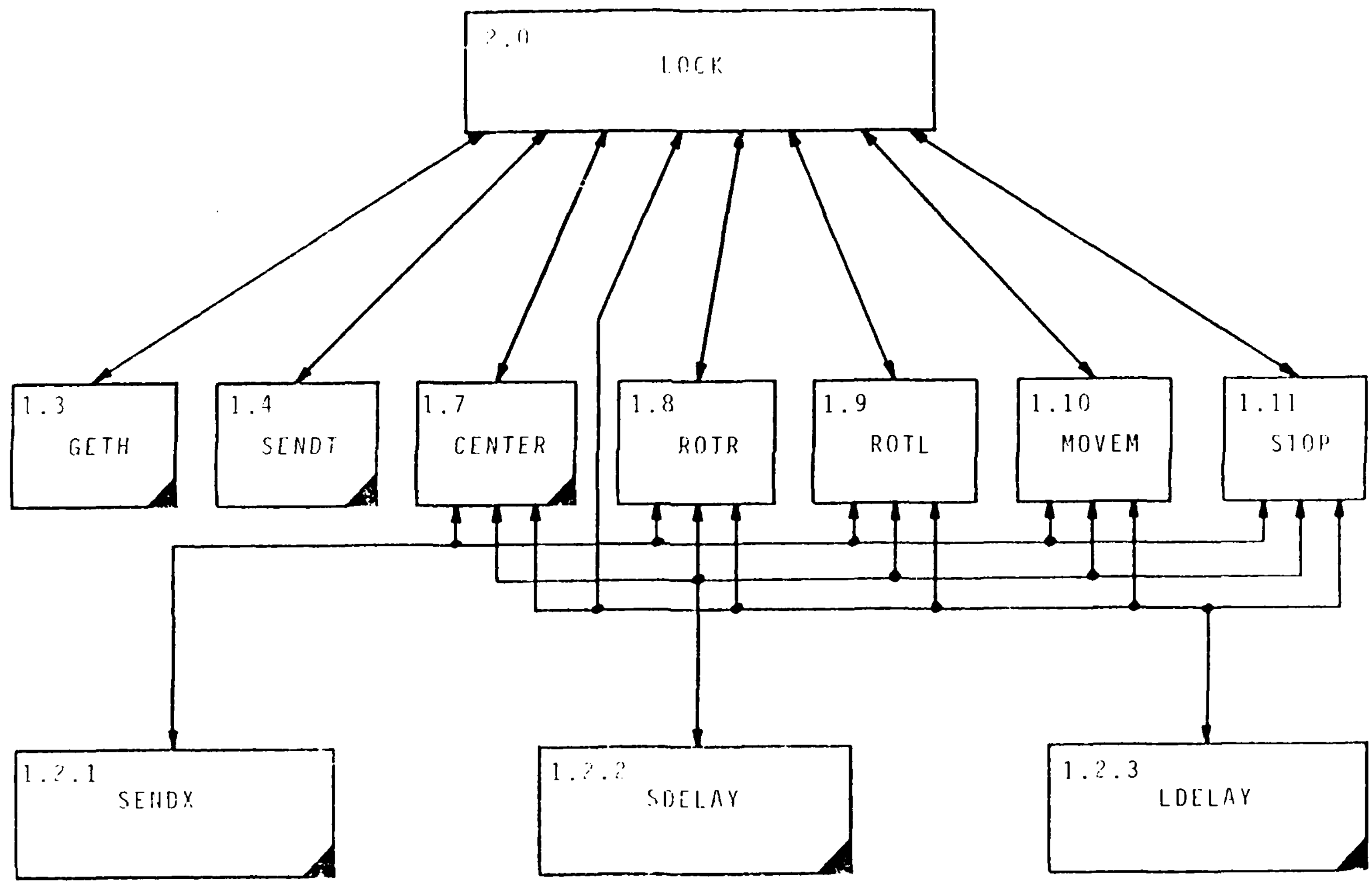
APPENDIX H



NAV.A STRUCTURE CHARTS



NAV.A STRUCTURE CHARTS (continued)



NAV.A STRUCTURE CHARTS (continued)

NAV.A PROGRAM LISTING


```
;*************************************************************************;
;                                                                         ;
;       DATE:    09/12/85                                                 ;
;       VERSION:  1.4                                                     ;
;       TITLE:  NAVIGATION AND HEADING DATA COLLECTION PROGRAM            ;
;       FILENAME:  NAV.A                                                  ;
;       COORDINATOR:  CAPT WILLIAM J. RAMEY JR.                           ;
;       PROJECT:  GYRO AND ACCELEROMETER BASED NAVIGATION SYSTEM FOR A    ;
;                 MOBILE AUTONOMOUS ROBOT (THESIS)                        ;
;       OPERATING SYSTEM:  Z80/Z80 CP/M V2.242 MAGNOLIA MICROSYSTEM 1982; ;
;       LANGUAGE:  RODO-4 ASSEMBLER v1.1 VIRTUAL DEVICES 1984             ;
;       USE:  This program is assembled and then transferred to the      ;
;             robot s navigation computer via an RS-232 link (This must  ;
;             be preceeded by loading the file MARRS.NAV to the drive    ;
;             computer (MENOS) using M72 in Term mode).  NAV.HEX can     ;
;             be loaded into the nav computer using M72 in terminal (T)  ;
;             mode with the following command: L,1000,136F followed by   ;
;             transmitting the file NAV.HEX.  The heading of travel      ;
;             is gheadmm, gheadml, and gheadll which are set at assembly;
;             time and therefore fixed in NAV.HEX.                        ;
;       CONTENTS:  forward                                                ;
;                  lock                                                   ;
;                  clearv                                                 ;
;                  movef                                                  ;
;                  geth                                                   ;
;                  sendt                                                  ;
;                  sendv                                                  ;
;                  right                                                  ;
;                  left                                                   ;
;                  center                                                 ;
;                  cntr                                                   ;
;                  rot                                                    ;
;                  movem                                                  ;
;                  stop                                                   ;
;                  sdelay                                                 ;
;                  ldelay                                                 ;
;       FUNCTION:  This program allows the nav computer to request data  ;
;                  from the GYRAC (over an RS-232 link), transmit GYRAC  ;
;                  heading data to an external computer over an RS-232   ;
;                  link, and steer the robot along a specified heading.  ;
;*************************************************************************;


1000        begin      equ   01000H      ;Starting address of nav program
000?        gheadmm    equ   0?H         ;MSB LS nibble given heading in he
000E        gheadml    equ   0eH         ;LSB MS nibble given heading in he
000?        gheadll    equ   0?H         ;LSB LS nibble given heading in he
0041        comA       equ   41H         ;GYRAC command-- to all 2 bytes
0042        comB       equ   42H         ;GYRAC command-- to all 1 byte
0043        comC       equ   43H         ;GYRAC command-- t  heading 2 bytes
0044        comD       equ   44H         ;GYRAC command-- t  heading 1 byte
0045        comE       equ   45H         ;GYRAC command-- t  velocity 2 bytes
0046        comF       equ   46H         ;GYRAC command-- t  velocity 1 byte
0047        comG       equ   47H         ;GYRAC command-- clear velocity
004?        com?       equ   4?H         ;GYRAC command-- t  ang  [illegible] 2 bytes
004?        comI       equ   4?H         ;GYRAC command-- t  [illegible]
004A        com?       equ   4aH         ;GYRAC command-- t  head & [illegible]
004C        com?       equ   4?H         ;GYRAC command-- t  head & [illegible]
00??        comL       equ   4?H         ;GYRAC command-- reset system
004D        com?       equ   4?H         ;GYRAC command-- t  t  [illegible] RAM
004E        comN       equ   4eH         ;GYRAC command-- RS-232 test
00??        com?       equ   4?H         ;GYRAC command-- t  head 1 & vel 2 bytes
```

```
[illegible]            acia1stat equ   [illegible]   ;[illegible]RAC serial port status & control reg
[illegible]            acia1data equ   [illegible]1H ;[illegible]RAC serial port data register
[illegible]            acia2stat equ   [illegible]   ;MEN[illegible] serial port status & control reg
[illegible]            acia2data equ   [illegible]   ;MEN[illegible] serial port data register
[illegible]            acia3stat equ   [illegible]   ;TERMINAL serial port status & ctrl reg
[illegible]            acia3data equ   [illegible]   ;TERMINAL serial port data register
[illegible]            tdre      equ   [illegible]   ;ACIA transmit data register empty
[illegible]            rdrf      equ   [illegible]   ;ACIA receive data register full
[illegible]            hmask     equ   [illegible]   ;MID mask for heading data

[illegible]                      org   begin         ;Starting address of nav program
[illegible]  BD 10 42 main       jsr   [illegible]   ;Rotate robot until facing given head
[illegible]  BD 10 [illegible]   jsr   forward       ;Move forward along given heading
[illegible]  20 F[illegible]     bra   main
```

```
;***************************************************************************;
;                                                                           ;
;        DATE: [illegible] 12 [illegible]                                   ;
;        VERSION: 1.[illegible]                                             ;
;        NAME: forward                                                      ;
;        MODULE NUMBER: 1.[illegible]                                       ;
;        DESCRIPTION: This module will steer the robot along a given        ;
;                     heading (gheadmm, gheadml, & gheadll).                ;
;        PASSED VARIABLES: None                                             ;
;        RETURNS: None                                                      ;
;        GLOBAL VARIABLES USED: cheadm, cheadl                              ;
;        GLOBAL VARIABLES CHANGED: cheadm, cheadl                           ;
;        REGISTERS USED: A, B                                               ;
;        FILES READ: None                                                   ;
;        FILES WRITTEN: None                                                ;
;        MODULES CALLED: clearv                                             ;
;                        center                                             ;
;                        movef                                              ;
;                        geth                                               ;
;                        sendt                                              ;
;                        right                                              ;
;                        left                                               ;
;        CALLING MODULES: main                                              ;
;        AUTHOR: CAPT WILLIAM J. RAMEY JR.                                  ;
;        HISTORY: None                                                      ;
;***************************************************************************;

100B   BD 10 A2 forward jsr  clearv     ;Clear GYRAC velocity constant
100E   BD 12 05         jsr  center     ;Center the steering wheel
10[illegible]E  BD 10 AF         jsr  movef      ;Start drive motor fast speed

1011   BD 10 E5 forwd   jsr  geth       ;Get GYRAC heading
1014   BD 11 0F         jsr  sendt      ;Send heading to external computer
1017   B6 13 [illegible]E         ldaa cheadm     ;Get MSB of current heading
101A   81 0[illegible]            cmpa #gheadmm   ;If given & current heading are equal
101C   27 07            beq  lowbm      ;Then test LSB MS nibble of heading
101E   2B 1C            bmi  left       ;Else if current < given head turn left
1020   BD 11 [illegible]         jsr  right      ;Else turn right
1023   20 EC            bra  forwd      ;Go get next heading update

1025   B6 13 [illegible]7 lowbm   ldaa cheadl     ;Get LSB of current heading
1028   44               lsra            ;Use only the MS nibble
1029   44               lsra
102A   44               lsra
102B   44               lsra
102C   81 [illegible]E            cmpa #gheadml
102E   27 07            beq  lowbl      ;If current=given test LSB LSN of head
1030   2B 0A            bmi  left       ;Else if current < given head turn left
1032   BD 11 [illegible]         jsr  right      ;Else turn right
1035   20 DA            bra  forwd      ;Go get next heading update

10[illegible]   BD 12 [illegible] lowbl   jsr  center     ;Center steering wheel
103A   [illegible] D5            bra  [illegible]

[illegible]   BD 11 [illegible] left    jsr  left       ;Turn left
10[illegible]F   [illegible]            bra  [illegible]      ;[illegible] next heading update
[illegible]   [illegible]               rts
```

```
;*****************************************************************;
;                                                                 ;
;       DATE: [illegible]                                         ;
;       VERSION: 1.0                                              ;
;       NAME: clearv                                              ;
;       MODULE NUMBER: 1.1                                        ;
;       DESCRIPTION: This module clears the [illegible] velocity constant.
;       PASSED VARIABLES: None                                    ;
;       RETURNS: None                                             ;
;       GLOBAL VARIABLES USED: None                               ;
;       GLOBAL VARIABLES CHANGED: None                            ;
;       REGISTERS USED: A, [illegible]                            ;
;       FILES READ: None                                          ;
;       FILES WRITTEN: None                                       ;
;       MODULES CALLED: None                                      ;
;       CALLING MODULES: forward                                  ;
;       AUTHOR: CAPT WILLIAM [illegible] RAMEY JR.                ;
;       HISTORY: None                                             ;
;*****************************************************************;

10A2   C6 47      clearv  ldab #[illegible]    ;Get GYRAC clear vel command
10A4   B6 C0 10   tcv     ldaa acia1stat       ;Get GYRAC ACIA status
10A7   84 02              anda #tdre           ;Is transmit data reg empty
10A9   27 F9              beq  tcv             ;If no recheck status
10AB   F7 C0 11           stab acia1data       ;Else send GYRAC clear vel command
10AE   39                 rts
```

```
;****************************************************************;
;                                                                ;
;       DATE: [illegible]                                        ;
;       VERSION: 1.[illegible]                                   ;
;       NAME: [illegible]                                        ;
;       MODULE NUMBER: 1.[illegible]                             ;
;       DESCRIPTION: This module sends a [illegible] drive motor fast ;
;                    command sequence to the drive computer (MENDE). ;
;       PASSED VARIABLES: None                                   ;
;       RETURNS: None                                            ;
;       GLOBAL VARIABLES USED: None                              ;
;       GLOBAL VARIABLES CHANGED: None                           ;
;       REGISTERS USED: A, B                                     ;
;       FILES READ: None                                         ;
;       FILES WRITTEN: None                                      ;
;       MODULES CALLED: send, sdelay, ldelay                     ;
;       CALLING MODULES: forward                                 ;
;       AUTHOR: CAPT WILLIAM [illegible] RAMEY JR.               ;
;       HISTORY: None                                            ;
;****************************************************************;

[illegible]  [illegible] 41     movef   ldab # A     ;Send to drive computer (MENDE)
[illegible]  BD 13 3D          jsr   send       ;start drive motor fast cont command
[illegible]  BD [illegible] 4[illegible]  jsr   sdelay     ;sequence

[illegible]  [illegible] 44             ldab # D
[illegible]  BD [illegible] 3D          jsr   send
[illegible]  BD 13 4[illegible]         jsr   sdelay

[illegible]  C6 30             ldab # 0
[illegible]  BD 13 3D          jsr   send
[illegible]  BD 13 4E          jsr   sdelay

[illegible]  C6 31             ldab # 1
[illegible]  [illegible] 3D    jsr   send
[illegible]  BD 13 4[illegible]  jsr   sdelay

[illegible]  [illegible]       ldab # [illegible]
[illegible]  BD 13 3D          jsr   send
[illegible]  BD 13 4E          jsr   sdelay

[illegible]  [illegible]       ldab # [illegible]
[illegible]  BD 13 3D          jsr   send
[illegible]  BD 13 4E          jsr   sdelay

[illegible]  BD 13 5[illegible]  jsr   ldelay
[illegible]  BD 13 5[illegible]  jsr   ldelay
[illegible]  39                rts
```

```
;****************************************************************
;                                                               ;
;       DATE:     [illegible]                                   ;
;       VERSION:  1.0                                           ;
;       NAME:  sendy                                            ;
;       MODULE NUMBER:  1.2.1                                   ;
;       DESCRIPTION:  This module sends a byte of data to the drive ;
;                     computer (MENOS) over an RS-232 link.     ;
;       PASSED VARIABLES:  Output data (passed in B reg)        ;
;       RETURNS:  None                                          ;
;       GLOBAL VARIABLES USED:  None                            ;
;       GLOBAL VARIABLES CHANGED:  None                         ;
;       REGISTERS USED:  A, B                                   ;
;       FILES READ:  None                                       ;
;       FILES WRITTEN:  None                                    ;
;       MODULES CALLED:  None                                   ;
;       CALLING MODULES:  move, right, left, center, rotate, stop ;
;       AUTHOR: CAPT WILLIAM [illegible] RAMEY JR.              ;
;       HISTORY : None                                          ;
;****************************************************************;

113D    B6 [illegible] 08 sendy   ldaa aciastat   ;Get MENOS ACIA status
1140    84 02                     anda #tdre      ;Is transmit data reg empty
1142    27 F9                     beq  sendy      ;If no recheck status
1144    F7 [illegible]            stab aciadata   ;Else send data to MENOS
1147    39                        rts
```

```
;*****************************************************************************
;
;       DATE: 08/21/85
;       VERSION: 1.0
;       NAME: sdelay
;       MODULE NUMBER: 1.2.2
;       DESCRIPTION:  This module provides for a short fixed time delay
;       PASSED VARIABLES: None
;       RETURNS: None
;       GLOBAL VARIABLES USED: None
;       GLOBAL VARIABLES CHANGED: None
;       REGISTERS USED: A, B
;       FILES READ: None
;       FILES WRITTEN: None
;       MODULES CALLED: None
;       CALLING MODULES: move, right, left, center, rotate, stop
;       AUTHOR: CAPT WILLIAM C. RAME, JR.
;       HISTORY: None
;*****************************************************************************

1148   37         sdelay  pshb              ;Save registers
1149   36                 psha
114A   86 1F              ldaa #1fH         ;Execute fixed time delay
114C   C6 FF      sd1     ldab #0ffH
114E   01         sd2     nop
114F   5A                 decb
1150   26 FC              bne  sd2
1152   4A                 deca
1153   26 F7              bne  sd1
1155   32                 pula              ;Restore registers
1156   33                 pulb
1157   39                 rts
```

```
;*****************************************************************************;
;                                                                             ;
;       DATE: [illegible]                                                     ;
;       VERSION: [illegible]                                                  ;
;       NAME: delay                                                           ;
;       MODULE NUMBER: 1.1.3                                                  ;
;       DESCRIPTION: This module provides a fixed long time delay             ;
;       PASSED VARIABLES: None                                                ;
;       RETURNS: None                                                         ;
;       GLOBAL VARIABLES USED: None                                           ;
;       GLOBAL VARIABLES CHANGED: None                                        ;
;       REGISTERS USED: A, B                                                  ;
;       FILES READ: None                                                      ;
;       FILES WRITTEN: None                                                   ;
;       MODULES CALLED: None                                                  ;
;       CALLING MODULES: move, right, left, center, rotate, stop, look        ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                                     ;
;       HISTORY: None                                                         ;
;*****************************************************************************;

[illegible]  37         delay  pshb               ;Save registers
[illegible]  36                psha
[illegible]  86 DF             ldaa #0dfH         ;Execute time delay
[illegible]  C6 FF      ld1    ldab #0ffH
[illegible]  01         ld2    nop
[illegible]  5A                decb
[illegible]  26 FC             bne  ld2
[illegible]  4A                deca
[illegible]  26 F7             bne  ld1
[illegible]  32                pula
[illegible]  33                pulb
[illegible]  39                rts


[illegible]             cheadh rmb  1
[illegible]             cheadl rmb  1

[illegible]                    END
```

```
;*****************************************************************;
;                                                                 ;
;        [illegible]                                              ;
;        [illegible]                                              ;
;        [illegible]                                              ;
;        MODULE NUMBER: 1.[illegible]                             ;
;        DESCRIPTION: This module reads the 2 bytes of heading data from ;
;                     the GYRAC and saves it in global variables  ;
;                     cheadm and cheadl.                          ;
;        PASSED VARIABLES: None                                   ;
;        RETURNS: None                                            ;
;        GLOBAL VARIABLES USED: None                              ;
;        GLOBAL VARIABLES CHANGED: cheadm, cheadl                 ;
;        REGISTERS USED: A, B                                     ;
;        FILES READ: None                                         ;
;        FILES WRITTEN: None                                      ;
;        MODULES CALLED: None                                     ;
;        CALLING MODULES: [illegible]                             ;
;        AUTHOR: CAPT WILLIAM J. RAME  JR.                        ;
;        HISTORY: None                                            ;
;*****************************************************************;

10E6   C6 40      geth   ldab #cmnd        ;Get GYRAC 2 byte heading command
10E8   B6 C0 10   g1     ldaa acialstat    ;Get GYRAC ACIA status
10EB   84 02             anda #tdre        ;Is transmit data reg empty,
10ED   27 F9             beq  g1           ;If no recheck status
10EF   F7 C0 11          stab acialdata    ;Else send GYRAC command

10F2   B6 C0 10   g2     ldaa acialstat    ;Get GYRAC ACIA status
10F5   84 01             anda #rdrf        ;Is receive data register full?
10F7   27 F9             beq  g2           ;If no recheck status
10F9   B6 C0 11          ldaa acialdata    ;Else get the MSB of heading data
10FC   84 0F             anda #mask        ;Mask out MS 4 bits
10FE   B7 13 66          staa cheadm       ;Save current MSB heading

1101   B6 C0 10   g3     ldaa acialstat    ;Get GYRAC ACIA status
1104   84 01             anda #rdrf        ;Is receive data register full?
1106   27 F9             beq  g3           ;If no recheck status
1108   B6 C0 11          ldaa acialdata    ;Else get the LSB of heading data
110B   B7 13 67          staa cheadl       ;Save current LSB of heading
110E   39                rts
```

NAV.A PROGRAM LISTING (continued)

```
;*****************************************************************;
;                                                                 ;
;       DATE: [illegible]                                         ;
;       VERSION: 1.0                                              ;
;       NAME: sendt                                               ;
;       MODULE NUMBER: 1.4                                        ;
;       DESCRIPTION: This module sends two bytes of heading data  ;
;                    represented as four ASCII hex characters to an ;
;                    external terminal.                           ;
;       PASSED VARIABLES: None                                    ;
;       RETURNS: None                                             ;
;       GLOBAL VARIABLES USED: cheadm, cheadl                     ;
;       GLOBAL VARIABLES CHANGED: None                            ;
;       REGISTERS USED: None                                      ;
;       FILES READ: None                                          ;
;       FILES WRITTEN: None                                       ;
;       MODULES CALLED: None                                      ;
;       CALLING MODULES: forward, [illegible]                     ;
;       AUTHOR: CAPT WILLIAM J. FAHEY JR.                         ;
;       HISTORY: None                                             ;
;*****************************************************************;

110F   F6 13 68 sendt   ldab cheadm      ;Get MSB of current heading
1112   54               lsrb             ;Convert upper nibble of MSB to ASCII
1113   54               lsrb
1114   54               lsrb
1115   54               lsrb
1116   C4 0F            andb #0FH
1118   CB 30            addb #30H
111A   C1 39            cmpb #39H
111C   23 02            bls  s1
111E   CB 07            addb #07H
1120   B6 C0 00 s1      ldaa aciatstat   ;Get terminal ACIA status
1123   84 02            anda #tdre       ;Is transmit data reg empty
1125   27 F9            beq  s1          ;If no recheck status
1127   F7 C0 01         stab aciatdata   ;Else send data

112A   F6 13 68         ldab cheadm      ;Get MSB of current heading
112D   C4 0F            andb #0FH        ;Convert lower nibble of MSB to ASCII
112F   CB 30            addb #30H
1131   C1 39            cmpb #39H
1133   23 02            bls  s2
1135   CB 07            addb #07H
1137   B6 C0 00 s2      ldaa aciatstat   ;Get terminal ACIA status
113A   84 02            anda #tdre       ;Is transmit data reg empty
113C   27 F9            beq  s2          ;If no recheck status
113E   F7 C0 01         stab aciatdata   ;Else send data

1141   F6 13 69         ldab cheadl      ;Get LSB of current heading
1144   54               lsrb             ;Convert upper nibble of LSB to ASCII
1145   54               lsrb
1146   54               lsrb
1147   54               lsrb
1148   C4 0F            andb #0FH
114A   CB [illegible]   addb #[illegible]H
114C   C1 [illegible]   cmpb #[illegible]H
114E   23 [illegible]   bls  s[illegible]
1150   CB [illegible]   addb #07H
1152   [illegible]  s[illegible]  ldaa aciatstat   ;Get terminal ACIA status
1155   84 02            anda #tdre       ;Is transmit data reg empty
1157   27 F9            beq  s[illegible] ;If no recheck status
115[illegible]   F7 [illegible]   stab aciatdata   ;Else send data
```

NAV.A PROGRAM LISTING (continued)

```
1150  F[illegible] 1[illegible] [illegible]   ldab [illegible]headl   ;Get L[illegible] of current heading
115F  C4 0F                andb #0FH      ;Convert lower nibble of L[illegible] to ASCII
[illegible]  [illegible]      addb #[illegible]H
[illegible]  [illegible]      cmpb #[illegible]H
[illegible]  [illegible]      bls  $4
[illegible]  [illegible]      addb #07H
[illegible]  [illegible] $4   ldaa aciatstat  ;Get terminal ACIA status
[illegible]  84 [illegible]   anda #tdre      ;Is transmit data reg empty
[illegible]E  27 F[illegible] beq  $4         ;If no recheck status
117[illegible]  F7 [illegible]  stab aciatdata ;Else send data

1173  [illegible] 0D       ldab #0DH      ;Get carriage return
1175  B6 [illegible] $5   ldaa aciatstat  ;Get terminal ACIA status
1178  84 02                anda #tdre      ;Is transmit data reg empty
117A  27 F9                beq  $5         ;If no recheck status
117[illegible]  F7 [illegible] 01  stab aciatdata ;Else send data

117F  [illegible] 0A       ldab #0AH      ;Get line feed
1181  B[illegible] [illegible] $[illegible]  ldaa aciatstat  ;Get terminal ACIA status
1184  84 02                anda #tdre      ;Is transmit data reg empty
11[illegible]  27 F9        beq  $5         ;If no recheck status
11[illegible]  F7 [illegible] 01  stab aciatdata ;Else send data

118B  39                   rts
```

```
;***********************************************************************;
;                                                                       ;
;       DATE:  03/26/85                                                 ;
;       VERSION:  1.0                                                   ;
;       NAME:  right                                                    ;
;       MODULE NUMBER:  1.5                                             ;
;       DESCRIPTION:  This module sends a right turn relative mode      ;
;                     command sequence to the drive computer (MENDS).   ;
;       PASSED VARIABLES:  None                                         ;
;       RETURNS:  None                                                  ;
;       GLOBAL VARIABLES USED:  None                                    ;
;       GLOBAL VARIABLES CHANGED:  None                                 ;
;       REGISTERS USED:  A, B                                           ;
;       FILES READ:  None                                               ;
;       FILES WRITTEN:  None                                            ;
;       MODULES CALLED:  sdelay, ldelay, send                           ;
;       CALLING MODULES:  forward                                       ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                               ;
;       HISTORY: None                                                   ;
;***********************************************************************;

118B    C6 41      right    ldab #'A'        ;Send to drive computer (MENDS)
118E    BD 13 3D            jsr  sendx       ;right turn continuous command
1191    BD 13 48            jsr  sdelay      ;sequence

1194    C6 44               ldab #'D'
1196    BD 13 3D            jsr  sendx
1199    BD 13 48            jsr  sdelay

119C    C6 30               ldab #'0'
119E    BD 13 3D            jsr  sendx
11A1    BD 13 48            jsr  sdelay

11A4    C6 31               ldab #'1'
11A6    BD 13 3D            jsr  sendx
11A9    BD 13 48            jsr  sdelay

11AC    C6 30               ldab #'0'
11AE    BD 13 3D            jsr  sendx
11B1    BD 13 48            jsr  sdelay

11B4    C6 39               ldab #'9'
11B6    BD 13 3D            jsr  sendx
11B9    BD 13 48            jsr  sdelay

11BC    BD 13 51            jsr  ldelay
11BF    BD 13 51            jsr  ldelay
11C2    BD 13 51            jsr  ldelay
11C5    BD 13 51            jsr  ldelay
11C8    39                  rts
```

```
;****************************************************************
;                                                               ;
;       DATE:  [illegible]                                      ;
;       VERSION:  1.0                                           ;
;       NAME:  left                                             ;
;       MODULE NUMBER:  1.6                                     ;
;       DESCRIPTION:  This module sends a left turn relative mode ;
;                     command sequence to the driver computer (MENOS ;
;       PASSED VARIABLES:  None                                 ;
;       RETURNS:  None                                          ;
;       GLOBAL VARIABLES USED:  None                            ;
;       GLOBAL VARIABLES CHANGED:  None                         ;
;       REGISTERS USED:  A, B                                   ;
;       FILES READ:  None                                       ;
;       FILES WRITTEN:  None                                    ;
;       MODULES CALLED:  sdelay, ldelay, sendx                  ;
;       CALLING MODULES:  forward                               ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                       ;
;       HISTORY: None                                           ;
;****************************************************************;

11C9   C6 41      left   ldab #'A      ;Send to drive computer (MENOS)
11CB   BD 11 3D          jsr  sendx    ;left turn continous command
11CE   BD 13 48          jsr  sdelay   ;sequence

11D1   C6 44             ldab #'D
11D3   BD 13 3D          jsr  sendx
11D6   BD 13 48          jsr  sdelay

11D9   C6 30             ldab #'0
11DB   BD 11 3D          jsr  sendx
11DE   BD 13 48          jsr  sdelay

11E1   [illegible] 31    ldab #'1
11E3   BD 11 3D          jsr  sendx
11E6   BD 13 48          jsr  sdelay

11E9   [illegible]       ldab #'0
11EB   BD 11 3D          jsr  sendx
11EE   BD 11 48          jsr  sdelay

[illegible]  [illegible] 34   ldab #'4
[illegible]  [illegible]      jsr  sendx
[illegible]  [illegible]      jsr  sdelay

[illegible]  [illegible]      jsr  ldelay
[illegible]  [illegible]      jsr  ldelay
[illegible]  [illegible]      jsr  ldelay
[illegible]  [illegible]      jsr  ldelay
[illegible]                   rts
```

```
;**************************************************************************;
;                                                                          ;
;       DATE: [illegible]                                                  ;
;       VERSION: 1.[illegible]                                             ;
;       NAME: center                                                       ;
;       MODULE NUMBER: 1.7                                                 ;
;       DESCRIPTION: This module sends a center steering motor             ;
;                    command sequence to the drive computer (MENTOR).      ;
;       PASSED VARIABLES: None                                             ;
;       RETURNS: None                                                      ;
;       GLOBAL VARIABLES USED: None                                        ;
;       GLOBAL VARIABLES CHANGED: None                                     ;
;       REGISTERS USED: A, B                                               ;
;       FILES READ: None                                                   ;
;       FILES WRITTEN: None                                                ;
;       MODULES CALLED: sendc, sdelay, ldelay                              ;
;       CALLING MODULES: forward, [illegible]                              ;
;       AUTHOR: CAPT WILLIAM J. FAME JR.                                   ;
;       HISTORY: None                                                      ;
;**************************************************************************;

1206   C6 41      center   ldab #'A'     ;Send to drive computer (MENTOR)
1208   BD 13 DD            jsr  sendc    ;center steering motor command
120B   BD 13 4E            jsr  sdelay   ;sequence

120E   C6 44               ldab #'D'
1210   BD 13 DD            jsr  sendc
1213   BD 13 4E            jsr  sdelay

1216   C6 30               ldab #'0'
1218   BD 13 DD            jsr  sendc
121B   BD 13 4E            jsr  sdelay

121E   C6 31               ldab #'1'
1220   BD 13 DD            jsr  sendc
1223   BD 13 4E            jsr  sdelay

1226   C6 30               ldab #'0'
1228   BD 13 DD            jsr  sendc
122B   BD 13 4E            jsr  sdelay

122E   C6 45               ldab #'E'
1230   BD 13 DD            jsr  sendc
1233   BD 13 4E            jsr  sdelay

1236   BD 13 5B            jsr  ldelay
1239   BD 13 5B            jsr  ldelay

123C   BD 13 5B            jsr  ldelay
123F   BD 13 5B            jsr  ldelay
1242   39                  rts
```

```
;*****************************************************************************:
;                                                                            :
;        DATE: [illegible]                                                   :
;        VERSION: [illegible]                                                :
;        NAME: [illegible]                                                   :
;        MODULE NUMBER: [illegible]                                          :
;        DESCRIPTION:  This module will issue the command sequence to        :
;                      cause the steering wheel to move so the robot         :
;                      will rotate right when the drive motor is turned      :
;                      on.                                                   :
;        PASSED VARIABLES:  None                                             :
;        RETURNS:  None                                                      :
;        GLOBAL VARIABLES USED:  None                                        :
;        GLOBAL VARIABLES CHANGED:  None                                     :
;        REGISTERS USED:  A, B                                               :
;        FILES READ:  None                                                   :
;        FILES WRITTEN:  None                                                :
;        MODULES CALLED:  sendx, sdelay, ldelay                              :
;        CALLING MODULES:  [illegible]                                       :
;        AUTHOR: CAPT WILLIAM J. RAMEY JR.                                   :
;        HISTORY: None                                                       :
;*****************************************************************************:

1243   C6 41      rotr     ldab # 'A'        ;Send to drive computer  MENDS)
1245   BD 13 9D            jsr  sendx        ;turn 90 deg right steering motor
1248   BD 13 4B            jsr  sdelay       ;command sequence

124B   C6 44               ldab # 'D'
124D   BD 13 9D            jsr  sendx
1250   BD 13 4B            jsr  sdelay

1253   C6 [illegible]      ldab # [illegible]
1255   BD 13 9D            jsr  sendx
1258   BD 13 4B            jsr  sdelay

125B   C6 [illegible]      ldab # [illegible]
125D   BD 13 9D            jsr  sendx
1260   BD 13 4B            jsr  sdelay

1263   C6 [illegible]      ldab # [illegible]
1265   BD 13 9D            jsr  sendx
1268   BD 13 4B            jsr  sdelay

126B   [illegible]         ldab # [illegible]
126D   BD 13 9D            jsr  sendx
1270   BD 13 4B            jsr  sdelay
1273   BD 13 5B            jsr  ldelay

1276   BD 13 5B            jsr  ldelay
1279   BD 13 5B            jsr  ldelay
127C   BD 13 [illegible]   jsr  ldelay
127F   BD 13 [illegible]   jsr  ldelay
1282   BD 13 [illegible]   jsr  ldelay
1285   BD 13 [illegible]   jsr  ldelay
1288   [illegible]         rts
```

```
;*****************************************************************;
;                                                                 ;
;       DATE:  11-12-85                                           ;
;       VERSION:  1.0                                             ;
;       NAME:  rotl                                               ;
;       MODULE NUMBER:  1.9                                       ;
;       DESCRIPTION:  This module will issue the command sequence to ;
;                     cause the steering wheel to move so the robot ;
;                     will rotate left when the drive motor is turned ;
;                     on.                                         ;
;       PASSED VARIABLES:  None                                   ;
;       RETURNS:  None                                            ;
;       GLOBAL VARIABLES USED:  None                              ;
;       GLOBAL VARIABLES CHANGED:  None                           ;
;       REGISTERS USED:  A, B                                     ;
;       FILES READ:  None                                         ;
;       FILES WRITTEN:  None                                      ;
;       MODULES CALLED:  sendc, sdelay, ldelay                    ;
;       CALLING MODULES:  look                                    ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                         ;
;       HISTORY: None                                             ;
;*****************************************************************;

1239   C6 41      rotl    ldab #'A'      ;Send to drive computer (MENDE)
123B   BD 13 0D           jsr  sendc     ;turn 90 deg left steering motor
123E   BD 13 4B           jsr  sdelay    ;command sequence

1241   C6 44              ldab #'D'
1243   BD 13 0D           jsr  sendc
1246   BD 13 4B           jsr  sdelay

1249   C6 30              ldab #'0'
124B   BD 13 0D           jsr  sendc
124E   BD 13 4B           jsr  sdelay

1251   C6 31              ldab #'1'
1253   BD 13 0D           jsr  sendc
1256   BD 13 4B           jsr  sdelay

1259   C6 31              ldab #'1'
125B   BD 13 0D           jsr  sendc
125E   BD 13 4B           jsr  sdelay

1261   C6 46              ldab #'F'
1263   BD 13 0D           jsr  sendc
1266   BD 13 4B           jsr  sdelay
1269   BD 13 5B           jsr  ldelay

126C   BD 13 5B           jsr  ldelay
126F   BD 13 5B           jsr  ldelay
1272   BD 13 5B           jsr  ldelay
1275   BD 13 5B           jsr  ldelay
1278   BD 13 5B           jsr  ldelay
127B   BD 13 5B           jsr  ldelay
127E   39                 rts
```

```
;*****************************************************************;
;                                                                 ;
;       DATE:  [illegible]                                        ;
;       VERSION:  1.0                                             ;
;       NAME:  [illegible]                                        ;
;       MODULE NUMBER:  1.1[illegible]                            ;
;       DESCRIPTION:  This module will issue the command sequence to
;                     cause the drive motor to turn on at med speed.
;       PASSED VARIABLES:  None                                   ;
;       RETURNS:  None                                            ;
;       GLOBAL VARIABLES USED:  None                              ;
;       GLOBAL VARIABLES CHANGED:  None                           ;
;       REGISTERS USED:  A, B                                     ;
;       FILES READ:  None                                         ;
;       FILES WRITTEN:  None                                      ;
;       MODULES CALLED:  sendc, sdelay, ldelay                    ;
;       CALLING MODULES:  [illegible]                             ;
;       AUTHOR: CAPT WILLIAM [illegible]. RAMEY JR.               ;
;       HISTORY : None                                            ;
;*****************************************************************;

12CF    C6 41      movem    ldab #'A'       ;Send to drive computer (MENCS)
12D1    BD 13 3D            jsr  sendc      ;turn 90 deg steering motor command
12D4    BD 13 43            jsr  sdelay     ;sequence

12D7    C6 44               ldab #'D'
12D9    BD 13 3D            jsr  sendc
12DC    BD 13 43            jsr  sdelay

12DF    C6 30               ldab #'0'
12E1    BD 13 3D            jsr  sendc
12E4    BD 13 43            jsr  sdelay

12E7    C6 31               ldab #'1'
12E9    BD 13 3D            jsr  sendc
12EC    BD 13 43            jsr  sdelay

12EF    C6 31               ldab #'1'
12F1    BD 13 3D            jsr  sendc
12F4    BD 13 43            jsr  sdelay

12F7    C6 42               ldab #'B'
12F9    BD 13 3D            jsr  sendc
12FC    BD 13 43            jsr  sdelay

12FF    BD 13 53            jsr  ldelay
1302    BD 13 53            jsr  ldelay
1305    39                  rts
```

```
;*************************************************************************;
;                                                                         ;
;       DATE:  [illegible]                                                ;
;       VERSION:  [illegible]                                             ;
;       NAME:  stop                                                       ;
;       MODULE NUMBER:  1.11                                              ;
;       DESCRIPTION:  This module sends the command sequence to stop      ;
;                     the drive and steering motors.                      ;
;       PASSED VARIABLES:  None                                           ;
;       RETURNS:  None                                                    ;
;       GLOBAL VARIABLES USED:  None                                      ;
;       GLOBAL VARIABLES CHANGED:  None                                   ;
;       REGISTERS USED:  A, B                                             ;
;       FILES READ:  None                                                 ;
;       FILES WRITTEN:  None                                              ;
;       MODULES CALLED:  sendc, sdelay, ldelay                            ;
;       CALLING MODULES:  [illegible]                                     ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                                 ;
;       HISTORY: None                                                     ;
;*************************************************************************;

1306    C6 41     stop    ldab #'A'       ;Send to drive computer (MENDS)
1308    BD 13 3D          jsr  sendc      ;stop steering and drive motor command
130B    BD 13 48          jsr  sdelay     ;sequence

130E    C6 44             ldab #'D'
1310    BD 13 3D          jsr  sendc
1313    BD 13 48          jsr  sdelay

1316    C6 30             ldab #'0'
1318    BD 13 3D          jsr  sendc
131B    BD 13 48          jsr  sdelay

131E    C6 31             ldab #'1'
1320    BD 13 3D          jsr  sendc
1323    BD 13 48          jsr  sdelay

1326    C6 [illegible]    ldab #'[illegible]'
1328    BD 13 3D          jsr  sendc
132B    BD 13 48          jsr  sdelay

132E    C6 [illegible]    ldab #'[illegible]'
1330    BD 13 3D          jsr  sendc
1333    BD 13 48          jsr  sdelay

1336    BD 13 [illegible] jsr  ldelay
1339    BD 13 [illegible] jsr  ldelay
133C    39                rts
```

```
;*****************************************************************;
;                                                                 ;
;       DATE: [illegible]/12/85                                   ;
;       VERSION: 1.0                                              ;
;       NAME: [illegible]                                         ;
;       MODULE NUMBER: [illegible]                                ;
;       DESCRIPTION: This module will rotate the robot until it is;
;                    within 1.5 degrees of the given heading.     ;
;       PASSED VARIABLES: None                                    ;
;       RETURNS: None                                             ;
;       GLOBAL VARIABLES USED: cheadm, cheadl                     ;
;       GLOBAL VARIABLES CHANGED: cheadm, cheadl                  ;
;       REGISTERS USED: A, B                                      ;
;       FILES READ: None                                          ;
;       FILES WRITTEN: None                                       ;
;       MODULES CALLED: center                                    ;
;                       moven                                     ;
;                       geth                                      ;
;                       sendt                                     ;
;                       rotr                                      ;
;                       rotl                                      ;
;                       stop                                      ;
;                       ldelay                                    ;
;       CALLING MODULES: main                                     ;
;       AUTHOR: CAPT WILLIAM J. RAMEY JR.                         ;
;       HISTORY: None                                             ;
;*****************************************************************;

1042   BD 13 E8  lock    jsr  geth       ;Get G RAC heading
1045   BD 11 [illegible]         jsr  sendt      ;Send heading to external computer
1048   [illegible] 10 [illegible]          ldaa cheadm     ;Get MSB of current heading
104B   [illegible] [illegible]             cmpa #gheadm    ;If given & current heading are equal
104D   27 07             beq  [illegible]4       ;Then test LSB MS nibble of heading
104F   [illegible] 17             bmi  [illegible]    ;Else if current < given turn left
1051   BD 12 4[illegible]          jsr  rotr       ;Else move steering wheel 90 right
1054   20 17             bra  [illegible]

1056   [illegible] 10 [illegible]  [illegible]4  ldaa cheadl     ;Get LSB of current heading
1059   44                lsra            ;Use only the MS nibble
105A   44                lsra
105B   44                lsra
105C   44                lsra
105D   [illegible] [illegible]             cmpa #gheadl    ;If current heading = given heading
105F   27 [illegible]             beq  [illegible]       ;Then then move forward
1061   [illegible] [illegible]5             bmi  [illegible]     ;Else if current < given head turn left
1063   BD 12 4[illegible]          jsr  rotr       ;Else move steering wheel 90 right
1066   2[illegible] [illegible]             bra  [illegible]

[illegible]   [illegible]  [illegible]             jsr  rotl       ;Move steering wheel 90 deg left
[illegible]   [illegible]                bra  [illegible]

[illegible]   [illegible]  [illegible]             jsr  [illegible]       ;Start drive [illegible] steer
[illegible]   [illegible]  [illegible]             jsr  geth       ;Get G RAC heading
[illegible]   [illegible]                ldaa cheadm     ;Get MSB of current heading
[illegible]   [illegible]                cmpa #gheadm    ;If given & current heading are equal
[illegible]   [illegible]                beq  [illegible]       ;Then test LSB MS nibble of heading
[illegible]   [illegible]                bra  [illegible]       ;Else go get next heading update
```

```
[illegible]  B6 11 [illegible] lock2   ldaa chead1     ;Get [illegible] of current heading
107F  44                   lsra            ;Use only the MS nibble
1080  44                   lsra
1081  44                   lsra
1082  44                   lsra
1083  81 [illegible]       cmpa #ghead[illegible]  ;If current heading = given heading
1085  27 02                beq  lock3      ;Then stop rotating
1087  20 E[illegible]      bra  lock1      ;Else go get next heading update

1089  BD 12 86 lock3       jsr  stop       ;Stop drive motor
108C  BD 12 [illegible]    jsr  center     ;Center steering wheel

108F  BD 13 [illegible]    jsr  1delay
1092  BD 13 [illegible]    jsr  1delay
1095  BD 13 [illegible]    jsr  1delay
1098  BD 13 [illegible]    jsr  1delay
109B  BD 13 [illegible]    jsr  1delay
109E  BD 13 [illegible]    jsr  1delay
10A1  39                   rts
```

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	forward	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	path	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
left	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	left	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
moven	[illegible]	[illegible]	[illegible]	main	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	right	[illegible]	[illegible]	[illegible]
s4	[illegible]	[illegible]	[illegible]	s2	[illegible]	s3	[illegible]
[illegible]	134E	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	send	[illegible]
				[illegible]	[illegible]		

MARRS.NAV PROGRAM LISTING

Address (HEX)	Instruction Code (HEX)	Comment
NONE	AA	Put MARRS-1 into
NONE	01	Instruction Input
NONE	00	Mode at Address 0100
0000	CC	Start Drive Motor
0101	1B	Forward at Fast
0102	FF	Speed (movef)
0103	3A	
0004	03	Turn Steering Motor
0105	DC	One Increment to the
0106	EC	Left (left)
0107	01	
0108	3A	
0009	03	Turn Steering Motor
010A	DC	One Increment to the
010B	E8	RIGHT (right)
010C	01	
010D	3A	
010E	03	Turn Steering Motor
010F	CC	to the Center,
0110	E8	straight, Position
0111	49	(center)
0112	3A	
0113	03	Turn Steering Motor
0114	CC	to the Full Right
0115	E8	Position (rotr)
0116	90	
0117	3A	
0118	02	Stop all Motor
0119	03	Movement (stop)
011A	3A	
010B	CC	Start Drive Motor
010C	13	Forward at Medium
010D	FF	Speed (movem)
010E	3A	
011F	03	Turn Steering Motor
0120	CC	to the Full Left
0121	EC	Position (rotl)
0122	00	
0123	3A	
NONE	R	Reset to Input Mode

NAV.A OPERATING INSTRUCTIONS

STEP 1: Power up the H89 computer system. Place the System disk in Drive A and the gyro program disk in Drive B. Boot the system (type "B 29") and change the mode of Drive B to single sided double density (type "mode B:ss,dd"). Change the working drive to Drive B (type "B:").

STEP 2: Connect the H89 to Robot RS-232 cable between the H89 DCE connector and the MARRS-1 Drive Computer (MENOS) connector. Power up the MARRS-1 Robot. Make sure the batteries are fully charged and the charger power line is connected and turned on. Press both the system reset key on the keypad and the Nav computer reset button.

STEP 3: Load and run the M72 modem program on the H89 by typing "M72". When this program is running type "SPD" to change the transmission time delays. When prompted for time delays reply with a "1" for both character and line delay times. Set the H89 keyboard caps lock on.

STEP 4: Load and transmit the MARRS.NAV file to the robot's Drive computer. This is done by typing "T" to enter the M72 Terminal mode. Next type "control shift " followed by "control T" and then "MARRS.NAV" to send the program file. When asked if time delays are desired, answer Yes. The file will be displayed as it is being transmitted. When it is finished you will see the data stop and hear the robot say "READY". Type "control shift " followed by "control T" to return to the M72 command mode. This entire step can be skipped if the program is hand keyed directly into MARRS-1 via the onboard keypad (which is the recommended way since it avoids moving cables). NOTE: MARRS-1 system resets do not erase this program.

STEP 5: Connect the H89 to Robot RS-232 cable between the H89 DCE connector and the MARRS-1 Nav T connector. Connect the Drive Computer to Nav Computer RS-232 cable between the Drive Computer connector and the Nav X connector on MARRS-1. Connect the GYRAC to Nav Computer RS-232 cable between the GYRAC connector and the Nav L connector on MARRS-1. Connect the external power cable to the GYRAC and turn on the power supplies. Flip the GYRAC power switch to the on position and press the GYRAC computer reset button. Flip the gyro control switch to the slaved mode.

STEP 6: Load and transmit the NAV.HEX file to the robot's navigation computer. This is done by typing "L,1000,1369" to load the file at Nav computer memory address 1000 (HEX). Next type "T filename". This will place the CRT in terminal mode and create an input buffer to store incoming data in disk file filename. Follow this by typing "control shift " then "control T" and "NAV.HEX" to transmit the program file to the Nav computer. Reply with Yes when asked for time delays. The program data will again be displayed as it is transmitted. If an error is made in STEP 6, the navigation reset button must be pressed and the entire step done over.

STEP 7: Begin program execution. First type "control shift " and then "control Y" to open the input data buffer. Now type "G,1000" to begin program execution. During execution time the robot will send to the H89 two bytes of heading data each time it considers a course change. This data will be displayed on the CRT and stored in the input buffer.

STEP 8: When the robot run is completed (i.e. you have manually stopped it with the MARRS-1 system reset button) the data stored in the input buffer may be written to disk. To do this press the Nav Computer reset button on MARRS-1. Next type "control shift " followed by "control E". Now type "WRT" to save the data to disk ("del" may also be typed to dump buffered data). If additional runs are required continue with STEP 6 and press all three reset buttons on MARRS-1.

STEP 9: Shutdown all systems. To exit M72, type "CPM". Remove both disks from the drives and turn of the power to the H89 system. Turn off power to the robot, GYRAC, and external power supplies.

NOTE: All references to "control" and "shift" in the H89 command lines refer to the control and shift keys and not the words control and shift. Care must be taken to ensure the various cables to MARRS-1 do not become tangled during movement. In addition, it is assumed that the robot has been "pointed" to the desired initial heading before movement commences. The actual direction of travel is set into the NAV.A program at assembly time. Also, the NAV.HEX program must be loaded each time a run is attempted, since the program is cleared on navigation computer reset.

APPENDIX I

CONVERT.BAS PROGRAM LISTING

```
REM*********************************************************
REM*                                                       *
REM*   DATE:  30SEP85                                      *
REM*   VERSION: 1.0                                        *
REM*   TITLE:  CONVERT                                     *
REM*   FILENAME: CONVERT.BAS                               *
REM*   AUTHOR:  CAPT ROLAND J. BLOOM                       *
REM*   PROJECT:  GYRO AND ACCELEROMETER BASED NAVIGATION   *
REM*             SYSTEM FOR A MOBILE AUTONOMOUS ROBOT      *
REM*             (THESIS)                                  *
REM*   OPERATING SYSTEM:  Z89/Z90 CP/M V2.242 MAGNOLIA     *
REM*                      MICROSYSTEM  1982                *
REM*   LANGUAGE:  MBASIC                                   *
REM*   USE:  This program is used to convert raw hex-      *
REM*         adecimal data obtained from the NAV computer  *
REM*         into integer data.  The whole data string     *
REM*         is read and converted to integer format.      *
REM*         The program interactively asks for the name   *
REM*         of the hex data file and asks for the name    *
REM*         of the file where the integer data is to be   *
REM*         stored.                                       *
REM*                                                       *
REM*********************************************************
REM*                                                       *
REM*                    MAIN   PROGRAM                     *
REM*                                                       *
REM*********************************************************
REM*
10 PRINT "INPUT THE NAME OF THE HEX DATA FILE TO CONVERT"
20 INPUT "INCLUDE THE DISK DRIVE AND ENCLOSE IN QUOTES",READFILE$
30 PRINT " "
40 PRINT "INPUT THE NAME OF THE FILE TO STORE THE INTEGER DATA"
50 INPUT "INCLUDE THE DISK DRIVE AND ENCLOSE IN QUOTES",PRINTFILE$
60 OPEN "I",#1,READFILE$
70 OPEN "O",#2,PRINTFILE$
REM*
REM*  A DATA  STRING IS READ AND THEN EACH DATA SEGMENT IS
REM*  CONVERTED AND STORED ON DISK
REM*
80 INPUT#1,DATALINE$
90 IF EOF(1)  THEN END
100 WORD$ = MID$(DATALINE$,3,4)
110 GOSUB 300
120 PRINT#2,VALUE%,",";
130 FOR I% = 11  TO 26 STEP 5
140 WORD$ = MID$(DATALINE$,I%,4)
150 GOSUB  300
160 GOSUB  450
```

```
170 NEXT I%
180 FOR I% = 40  TO 47 STEP 7
190 WORD$ = MID$(DATALINE$,I%,4)
200 GOSUB  300
210 GOSUB  450
220 NEXT I%
230 GOTO  80
REM*********************************************************
REM*                                                       *
REM*              SUBROUTINES   FOLLOW                     *
REM*                                                       *
REM*********************************************************
REM*
REM*   THIS SUBROUTINE CONVERTS THE HEX VALUE TO INTEGER
REM*
300 VALUE% = 0
310 FOR J = 2 TO 4
320 CHAR$ = MID$(WORD$,J%,1)
330 DIGIT = VAL(CHAR$)
340 IF CHAR$ = "A"  THEN  DIGIT = 10
350 IF CHAR$ = "B"  THEN  DIGIT = 11
360 IF CHAR$ = "C"  THEN  DIGIT = 12
370 IF CHAR$ = "D"  THEN  DIGIT = 13
380 IF CHAR$ = "E"  THEN  DIGIT = 14
390 IF CHAR$ = "F"  THEN  DIGIT = 15
400 VALUE% = VALUE% * 16 + DIGIT
410 NEXT J%
420 RETURN
REM*
REM*  THIS SUBROUTINE STORES THE INTEGER VALUES ON DISK
REM*
450 IF I% = 47 GOTO 480
460 PRINT#2,VALUE%,",";
470 GOTO 490
480 PRINT#2,VALUE%
490 RETURN
```

POSITION.BAS PROGRAM LISTING

```
REM*************************************************************
REM*                                                           *
REM*   DATE:  30SEP85                                          *
REM*   VERSION: 1.0                                            *
REM*   TITLE:  POSITION                                        *
REM*   FILENAME: POSITION.BAS                                  *
REM*   AUTHOR:  CAPT ROLAND J. BLOOM                           *
REM*   PROJECT:  GYRO AND ACCELEROMETER BASED NAVIGATION       *
REM*             SYSTEM FOR A MOBILE AUTONOMOUS ROBOT          *
REM*             (THESIS)                                      *
REM*   OPERATING SYSTEM:  Z89/Z90 CP/M V2.242 MAGNOLIA         *
REM*                      MICROSYSTEM  1982                    *
REM*   LANGUAGE:  MBASIC                                       *
REM*   USE:  This program is used to compute the position      *
REM*         of the MARRS-1 robot based on heading and         *
REM*         velocity data from the GYRAC.  The raw data       *
REM*         from the NAV computer (which gathers the          *
REM*         GYRAC data) must first be converted to            *
REM*         integer format by the CONVERT program.            *
REM*         The computed position will be in terms of         *
REM*         x and y coordinates.  An initial (x,y)            *
REM*         position is provided to the program               *
REM*         interactively.  Program output is sent            *
REM*         to the printer where time, x-coordinate           *
REM*         and y-coordinate are printed.                     *
REM*                                                           *
REM*************************************************************
REM*                                                           *
REM*                 DEFINITION OF VARIABLES                   *
REM*                                                           *
REM**************                             ***************
REM*                                                           *
REM*            X = X-COORDINATE                               *
REM*            Y = Y-COORDINATE                               *
REM*       DELTAX = INCREMENT OF MOVEMENT IN X-DIRECTION       *
REM*       DELTAY = INCREMENT OF MOVEMENT IN Y-DIRECTION       *
REM*     DISTANCE = LINEAR DISTANCE TRAVELLED IN T SECONDS     *
REM*            T = 0.1 SECONDS (WHICH IS THE SAMPLE TIME)     *
REM*      HEADING = THE HEADING OF THE ROBOT IN DEGREES        *
REM*     VELOCITY = THE VELOCITY OF THE ROBOT (FT/SEC)         *
REM*       WEIGHT = WEIGHT OF EACH BIT OF VELOCITY,            *
REM*                1024 BITS REPRESENT 10 VOLTS               *
REM*                THEREFORE WEIGHT = 0.00977 VOLTS/BIT       *
REM*         CONV = CONVERSION FACTOR FOR CONVERTING THE       *
REM*                VELOCITY MEASUREMENT FROM VOLTS TO         *
REM*                FT/SEC.  THIS VALUE IS BASED ON LOCAL      *
REM*                ACCELERATION DUE TO GRAVITY OF 32.174      *
REM*                FT/S/S/G AND THE SENSITIVITY OF THE        *
REM*                ACCELEROMETER (VOLTS/G).                   *
```

```
REM*           GF = GAIN FACTOR. THIS IS THE GAIN IN THE   *
REM*                INTEGRATOR CIRCUIT.                    *
REM*         TIME = TIME OF MEASUREMENT (SECONDS)          *
REM*      RAWTIME = INTEGER VALUE OF TIME. THIS VALUE IS   *
REM*                A FACTOR OF 10 TIMES THE REAL TIME.    *
REM*       RAWVEL = INTEGER VALUE FOR VELOCITY (BITS)      *
REM*      RAWHEAD = INTEGER VALUE FOR HEADING (BITS)       *
REM*      LEFTREV = REVERSE WHEEL COUNTS FROM OPTICAL      *
REM*                SHAFT ENCODER ON LEFT REAR WHEEL.      *
REM*      LEFTFOR = FORWARD WHEEL COUNTS FROM OPTICAL      *
REM*                SHAFT ENCODER ON LEFT REAR WHEEL.      *
REM*     RIGHTREV = REVERSE WHEEL COUNTS FROM OPTICAL      *
REM*                SHAFT ENCODER ON RIGHT REAR WHEEL.     *
REM*     RIGHTFOR = FORWARD WHEEL COUNTS FROM OPTICAL      *
REM*                SHAFT ENCODER ON RIGHT REAR WHEEL.     *
REM*                                                       *
REM*     NOTE - WHEEL COUNTS ARE NOT USED BY THIS          *
REM*            PROGRAM BUT COULD BE INCORPORATED          *
REM*            TO PROVIDE A SEPARATE POSITION             *
REM*            CALCULATION.                               *
REM*                                                       *
REM*********************************************************
REM*                                                       *
REM*                  MAIN PROGRAM FOLLOWS                 *
REM*                                                       *
REM*********************************************************
REM*
10 INPUT "INPUT THE NAME OF THE DATA FILE(INCLUDE DISK DRIVE)",
    FILE$
20 OPEN "I",#1,FILE$
30 WEIGHT = 0.00977
33 CONV = 32.174/0.6
37 GF = 1/19.7
39 T = 0.1
40 INPUT "INPUT THE INITIAL POSITION (X,Y) IN FEET,XO,YO
50 X = XO
60 Y = YO
70 INPUT "INPUT THE TEST DESIGNATION",TEST$
80 LPRINT TEST$
90 LPRINT " "
100 LPRINT "                       POSITION"
110 LPRINT " TIME(SEC)       X(FT)          Y(FT)"
120 LPRINT "***********    ********        *******
130 LPRINT " "
140 INPUT#1,RAWTIME,LEFTREV,LEFTFOR,RIGHTREV,RIGHTFOR,RAWVEL,
     RAWHEAD
150 IF EOF(1) THEN END
```

POSITION.BAS PROGRAM LISTING (continued)

```
160 TIME = RAWTIME * 0.1
165 PRINT TIME
170 VELOCITY = (RAWVEL - 512) * WEIGHT * CONV * GF
180 HEADING = RAWHEAD * 0.001534
190 DISTANCE = VELOCITY * T
200 DELTAX = DISTANCE * COS(HEADING)
210 DELTAY = DISTANCE * SIN(HEADING)
220 X = X + DELTAX
230 Y = Y + DELTAY
240 LPRINT USING "    ###.#        ##.##        ##.##";TIME,X,Y
250 GOTO 140
```

APPENDIX J

GYRAC Phase II Sample Test Data

TEST #2B (SENSITIVITY = 0.393)

TIME(SEC)	POSITION X(FT)	POSITION Y(FT)
0.0	6.75	4.58
0.1	6.75	4.58
0.2	6.75	4.58
0.3	6.75	4.59
0.4	6.75	4.59
0.5	6.75	4.59
0.6	6.75	4.59
0.7	6.75	4.59
0.8	6.75	4.60
0.9	6.75	4.60
1.0	6.75	4.60
1.1	6.75	4.60
1.2	6.75	4.60
1.3	6.75	4.61
1.4	6.75	4.61
1.5	6.75	4.61
1.6	6.75	4.61
1.7	6.75	4.62
1.8	6.75	4.62
1.9	6.75	4.63
2.0	6.75	4.64
2.1	6.75	4.65
2.2	6.75	4.66
2.3	6.74	4.67
2.4	6.74	4.68
2.5	6.74	4.69
2.6	6.74	4.70
2.7	6.74	4.70
2.8	6.74	4.71
2.9	6.74	4.72
3.0	6.74	4.73
3.1	6.74	4.74
3.2	6.74	4.75
3.3	6.74	4.76
3.4	6.74	4.77
3.5	6.74	4.78
3.6	6.74	4.79
3.7	6.73	4.80
3.8	6.73	4.81
3.9	6.73	4.82
4.0	6.73	4.83
4.1	6.73	4.84
4.2	6.73	4.85

APPENDIX K

Phase III Test Data

GYRO NAVIGATION TEST RUN NUMBER 1

Given: 33 Foot Course
Heading of 3EC (hex) = 1004 (integer) = 88.2421875 (degrees)

Steering Window:			
	3E0	to	3EF (Hex)
	992	to	1007 (integer)
	87.1875	to	88.50585938 (Degrees)

Measured: Heading at each course change decision point

HEADING (Hex)	HEADING (Integer)	HEADING (Degrees)	DEVIATION (Integer)	DEVIATION (Degrees)
3EB	1003	88.15429688	- 1	-0.087890625
3EB	1003	88.15429688	- 1	-0.087890625
3EB	1003	88.15429688	- 1	-0.087890625
401	1025	90.08789063	+21	+1.845703125
410	1040	91.40625	+36	+3.1640625
40F	1039	91.31835938	+35	+3.076171875
3FF	1023	89.91210938	+19	+1.669921875
3D8	984	86.484375	-20	-1.7578125
3B0	944	82.96875	-60	-5.2734375
3A2	930	81.73828125	-74	-6.50390625
3A2	930	81.73828125	-74	-6.50390625
3B8	952	83.671875	-52	-4.5703125
3E0	992	87.1875	-12	-1.0546875
3FF	1023	89.91210938	+19	+1.669921875
40D	1037	91.14257813	+33	+2.900390625
410	1040	91.40625	+36	+3.1640625
3FC	1020	89.6484375	+16	+1.40625
3DB	987	86.74804688	-17	-1.494140625
3B9	953	83.75976563	-51	-4.482421875
3AE	942	82.79296875	-62	-5.44921875
3AA	938	82.44140625	-66	-5.80078125
3BA	954	83.84765625	-50	-4.39453125
3E2	994	87.36328125	-10	-0.87890625
3FF	1023	89.91210938	+19	+1.669921875
40A	1034	90.87890625	+30	+2.63671875

GYRO NAVIGATION TEST RUN NUMBER 2

Given: 33 Foot Course
Heading of 3EC (hex) = 1004 (integer) = 88.2421875 (degrees)

Steering Window:			
3E0	to	3EF (Hex)
992	to	1007 (Integer)
87.1875	to	88.50585938 (Degrees)

Measured: Heading at each course change decision point

HEADING (Hex)	HEADING (Integer)	HEADING (Degrees)	DEVIATION (Integer)	DEVIATION (Degrees)
2F5	757	66.53320313	-247	-21.70898438
3EB	1003	88.15429688	- 1	- 0.087890625
3EB	1003	88.15429688	- 1	- 0.087890625
3EB	1003	88.15429688	- 1	- 0.087890625
3EB	1003	88.15429688	- 1	- 0.087890625
3EB	1003	88.15429688	- 1	- 0.087890625
3EB	1003	88.15429688	- 1	- 0.087890625
3EB	1003	88.15429688	- 1	- 0.087890625
3E7	999	87.80273438	- 5	- 0.439454125
3E7	999	87.80273438	- 5	- 0.439454125
3E7	999	87.80273438	- 5	- 0.439454125
3E4	996	87.5390625	- 8	- 0.703125
3E1	993	87.27539063	- 11	- 0.966796875
3DE	990	87.01171875	- 14	- 1.23046875
3E2	994	37.36328125	- 10	- 0.87890625
3E8	1000	87.890625	- 4	- 0.3515625
3E8	1000	87.890625	- 4	- 0.3515625
3E8	1000	87.890625	- 4	- 0.3515625
3E8	1000	87.890625	- 4	- 0.3515625
3E8	1000	87.890625	- 4	- 0.3515625
3E8	1000	87.890625	- 4	- 0.3515625
3E8	1000	87.890625	- 4	- 0.3515625
3E8	1000	87.890625	- 4	- 0.3515625
3E8	1000	87.890625	- 4	- 0.3515625
3E8	1000	87.890625	- 4	- 0.3515625
3E7	999	87.80273438	- 5	- 0.439454125
3E7	999	87.30273438	- 5	- 0.439454125

GYRO NAVIGATION TEST RUN NUMBER 3

Given: 33 Foot Course
Heading of 3EC (hex) = 1004 (integer) = 88.2421875 (degrees)

Steering Window:	3e0	to	3EF (Hex)
	992	to	1007 (Integer)
	87.1875	to	88.50585938 (Degrees)

Measured: Heading at each course change decision point

HEADING (Hex)	HeADING (Integer)	HEADING (Degrees)	DEVIATION (Integer)	DEVIATION (Degrees)
502	1282	112.6757813	+278	+24.43359375
3E4	996	87.5390625	- 8	- 0.703125
3E1	993	87.27539063	- 11	- 0.966796875
3E1	993	87.27539063	- 11	- 0.966796875
3E5	997	87.62695313	- 7	- 0.615234375
3E8	1000	87.890625	- 4	- 0.3515625
3EE	1006	88.41796875	+ 2	+ 0.17578125
3F4	1012	88.9453125	+ 8	+ 0.703125
3F9	1017	89.38476563	+ 13	+ 1.142578125
3E1	993	87.27539063	- 11	- 0.966796875
3DB	989	86.92382813	- 15	- 1.318359375
3F1	1009	88.68164063	+ 5	+ 0.439453125
3F9	1017	89.38476563	+ 13	+ 1.142578125
3FB	1019	89.56054688	+ 15	+ 1.318359375
3E7	999	87.80273438	- 5	- 0.439454125
3DE	990	87.01171875	- 14	- 1.23046875
3F1	1009	88.68164063	+ 5	+ 0.439453125
3FC	1020	89.6484375	+ 16	+ 1.40625
3FF	1023	89.91210938	+ 19	+ 1.669921875
3EA	1002	88.06640625	- 2	- 0.017578125
3E1	993	87.27539063	- 11	- 0.966796875
3E8	1000	87.890625	- 4	- 0.3515625
3EB	1003	88.15429688	- 1	- 0.087890625
3F1	1009	88.68164063	+ 5	+ 0.439453125
3F4	1012	88.9453125	+ 8	+ 0.703125
3DE	990	87.01171875	- 14	- 1.23046875
3D2	977	85.86914063	- 27	- 2.373046875

APPENDIX L

LAB EQUIPMENT, COMPUTER, HARDWARE, and SOFTWARE

LAB EQUIPMENT

Quantity	Description
1	AFIT Mobile Robotics Laboratory
1	AFIT MARRS-1 Robot
1	1607 Eldorado Frequency Counter
1	186 Wavetek Waveform Generator
1	1610A Hewlitt Packard Logic State Analyzer
1	465M Tektronics Oscilloscope
1	3466A Hewlitt Packard Digital Multimeter
1	M-15 Trygon Power Supply
1	M-36 Trygon Power Supply
1	6C3000 Powertec Power Supply
1	S-10 Bytek EEPROM Programmer
1	S-52 Ultra Violet Products EEPROM Eraser

COMPUTER HARDWARE

Quantity	Description
1	MARRS-1 Navigation Computer
1	MARRS-1 Drive Computer
1	MARRS-1 GYRAC Computer
1	Heath H89 Computer
1	Heath H27 Eight Inch Dual Disk Drive System
1	Heath H125 Dot Matrix Printer

LAB EQUIPMENT, COMPUTER, HARDWARE, and SOFTWARE (continued)

COMPUTER SOFTWARE

Quanity	Description
1	Wordmaster Word Processor
1	Wordstar Word Processor
1	Virtual Devices Robo A 6802 Cross Assembler
1	Modem 720 Communication Program
1	CP/M Operating System
1	MBASIC Compiler
1	MARRS-1 Drive Computer ROM Software
1	MARRS-1 Navigation Computer ROM Software

SECURITY CLASSIFICATION OF THIS PAGE

AD-A164036

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for Public Release;
Distribution Unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
AFIT/GA/GE/ENG/85D-33

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Air Force Institute of Technology

6b. OFFICE SYMBOL *(If applicable)*
EN

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS *(City, State and ZIP Code)*
Wright-Patterson AFB, OH 45433

7b. ADDRESS *(City, State and ZIP Code)*

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State and ZIP Code)*

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

11. TITLE *(Include Security Classification)*
on back

12. PERSONAL AUTHOR(S)
Bloom, Roland J. and Ramey, William J. Jr.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 85 Jan TO 85 Dec	1985/12/2	251

16. SUPPLEMENTARY NOTATION

Approved for public release: IAW AFR 190-1/.
LYNN E. WOLAVER 16 AUG 86
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

17. COSATI CODES

FIELD	GROUP	SUB. GR.
17	07	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Robot, Robotics, Robot Navigation

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

A navigation system for a mobile autonomous robot is presented. The navigation system is based upon a directional gyroscope and a single axis accelerometer which enables a robot to navigate independent of wheel optical shaft encoders and other commonly used positioning apparatus. The computer controlled navigation system is capable of providing absolute heading, heading rate (angular velocity), and linear velocity to a user computer. These data from the navigation system (heading and velocity) are used to compute the present location of the robot. In addition, the heading data is used to form a closed loop feedback control system for maintaining the robot on a desired course. The navigation system was designed

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL
Mathew Kabrisky

22b. TELEPHONE NUMBER *(Include Area Code)*
(513) 255-5276

22c. OFFICE SYMBOL
AFIT/EN

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

Block 11. Gyro and Accelerometer Based Navigation System for a Mobile Autonomous Robot

Block 19. (continued) specifically for application on an existing Air Force Institute of Technology (AFIT) robot; however, it could be easily adapted to any robot system with a standard IEEE RS-232 serial communication interface. Test results are provided which demonstrate the use of closed loop heading control on the AFIT robot and which identify problems associated with the use of an accelerometer system for distance measurement. This thesis includes all schematics, parts lists, software listings, and operating instructions for the navigation system. A new robot world modeling and path planning technique is also presented.

END
FILMED
3-86
DTIC