

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

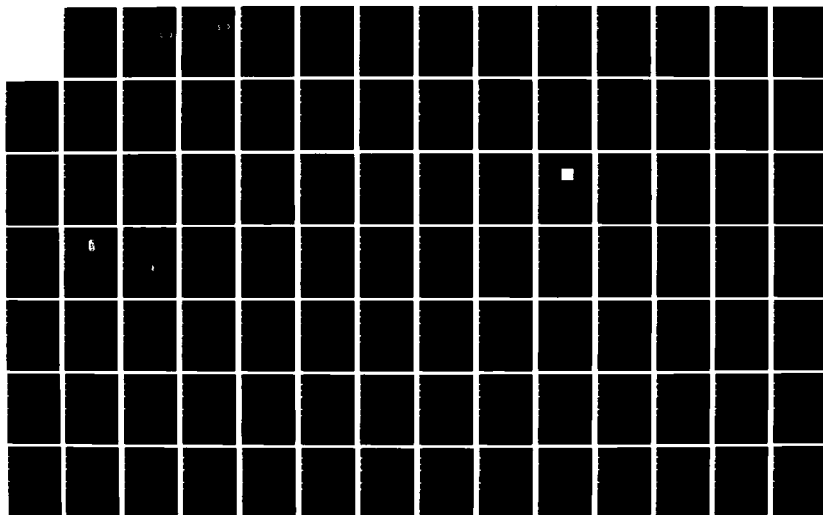
1/3

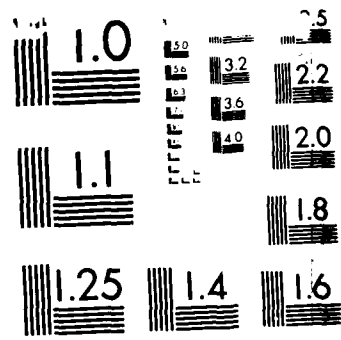
UNCLASSIFIED

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
 ELECTE
 FEB 13 1986

S
 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

NTIC FILE COPY

DISTRIBUTION STATEMENT A

Approved for public release;
 Distribution Unlimited

DEPARTMENT OF THE AIR FORCE
 AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

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

DTIC
ELECTE
FEB 13 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

Approved for public release; distribution unlimited

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.



Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	24

Contents

	Page
Preface	ii
List of Figures	vii
Abstract	x
I. Introduction	1
Background	1
Problem and Scope	12
Assumptions	13
Evolution and Capability of MARRS-1	14
General Approach	16
Sequence of Presentation	17
II. GYRAC Design and Theory of Operation	19
Summary of GYRAC System	19
The Gyro Subsystem	21
The Accelerometer Subsystem	31
GYRAC Computer and Interfacing Subsystem	36
GYRAC System - Theory of Operation	39
III. Integration of the GYRAC System onto MARRS-1	42
Structural	42
Electrical	42
Software	46
IV. General Robot Navigation Theory	49
Assumptions	50
Past Approaches	50
A New Technique	60
Detailed Path Planning	66
Conclusion	73
V. Testing, Analysis, and Results	75
Phase I	75
Phase II	78
Phase III	87
Review of Assumptions	96
Miscellaneous	97

VI.	Summary, Conclusions, and Recommendations	99
	Summary and Conclusions	99
	Recommendations	102
	Bibliography	107
Vita	Captain Bloom	109
	Captain Ramey	110
Appendix A:	SDC1700 S/D Converter	A-2
	AD573 A/D Converter	A-8
	AD544 Operational Amplifier.....	A-15
Appendix B:	Digital Interface	
	Schematic Diagrams	B-2
	Parts List	B-7
	Device Layout	B-10
Appendix C:	Syncro-to-Digital	
	Schematic Diagrams	C-2
	Parts list	C-3
	Device Layout	C-4
Appendix D:	GYRAC Computer	
	Schematic Diagrams	D-2
	Parts List	D-4
	Memory Map	D-7
	Device Layout	D-9
Appendix E:	GYRAC Power Panel Diagram	E-2
	S/D Edge Connector	E-3
	GYRAC Computer Sensor Bus Connector	E-4
	GYRAC Computer Egde Connector	E-5
	Digital Interface Edge Connector	E-6
	H89 to GYRAC RS-232 Cable	E-7
	H89 to NAV X or Drive Computer	
	RS-232 Cable	E-8
	GYRAC to Drive Computer RS-232 Cable	E-9
	Drive Computer to NAV X RS-232 Cable	E-10
	GYRAC System Wiring Harness.....	E-11
Appendix F:	GYRAC.A Program	
	Structure Chart	F-2
	Listing	F-3
	Operating Instructions	F-21
Appendix G:	GTEST.A Program	
	Structure Chart	G-2
	Listing	G-3
	Operating Instructions	G-11

Appendix H:	NAV.A Program	
	Structure Chart	H-2
	Listing	H-5
	MARRS.NAV Program Listing	H-26
	NAV.A Operating Instructions	H-27
Appendix I:	CONVERT Program Listing	I-2
	POSITION Program Listing	I-4
Appendix J:	GYRAC Phase II Sample Test Data	J-2
Appendix K:	GYRAC Phase III Test Data	
	Gyro Navigation Test #1	K-2
	Gyro Navigation Test #2	K-3
	Gyro Navigation Test #3	K-4
Appendix L:	Lab Equipment, Hardware, Software	L-2

List of Figures

	Page
1.1. Effect of Initial Heading Error	9
2.1. GYRAC System Block Diagram	20
2.2. Directional Gyro	22
2.3. Gyro, Exploded View	23
2.4. The Heading Indicator	24
2.5. Stepper Motor Drive Circuit	25
2.6. Indicator, Exploded View	26
2.7. Internal Structure of a Syncro Control Transmitter and its Electrical Representation ..	27
2.8. Flux Detector, Exploded View	29
2.9. Slaving Unit	30
2.10. The QA-1100 Accelerometer	31
2.11. Basic Structure of a Servo-Type Accelerometer ..	32
2.12. QA-1100 Sensor Assembly, Exploded View	34
2.13. Accelerometer Integrator and Scaling Circuit ..	36
3.1. MARRS-1 Computer System	44
4.1. Circular Approximation of Physical Objects	51
4.2. Polygon Approximation to Real World Obstacles ..	52
4.3. Technique of Lozano-Perez for Going Around an Obstacle.....	53
4.4. Examples of How an Obstacle May Be Modeled Using Lozano-Perez Technique	54
4.5. Vertices of All Obstacles are Extended Outward so the Robot can be Treated as a Point.....	55
4.6a. Monaghan's Modeling Technique	56

4.6b.	Inside Corners are Not Used as "Way Points"	57
4.7.	Generalized Cones Form Freeways Between Obstacles	58
4.8.	Convex Regions Separated by Doorways Represent Free Space	59
4.9.	Any Two Points, P and Q, In or On a Convex Polygon May be Connected by an Unobstructed Straight Line	59
4.10.	Room with Two Obstacles	61
4.11.	Free Space is Divided into Convex Regions to Define "Safe Points"	62
4.12.	Only Obstacles and Safe Points are Modeled	62
4.13.	Indirect Pathways Pass Through Safe Points	63
4.14.	Problems Occur if Free Space Regions are Not Chosen Correctly	64
4.15.	Free Space Regions Can be Chosen to Minimize the Probability of Collision	64
4.16.	Modeling of a Room with One Obstacle as a Set of Points	67
4.17	Line Connecting Two Points can be Represented Through Parametric Equations	68
4.18	Line Connecting Two Arbitrary Points	69
4.19	Two Safe Points can be Reached Through a Direct Path From the Robots Present Position ...	71
4.20	Two Safe Point Paths Lead to the Goal	72
5.1	Cartesian Coordinate System is Centered in Corner of Test Area	80
5.2	GYRAC Computed Position Test Using Accelerometer Sensitivity of 2v/g	81
5.3	GYRAC Computed Position Test Using Accelerometer Sensitivity of 0.393v/g	83

5.4	GYRAC Computed Position Test Using Accelerometer Sensitivity of 0.6v/g	84
5.5	Acceleration Test #1	86
5.6	GYRAC Navigation Test #1, Automatic Control of MARRS-1 Heading	93
5.7	GYRAC Navigation Test #2, Automatic Control of MARRS-1 Heading	94
5.8	GYRAC Navigation Test #3, Automatic Control of MARRS-1 Heading	95

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 it 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

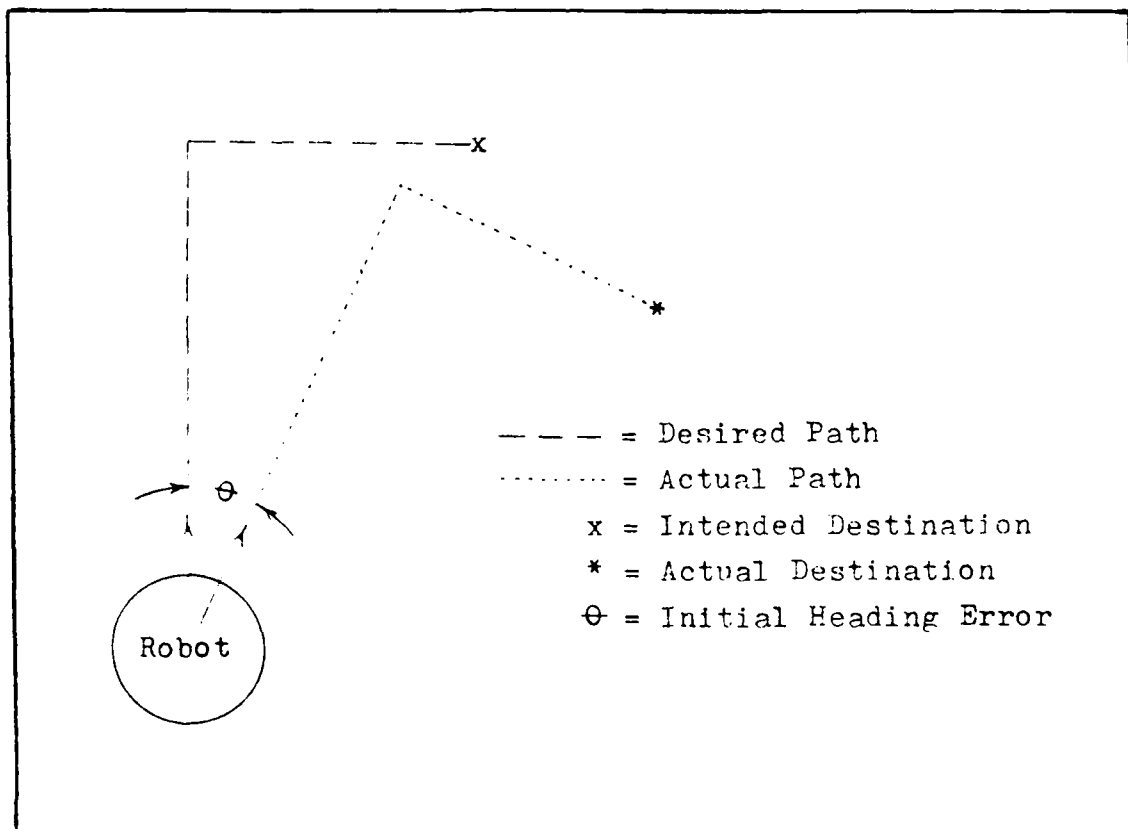


Figure 1.1. Effect of initial heading error.

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 earth's 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 programmed 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 referred 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

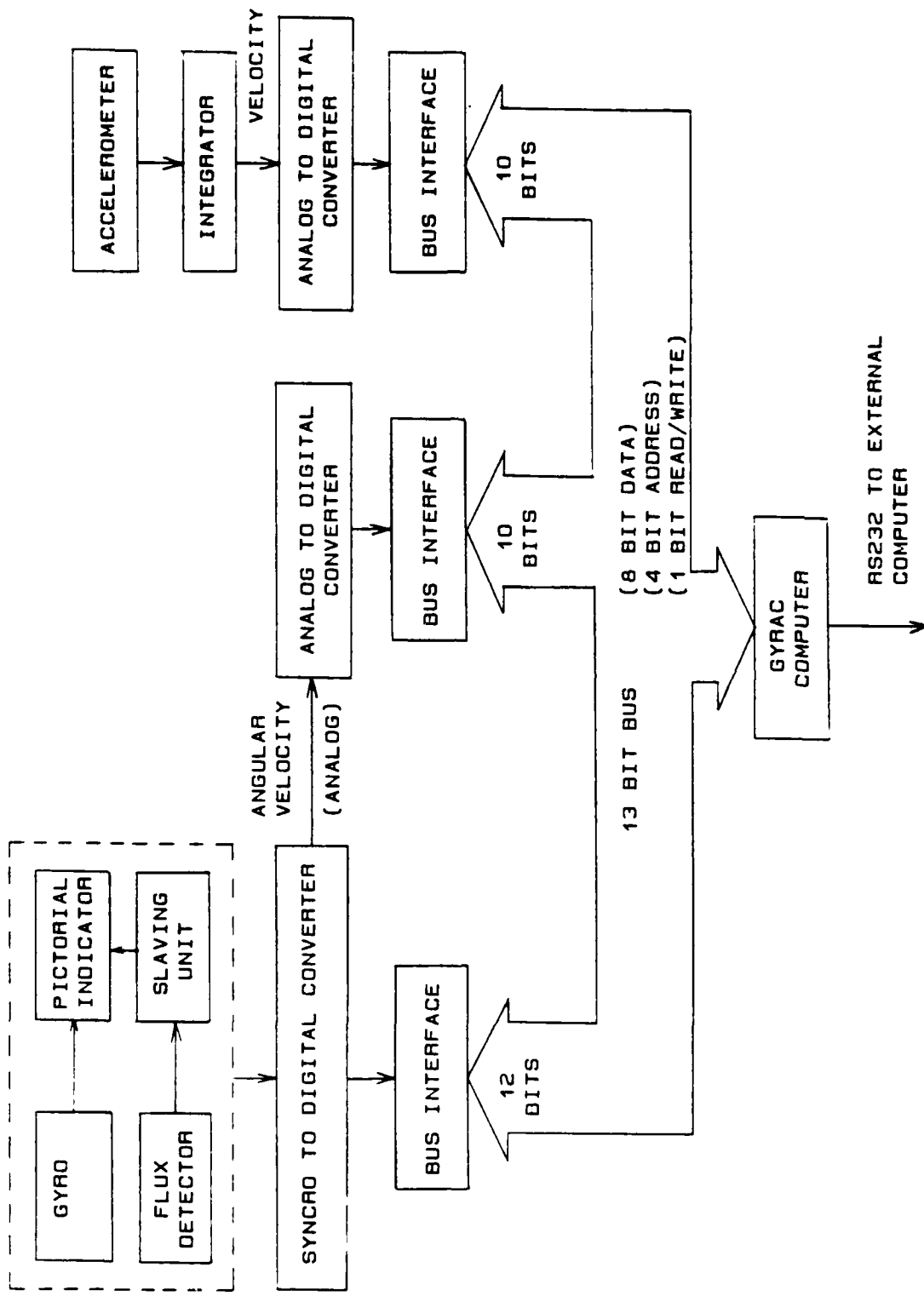


Figure 2.1. GYRAC System Block Diagram

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

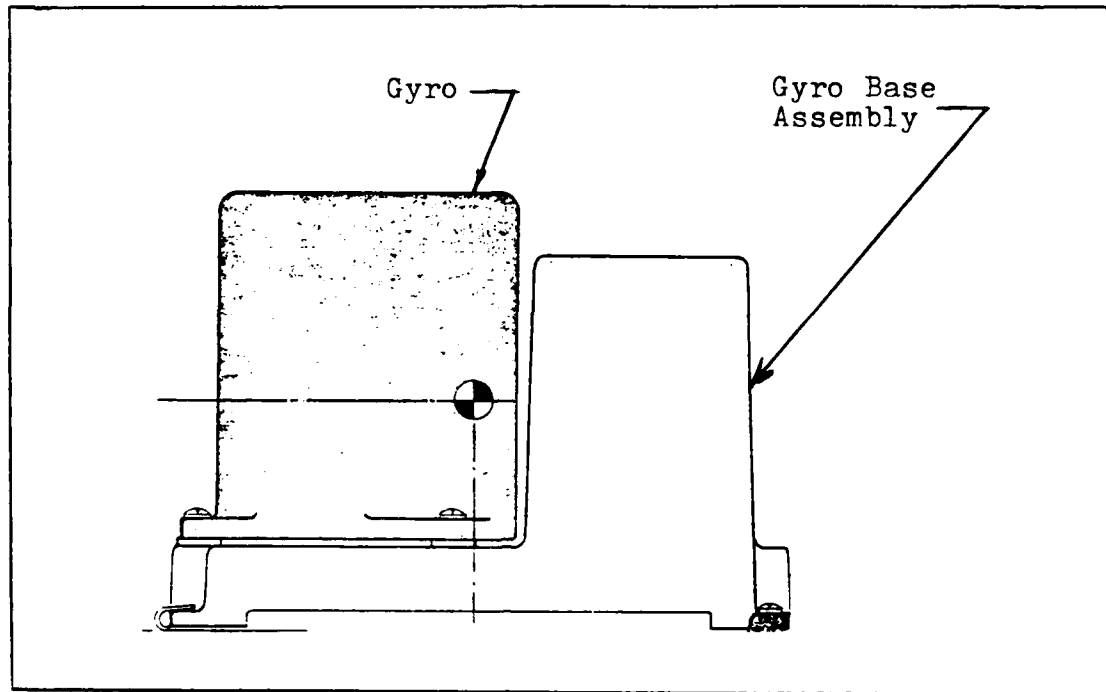


Figure 2.2. Directional Gyro [1:2-11]

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 synchro excitation; + and - 15 VDC regulated supply for the analog circuitry in the system; +15 VDC unregulated voltage for the stepper motor in the indicator; and +5 VDC regulated supply for the system digital logic circuitry [3:5-33].

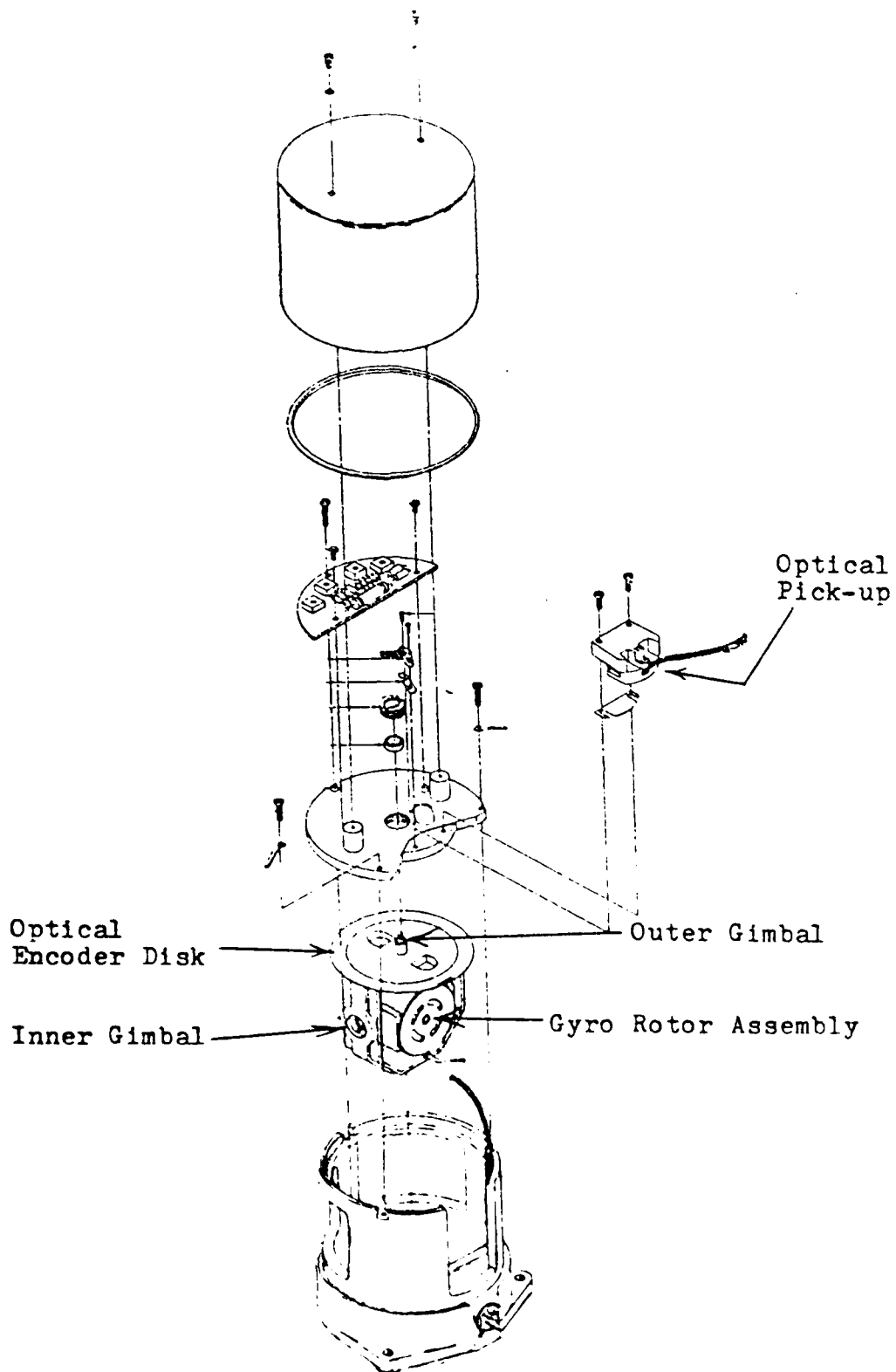


Figure 2.3. Gyro, Exploded View [2:5-33].

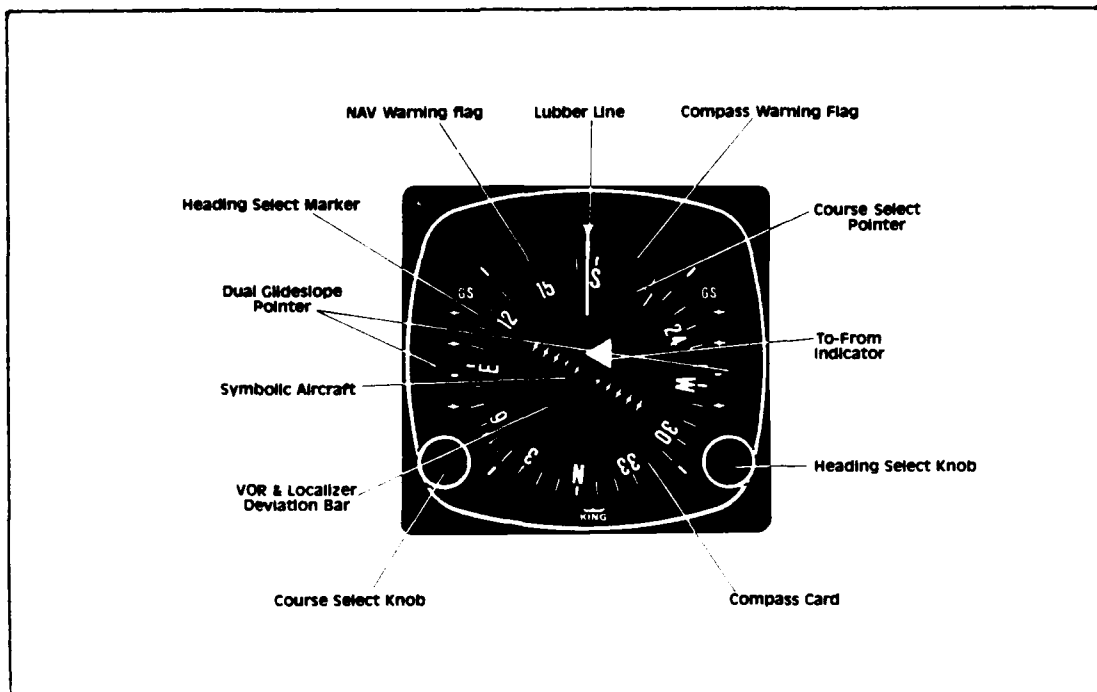


Figure 2.4. The Heading Indicator [1:2-9]

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

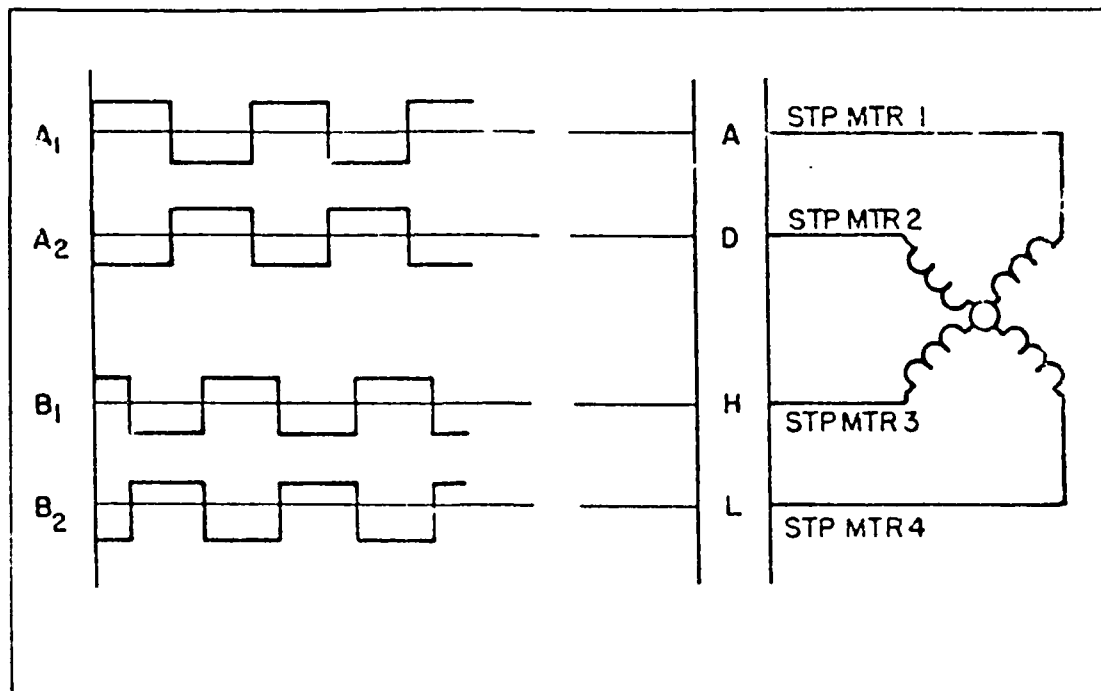


Figure 2.5. Stepper Motor Drive Circuit [2:4-1]

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 synchro 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

Syncro Control Transmitter (CX)

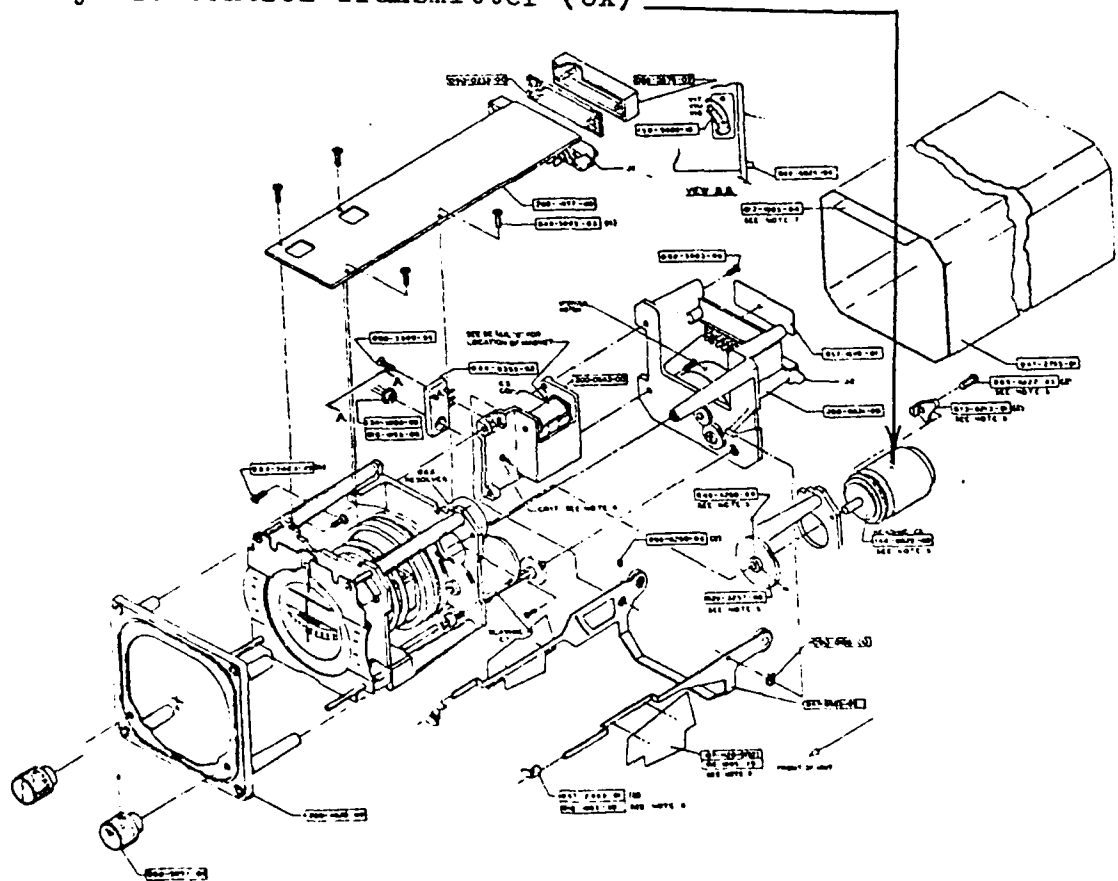


Figure 2.6. Indicator, Exploded View [2:5-5].

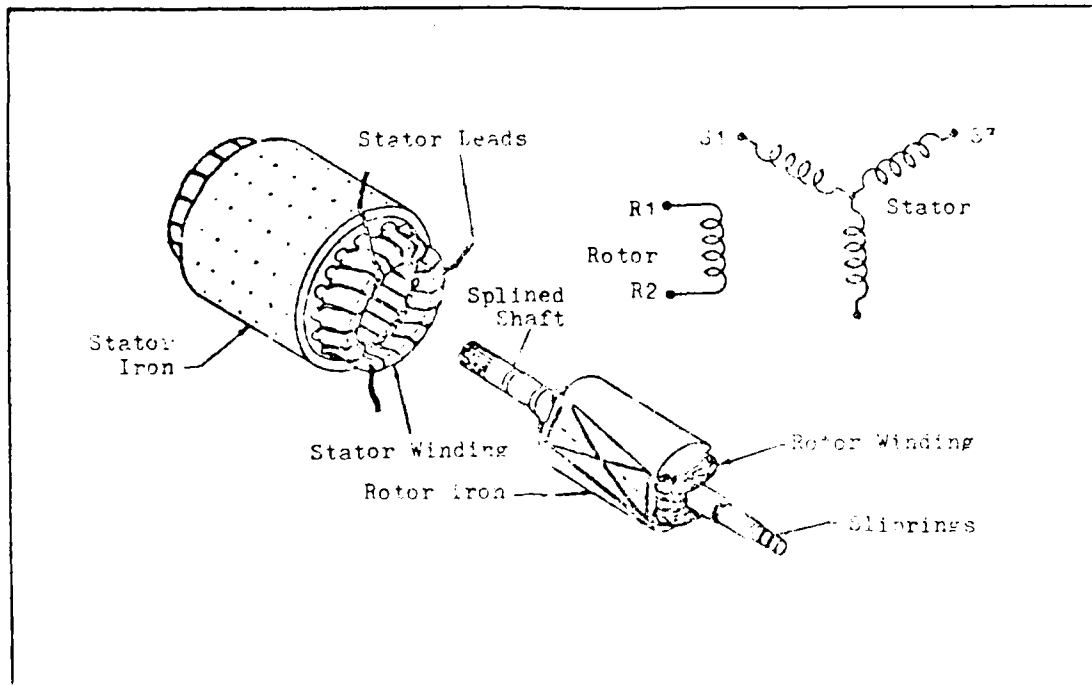


Figure 2.7. Internal Structure of a Syncro Control Transmitter and its electrical representation. [18:2]

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 θ . 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:

$$A \sin(\omega t)$$

Then the voltages which will appear across the stator

terminals will be:

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

$$S3 \text{ to } S2 = A \sin(\omega t) \sin(\theta + 120^\circ)$$

$$S2 \text{ to } S1 = A \sin(\omega t) \sin(\theta + 240^\circ)$$

where θ is the shaft angle.

These voltages are known as synchro format voltages [18:2]. A desirable result of the synchro output is that it can be easily converted to a digital signal with a standard synchro-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 synchro 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

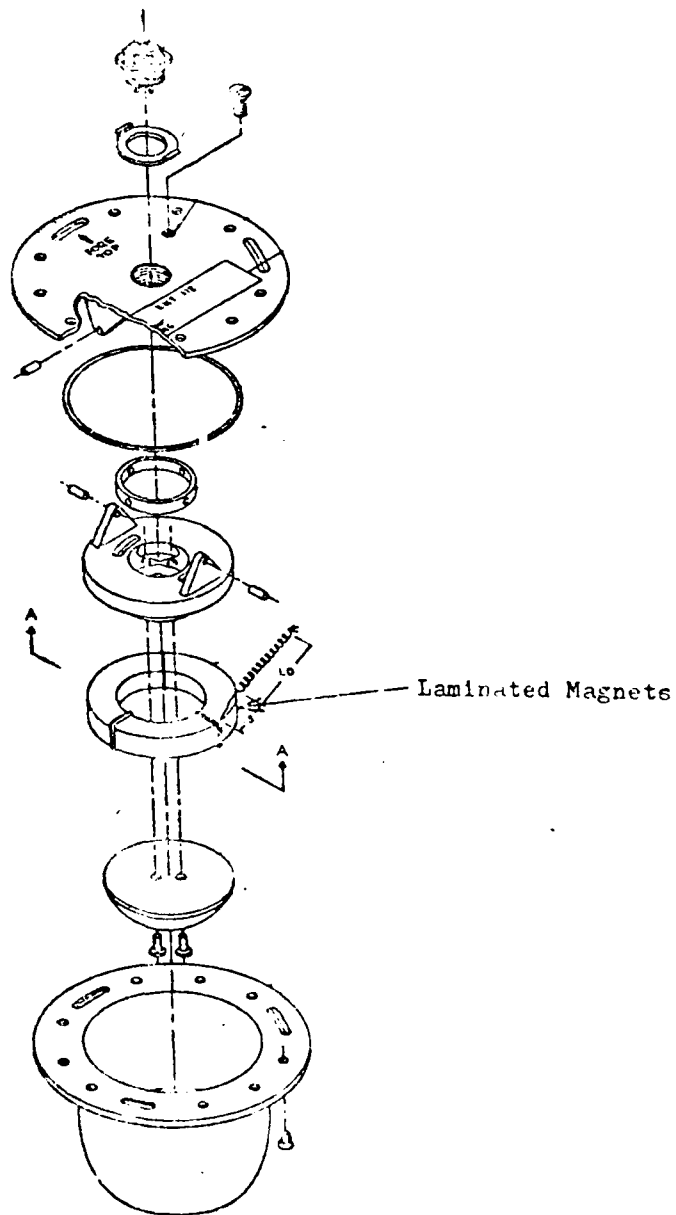


Figure 2.8. Flux Detector, Exploded View [4:5-3] .

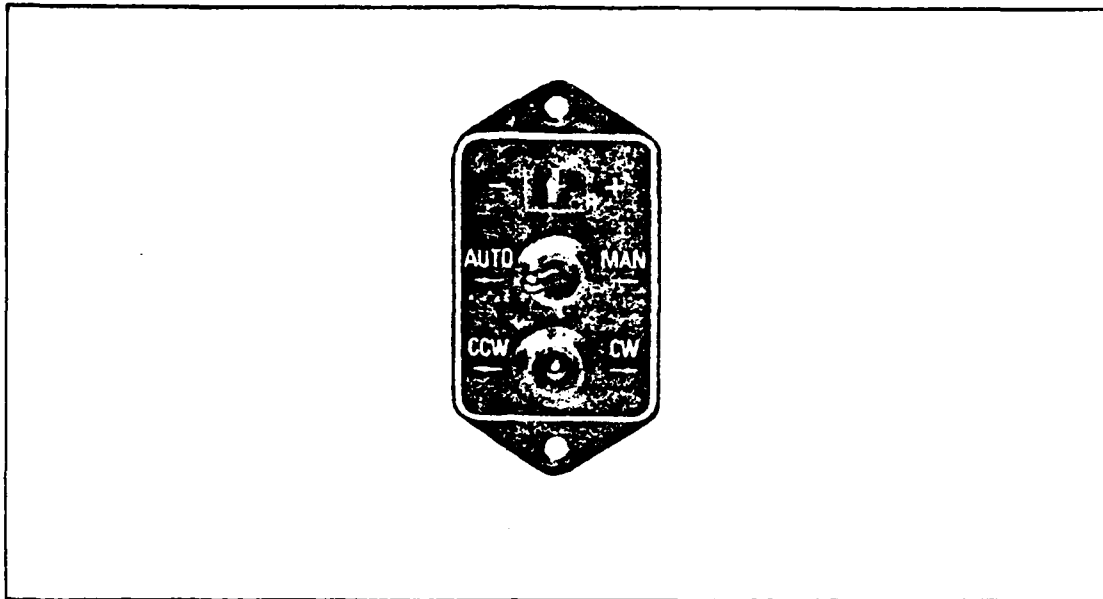


Figure 2.9. Slaving Unit [1:2-19]

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 nearby 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

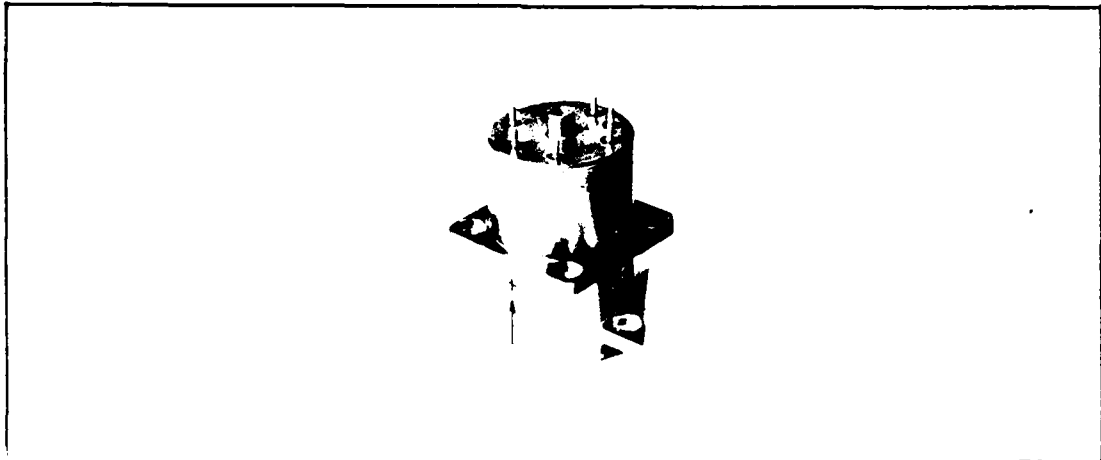


Figure 2.10. The QA-1100 Accelerometer [20]

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

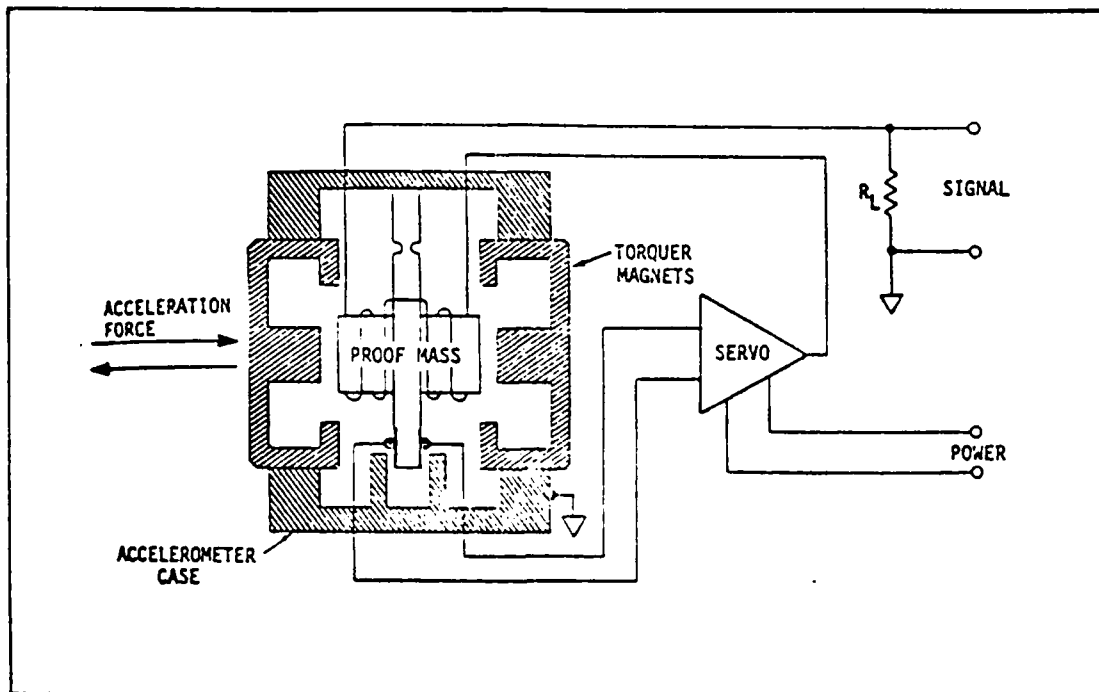


Figure 2.11. Basic Structure of a Servo-Type Accelerometer [7:1-3]

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

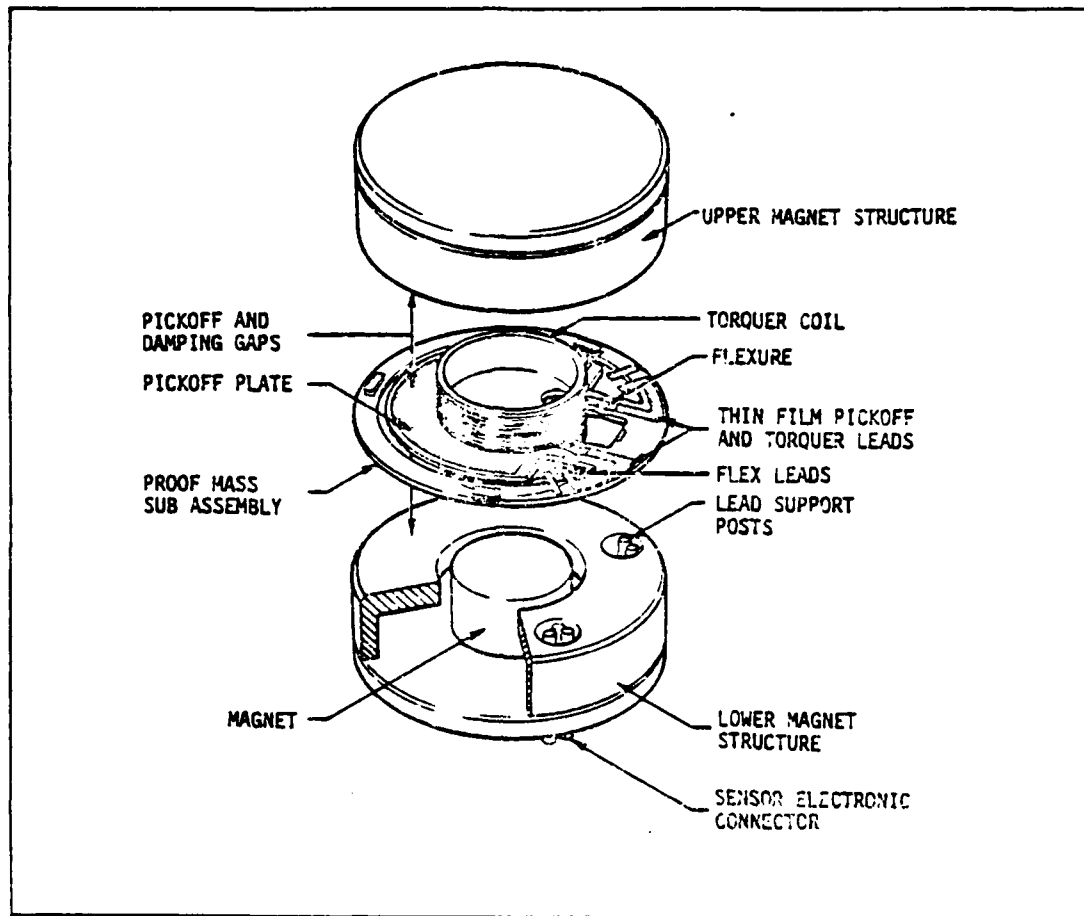


Figure 2.12. QA-1100 Sensor Assembly, Exploded View [7:1-4]

sensitivity of 2 volts/g. R consists of 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, referred 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 millivolts) 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 Assumptions.

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

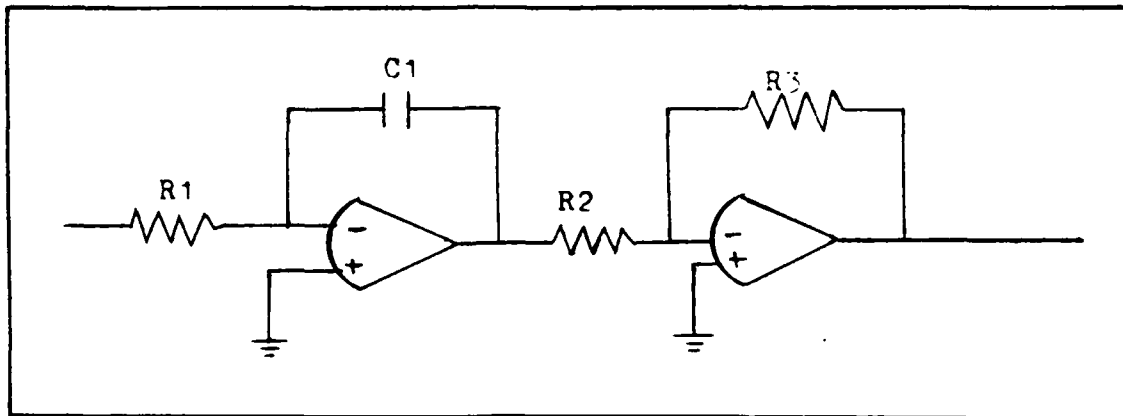


Figure 2.13. Accelerometer Integrator and Scaling Circuit

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 hexadecimal 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 hexadecimal. 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.

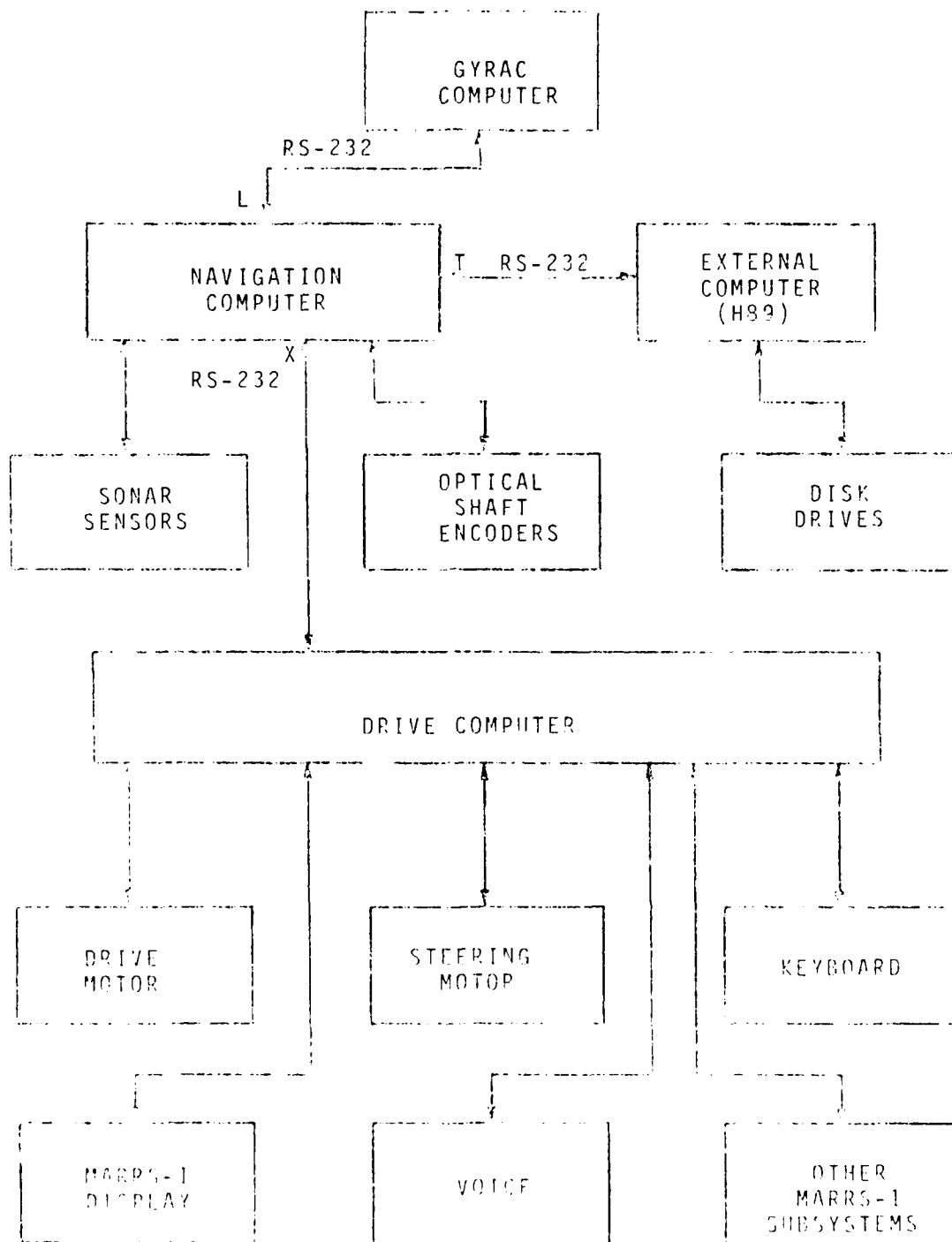


Figure 3.1 MARRS-1 Computer System

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 robot's 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.

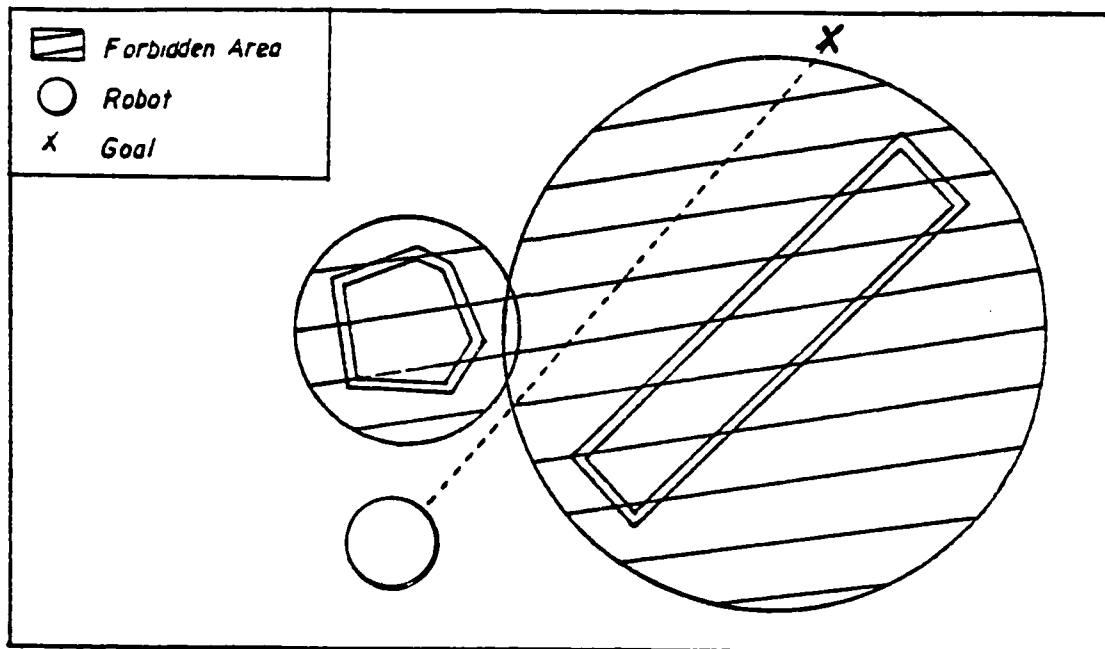


Figure 4.1. Circular approximations of physical objects [16:24]

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.

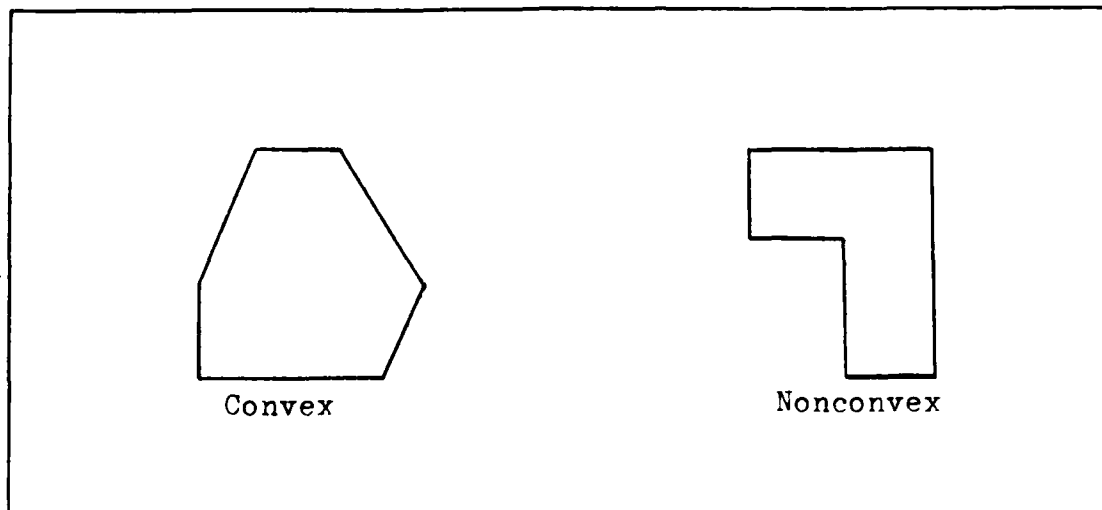


Figure 4.2. Polygon approximations to real world obstacles

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).

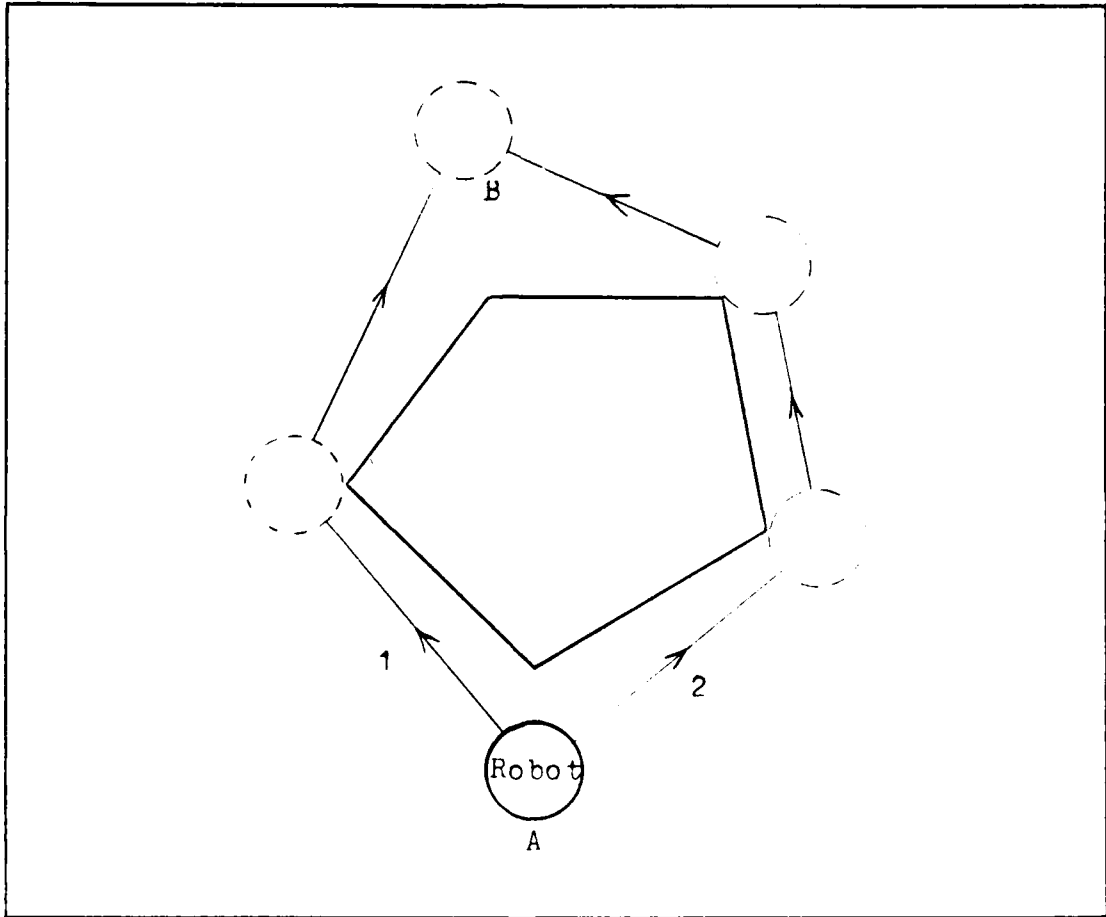


Figure 4.3. Technique of Lozano-Perez for going around an obstacle.

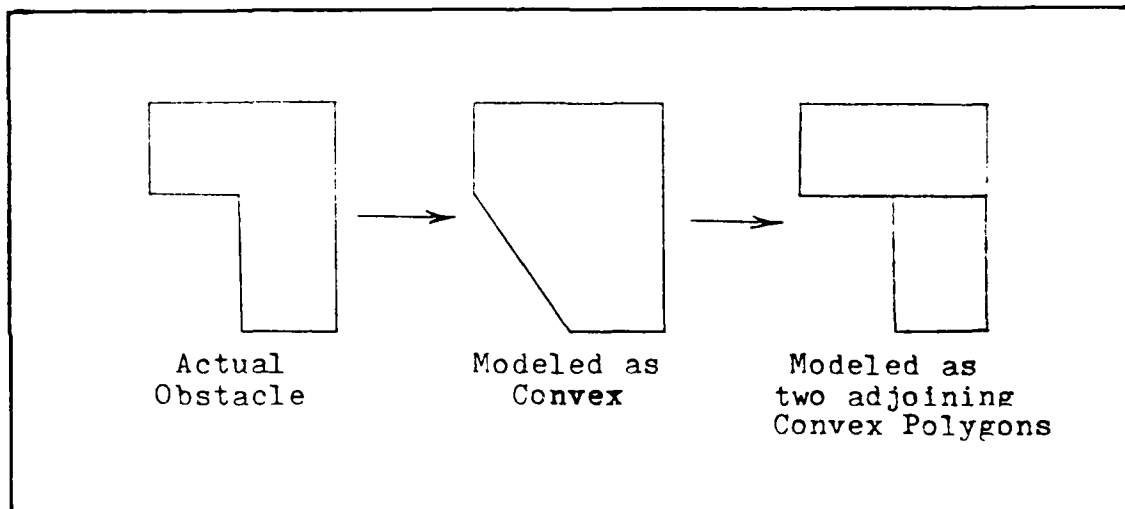


Figure 4.4. Examples of how an obstacle may be modeled using Lozano-Perez technique.

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 wasteful 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.

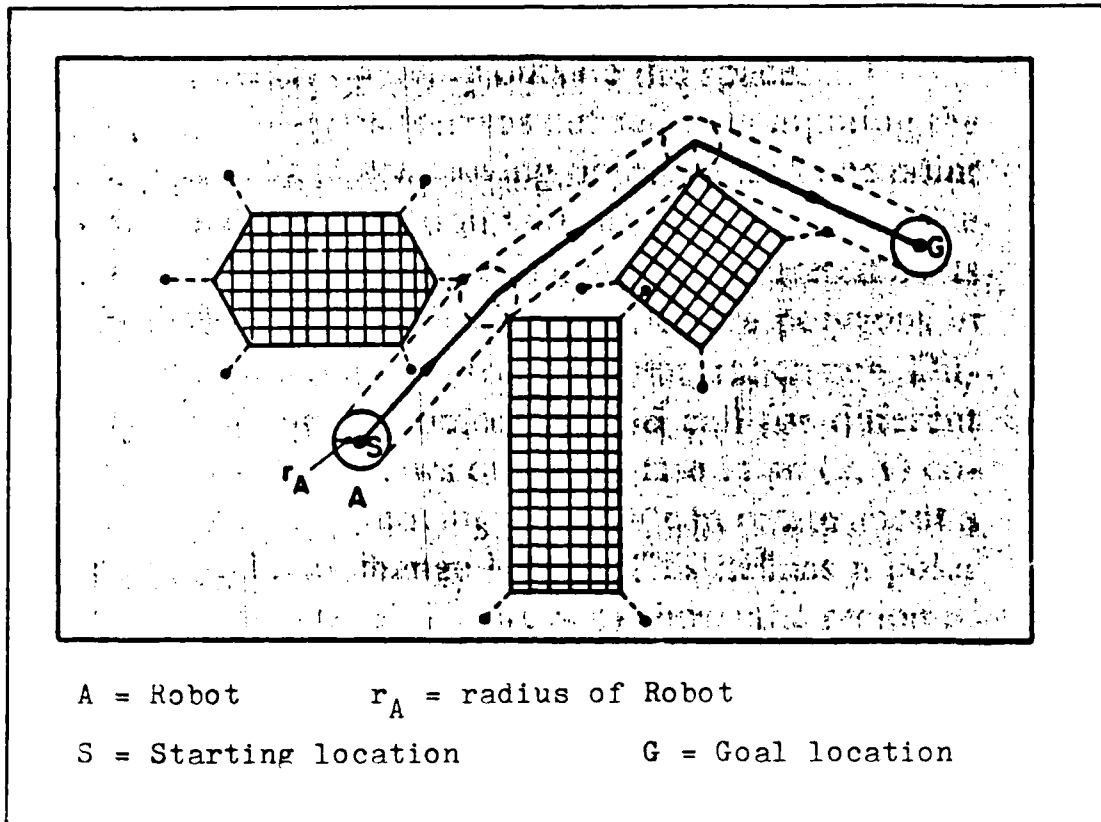


Figure 4.5. Vertices of all obstacles are extended outward so the robot can be treated as a point. [15:562]

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

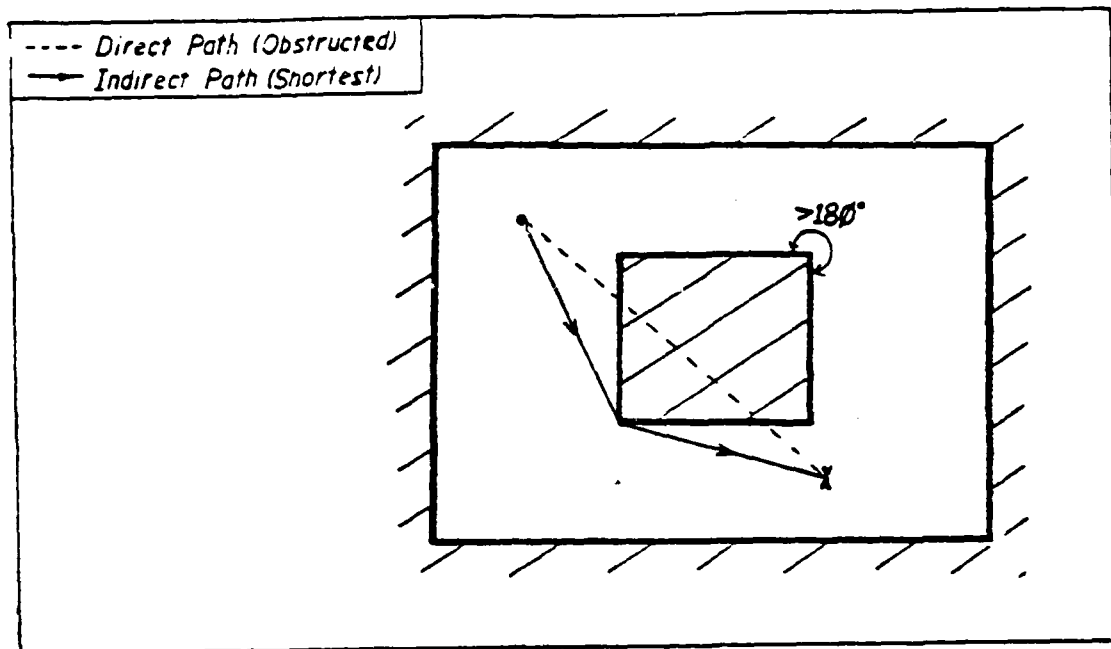


Figure 4.6a. Monaghan's modeling technique [16:47]

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 vertex 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

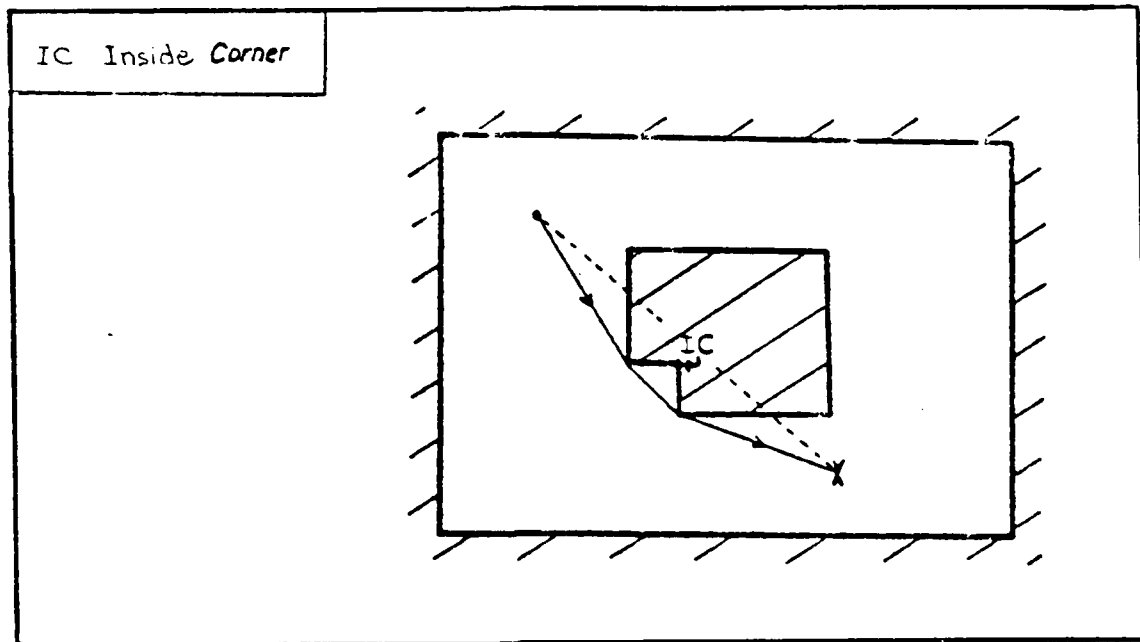


Figure 4.6b. Inside corners are not used as "way points"
[16:48]

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

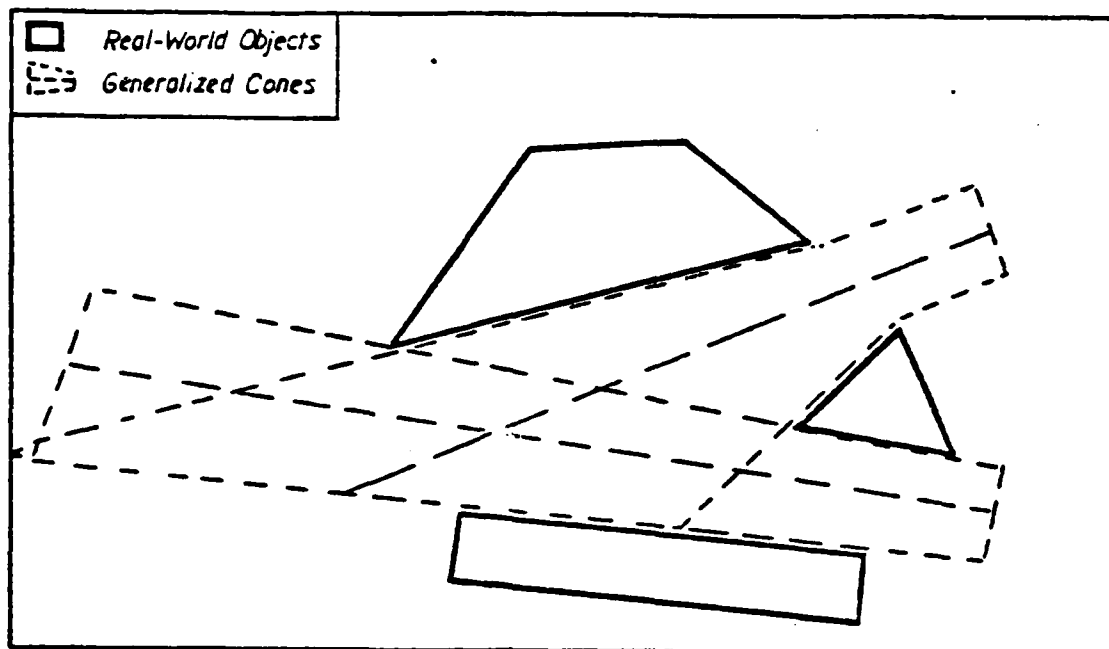


Figure 4.7. Generalized cones form freeways between obstacles [16:27]

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

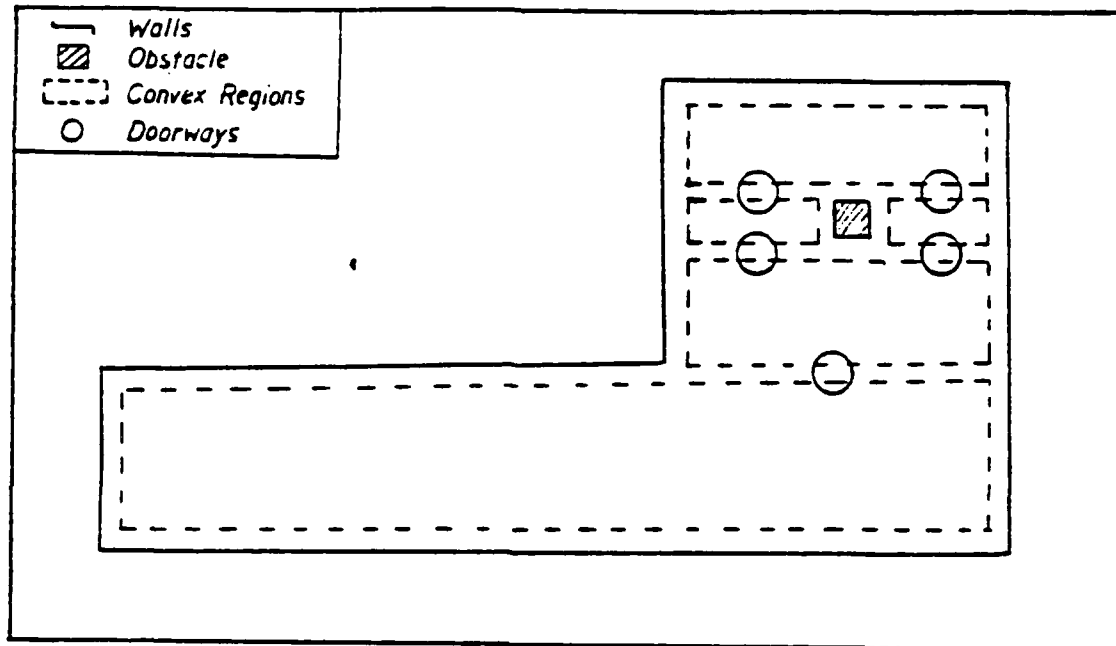


Figure 4.8. Convex regions separated by doorways represent free space [16:29] .

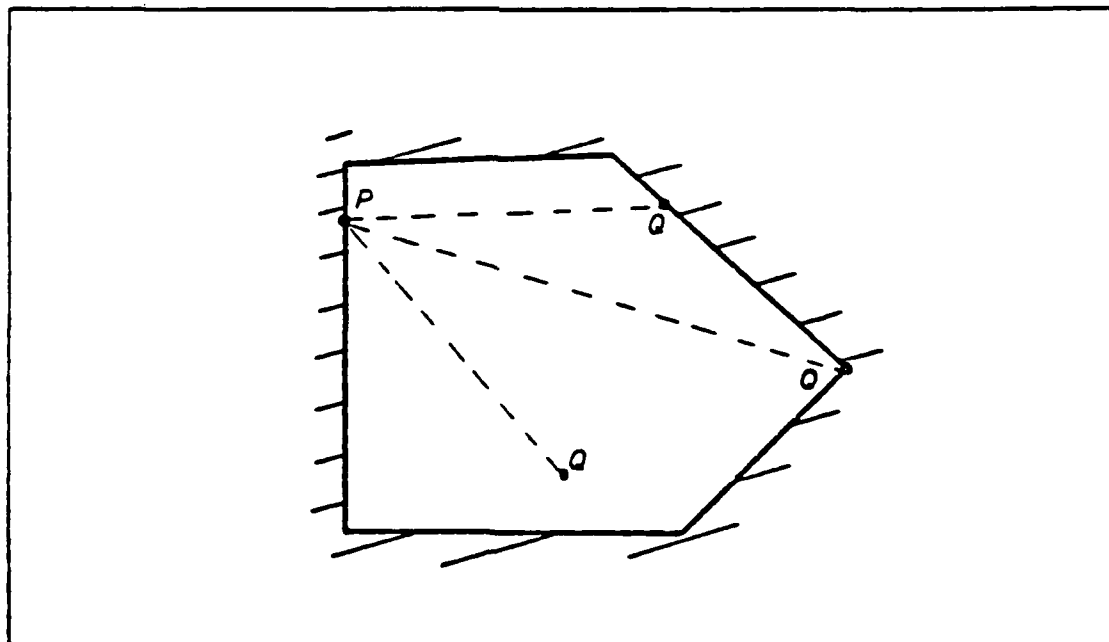


Figure 4.9. Any two points, P and Q, in or on a convex polygon may be connected by an unobstructed straight line [16:45] .

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

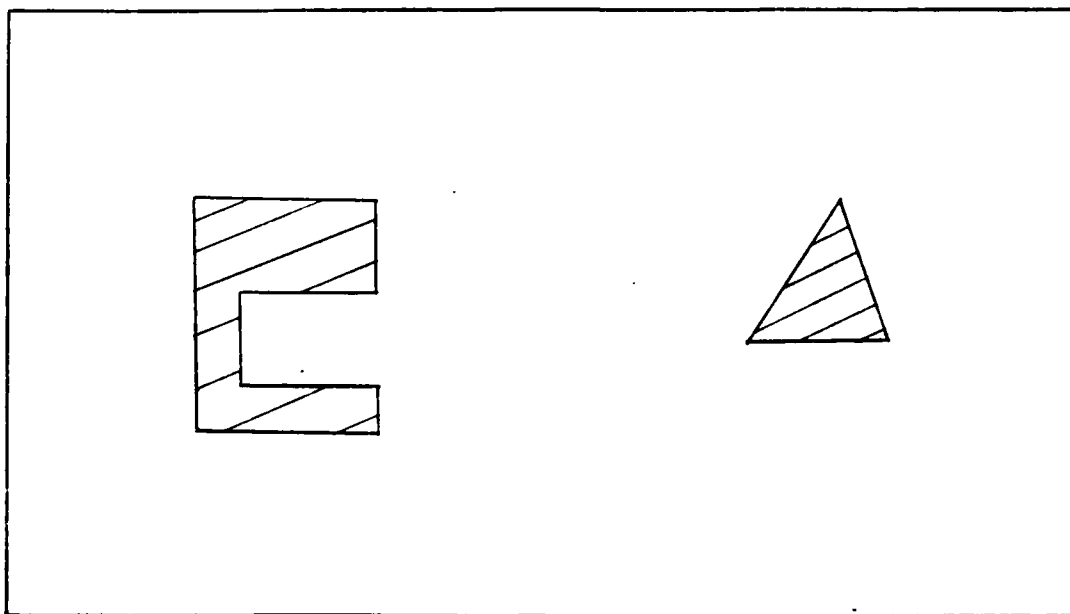


Figure 4.10 Room with two obstacles.

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

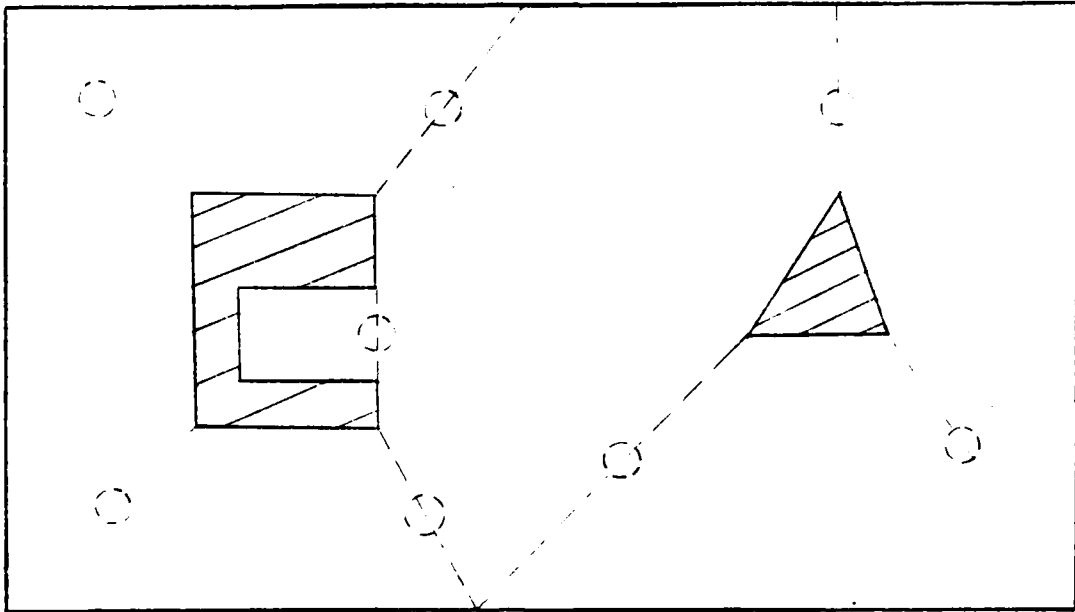


Figure 4.11. Free space is divided into convex regions to define "safe points."

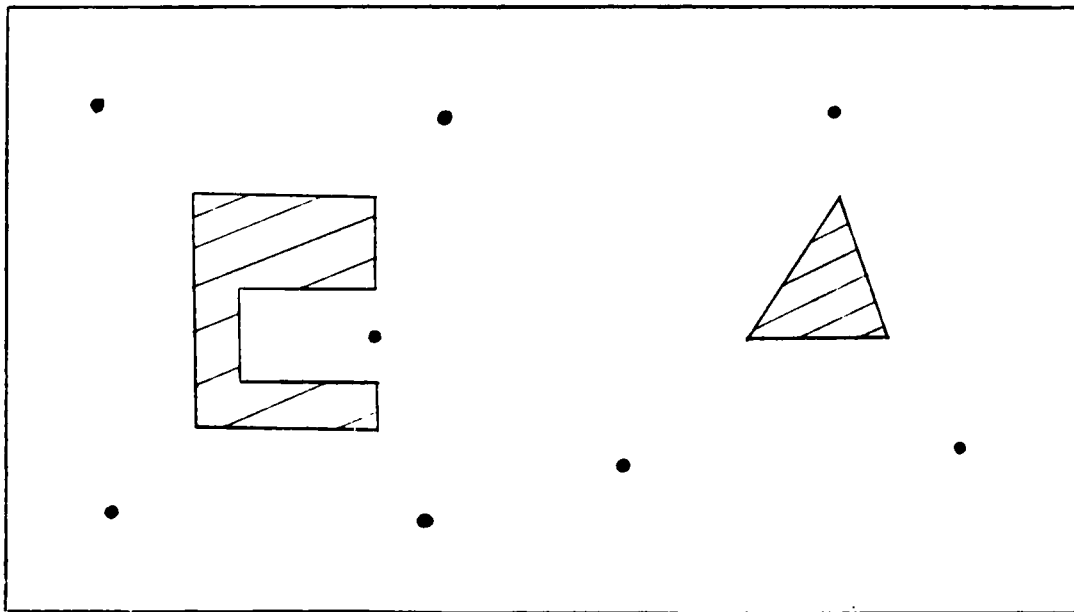


Figure 4.12. Only obstacles and safe points are modeled.

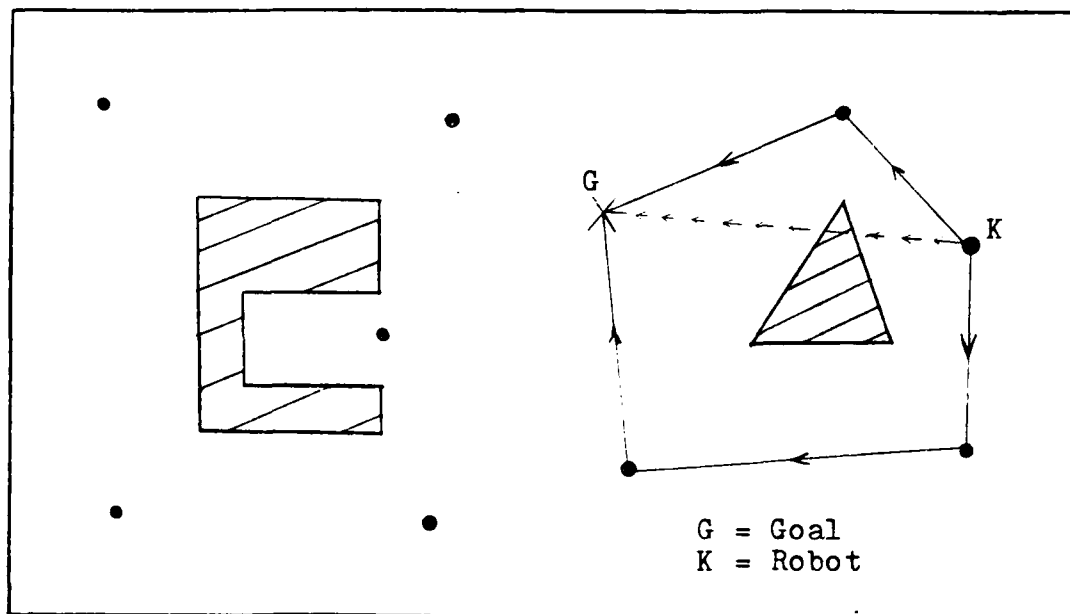


Figure 4.13. Indirect pathways pass through safe points.

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

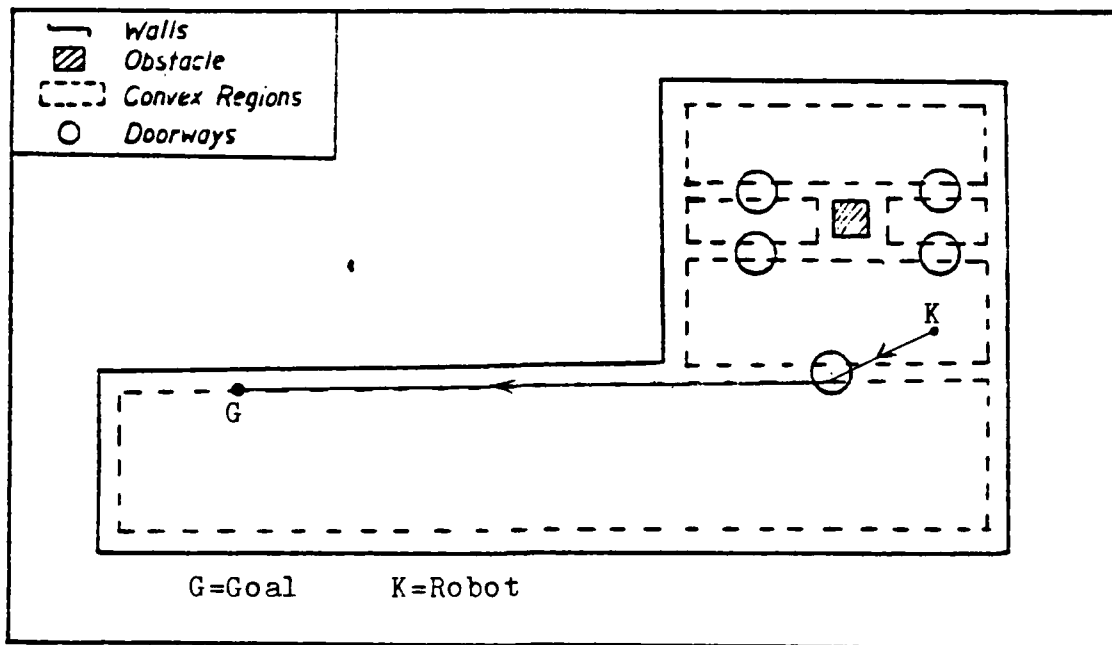


Figure 4.14. Problems occur if free space regions are not chosen correctly.

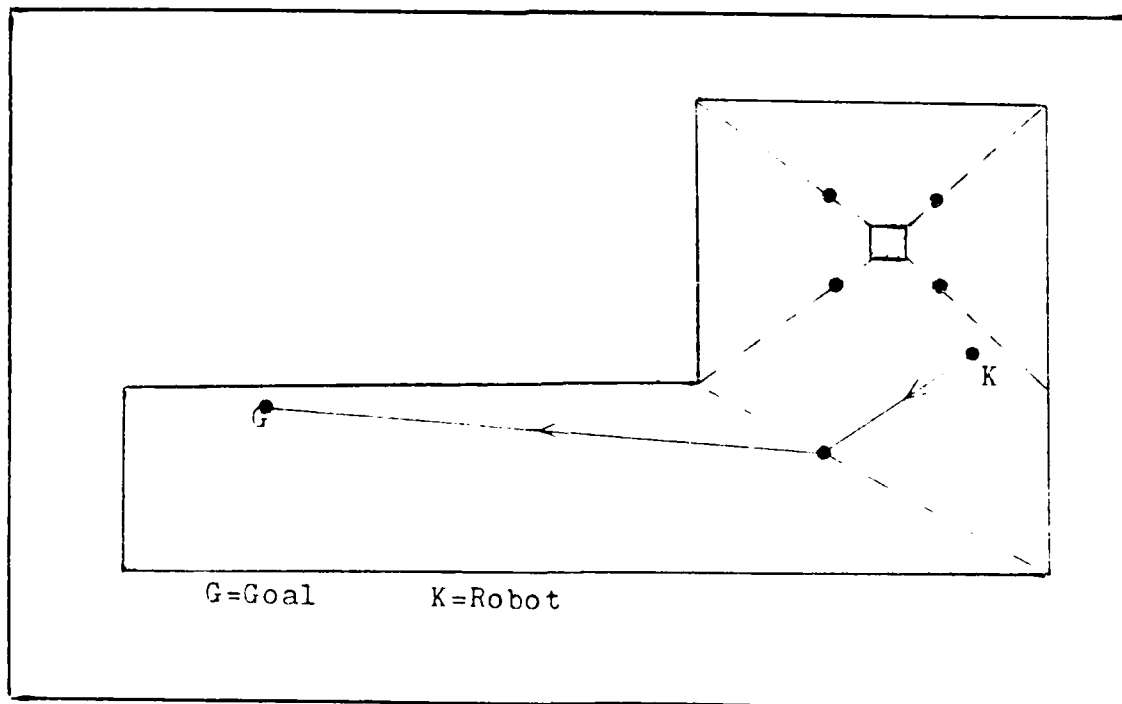


Figure 4.15. Free space regions can be chosen to minimize the probability of collision.

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

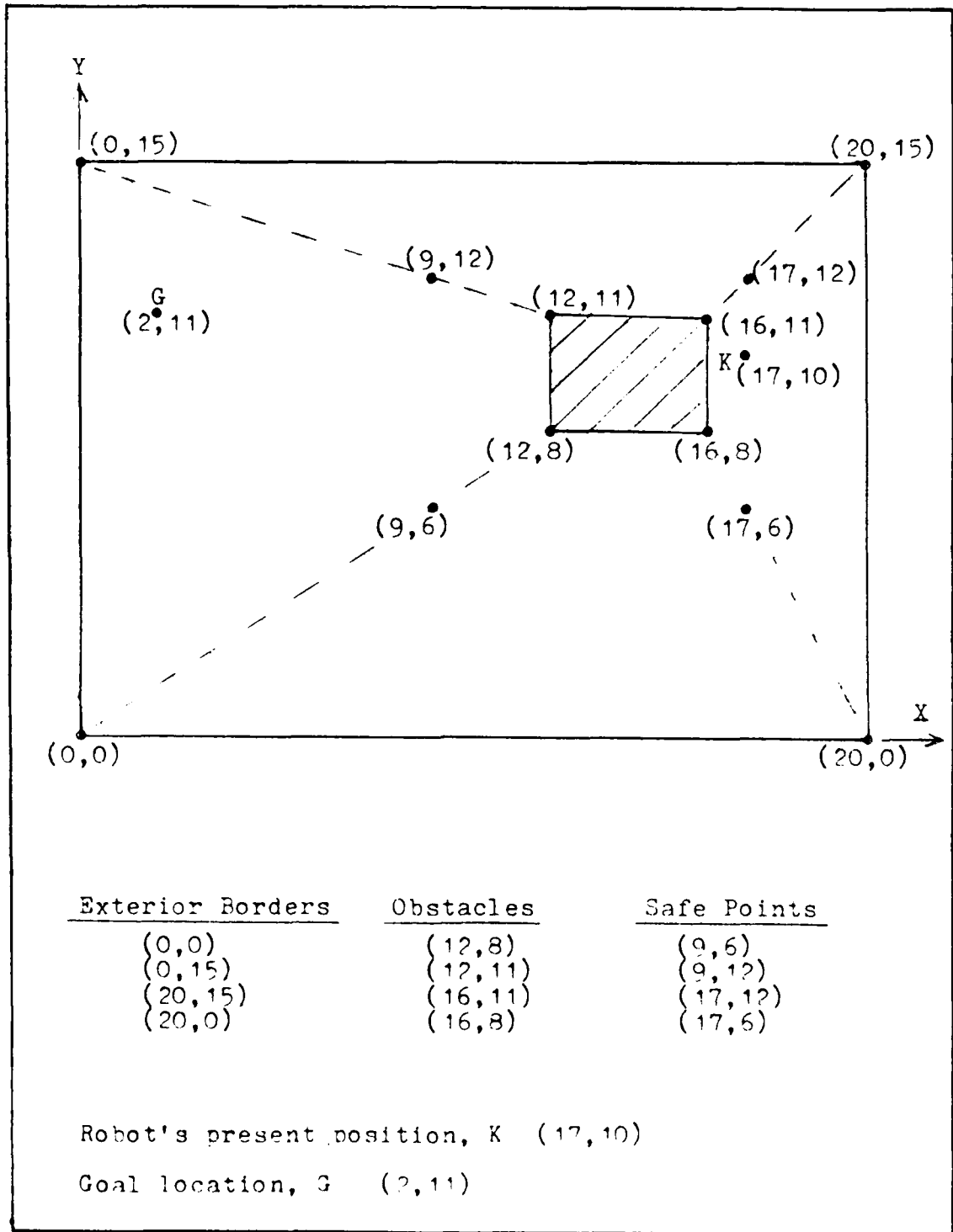


Figure 4.16. Modeling of a room with one obstacle as a set of points.

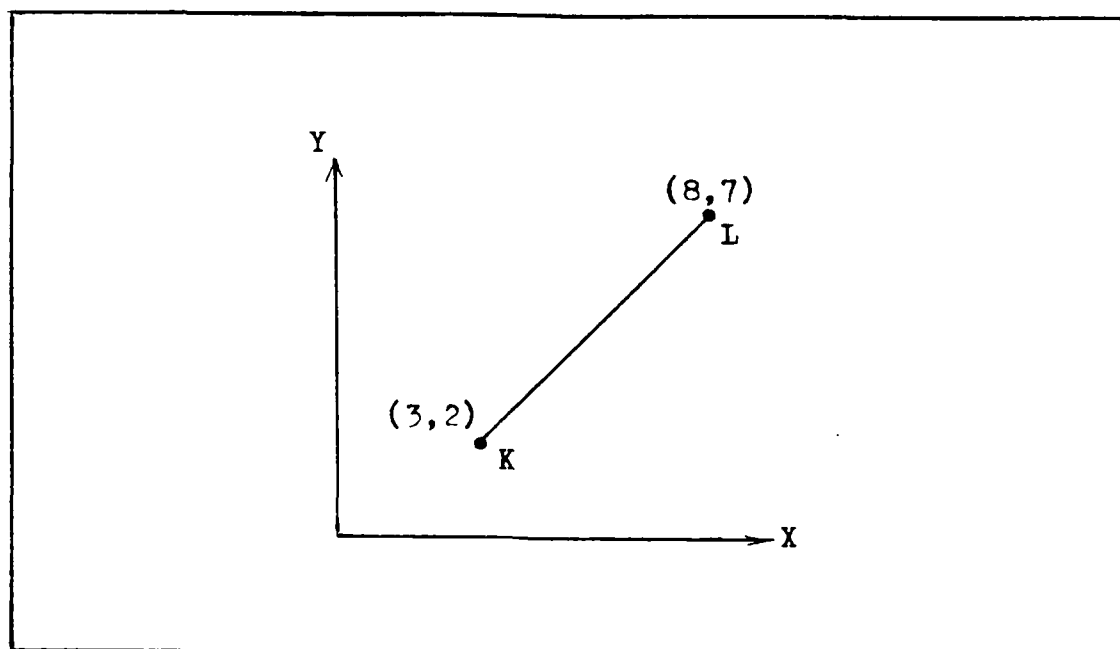


Figure 4.17. Line connecting two points can be represented through parametric equations.

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

$$\begin{aligned}
 X &= X_K + (X_L - X_K)s \\
 Y &= Y_K + (Y_L - Y_K)s \quad (1)
 \end{aligned}$$

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

$$\begin{aligned}
 X &= 3 + (8 - 3)s \\
 Y &= 2 + (7 - 2)s
 \end{aligned}$$

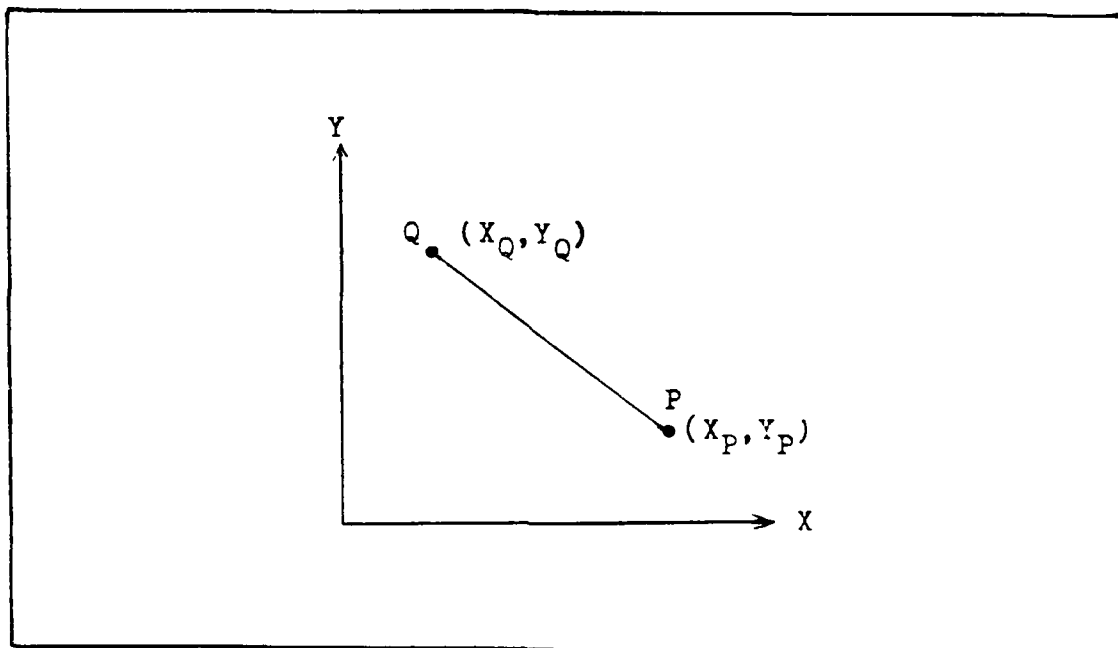


Figure 4.18. Line connecting two arbitrary points.

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

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

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

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:

$$s = \frac{(X_Q - X_P)(Y_P - Y_K) - (Y_Q - Y_P)(X_P - X_K)}{(X_Q - X_P)(Y_L - Y_K) - (Y_Q - Y_P)(Y_L - X_K)} \quad (3)$$

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

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 referred to as the parameter test.

To determine if an obstacle lies in the direct path of the robot, the parameter test is performed. The robot's 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

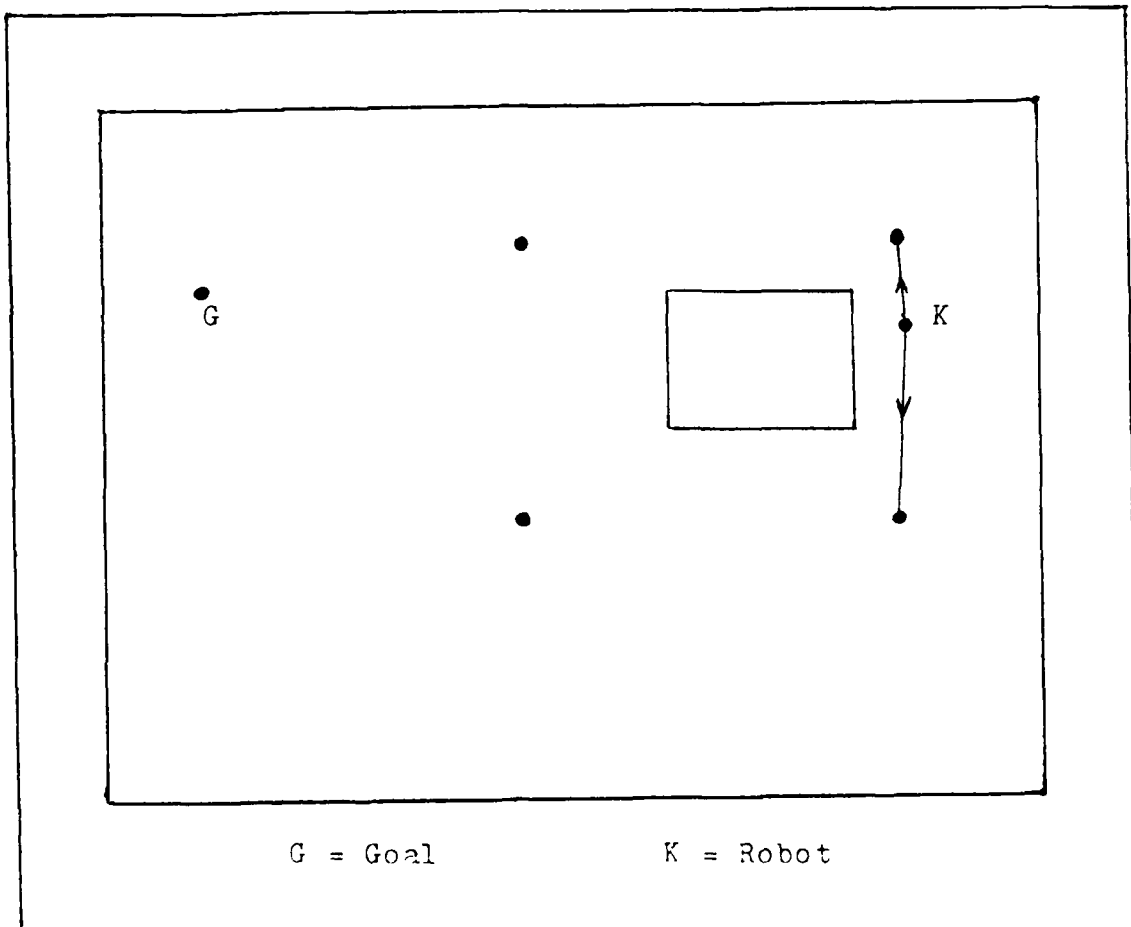


Figure 4.19 Two safe points can be reached through a direct path from the robots present position.

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

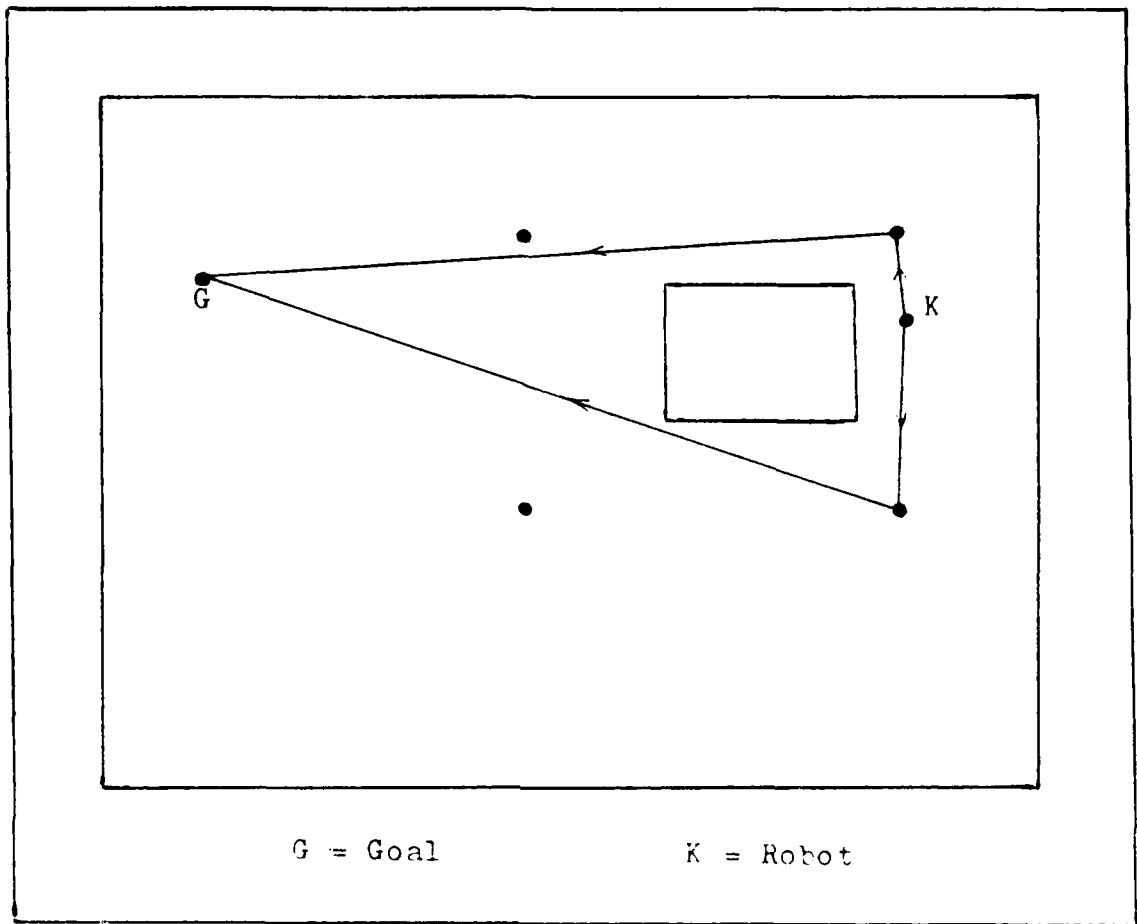


Figure 4.20. Two safe point paths lead to the goal.

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:

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

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:

- Phase I. Verify the functionality of the GYRAC system.
- Phase II. 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.
- Phase III. 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 output 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:

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

where R = drift rate
I = current bias of operational amplifier
B
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 be 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 hexadecimal 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

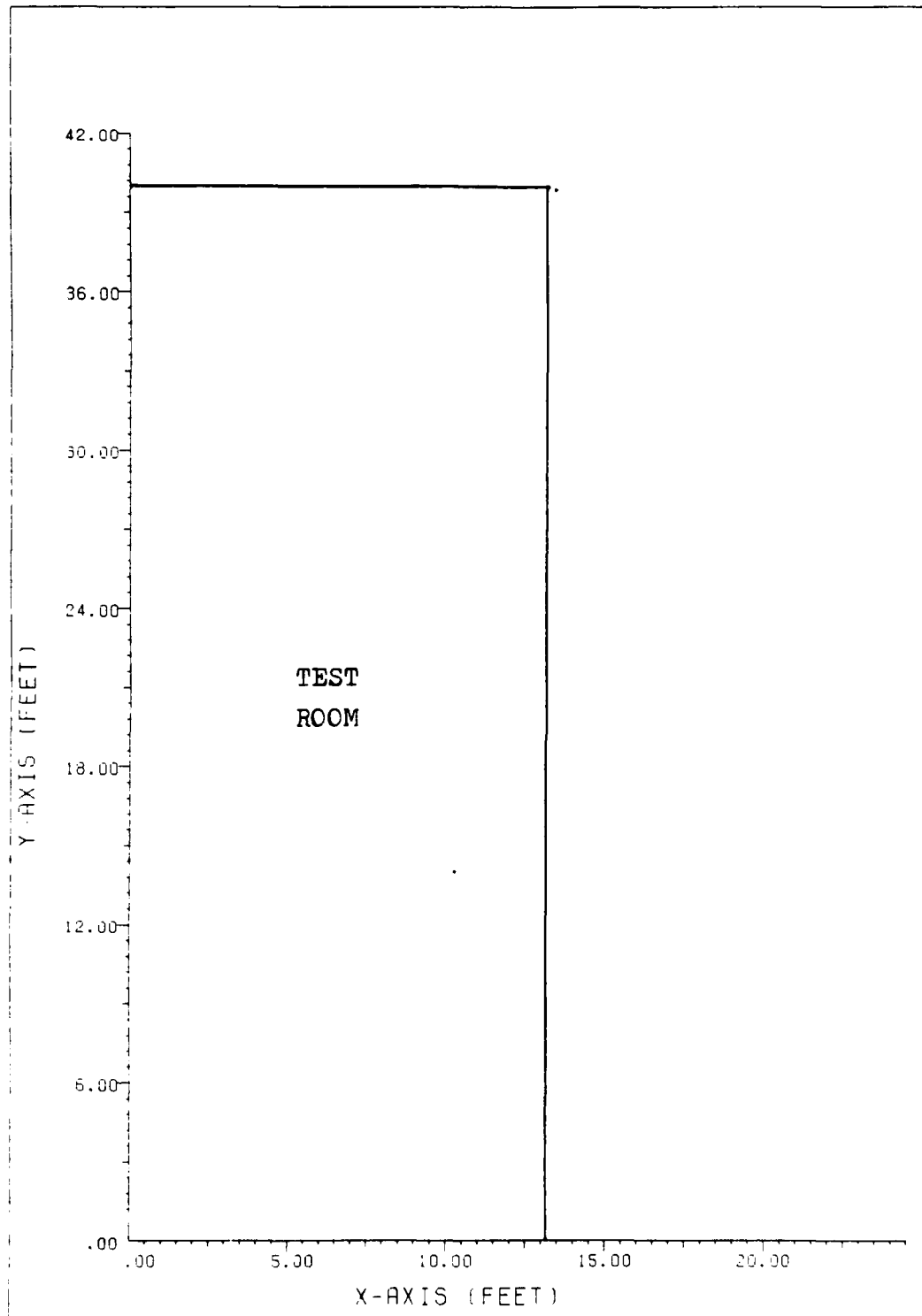


Figure 5.1. Cartesian coordinate system is centered in corner of test area.

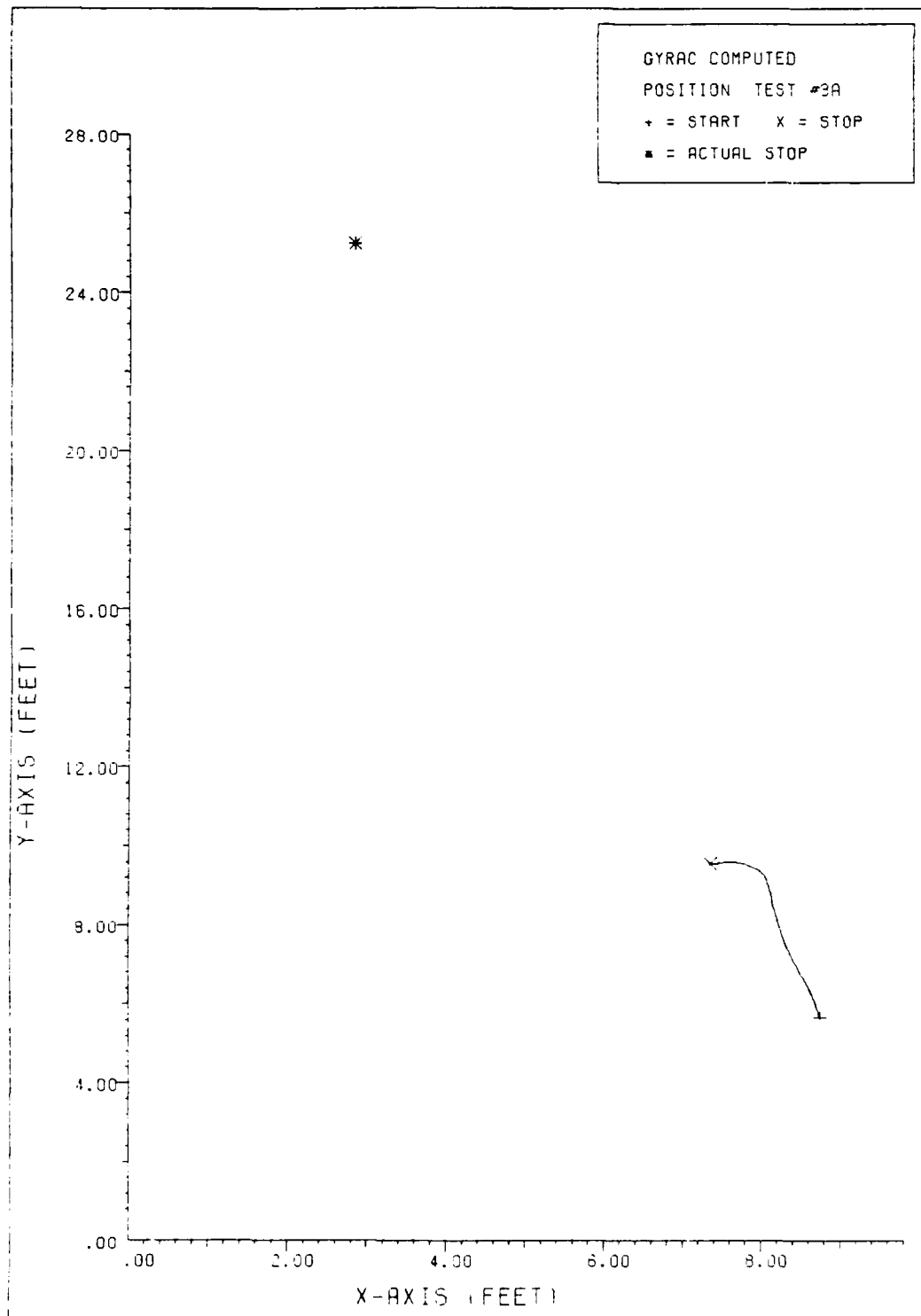


Figure 5.2. GYRAC computed position test using accelerometer sensitivity of 2v/g.

no longer valid. A tumble test of the accelerometer was performed with the accelerometer completely connected to the 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

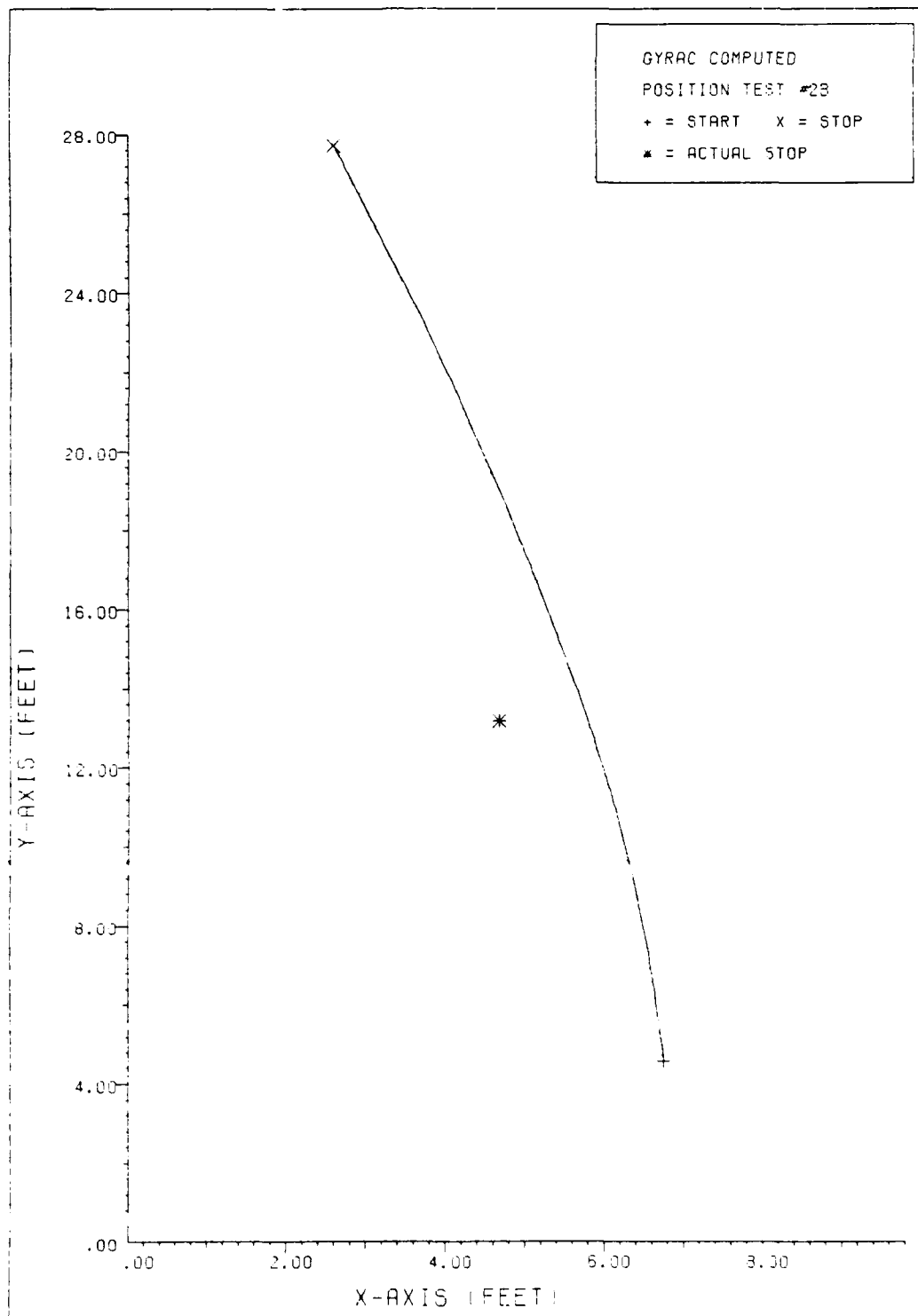


Figure 5.3. GYRAC computed position test using accelerometer sensitivity of 0.393v/g.

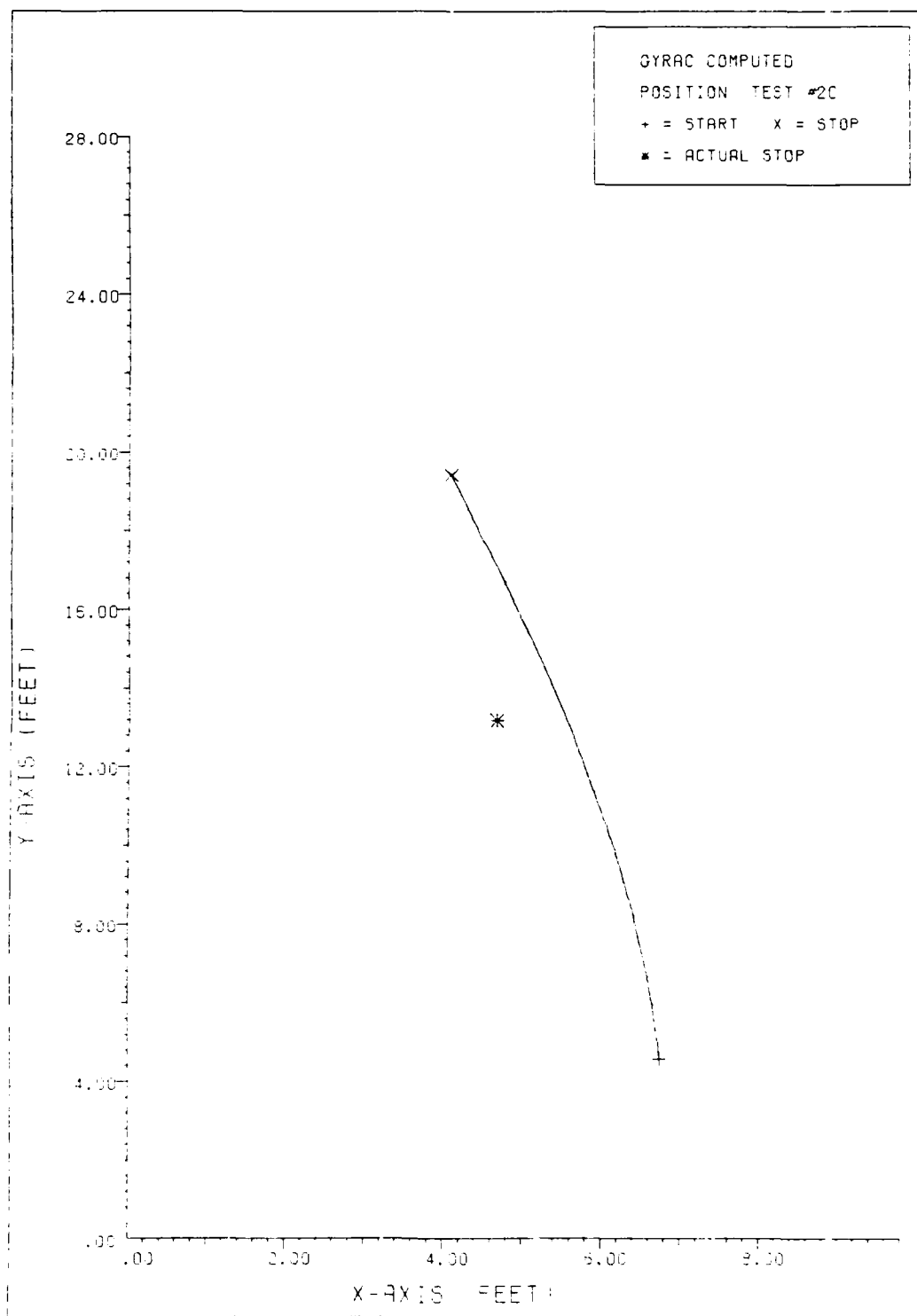
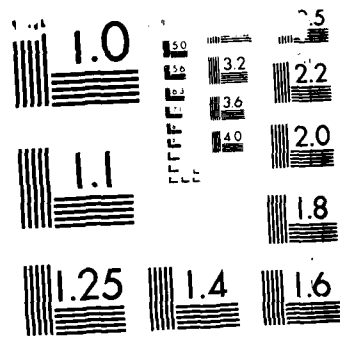


Figure 5.4. GYRAC computed position test using accelerometer sensitivity of 0.6v/g.



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

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 of 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

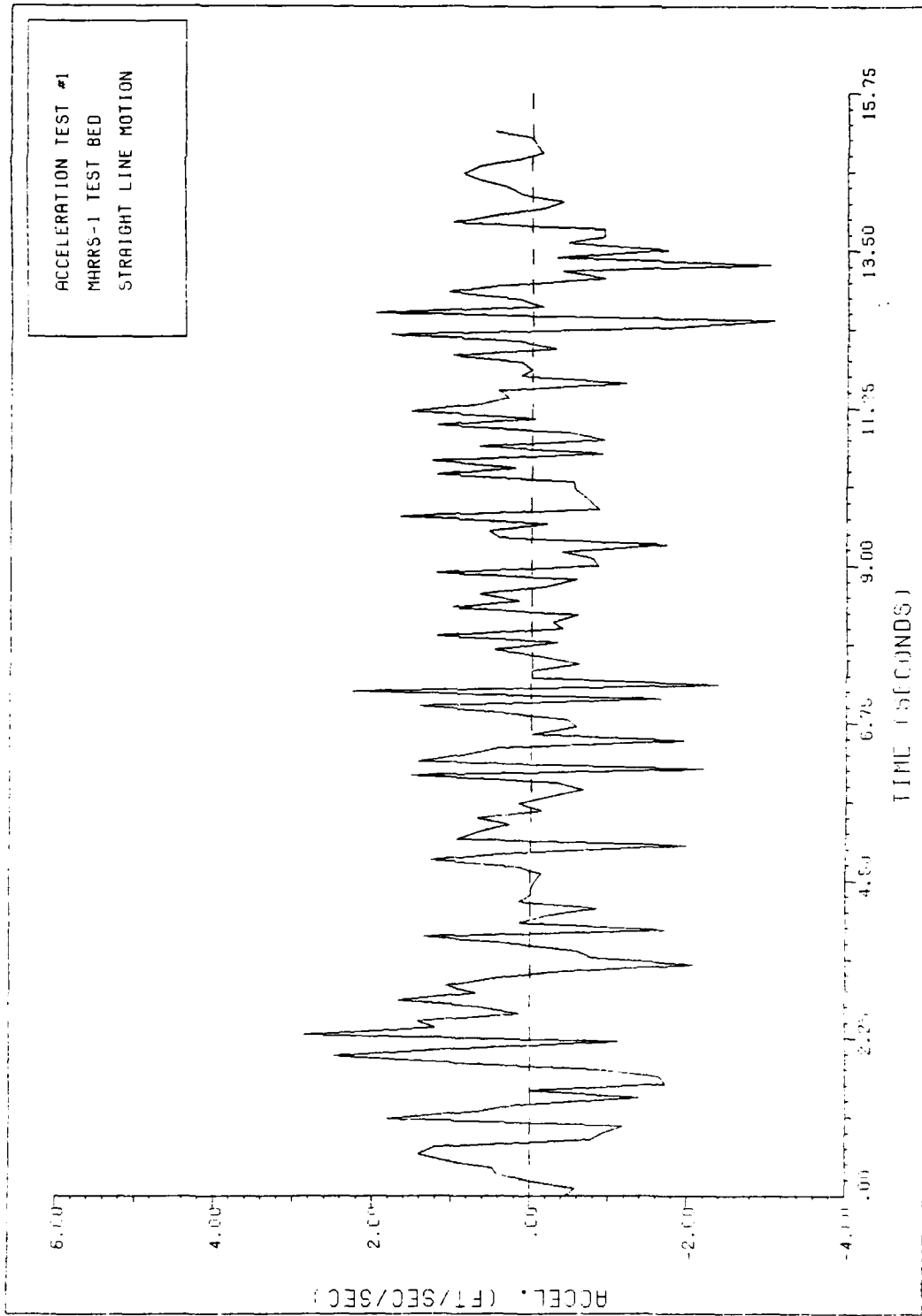


Figure 5.5 Acceleration Test #1.

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 overshoot 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

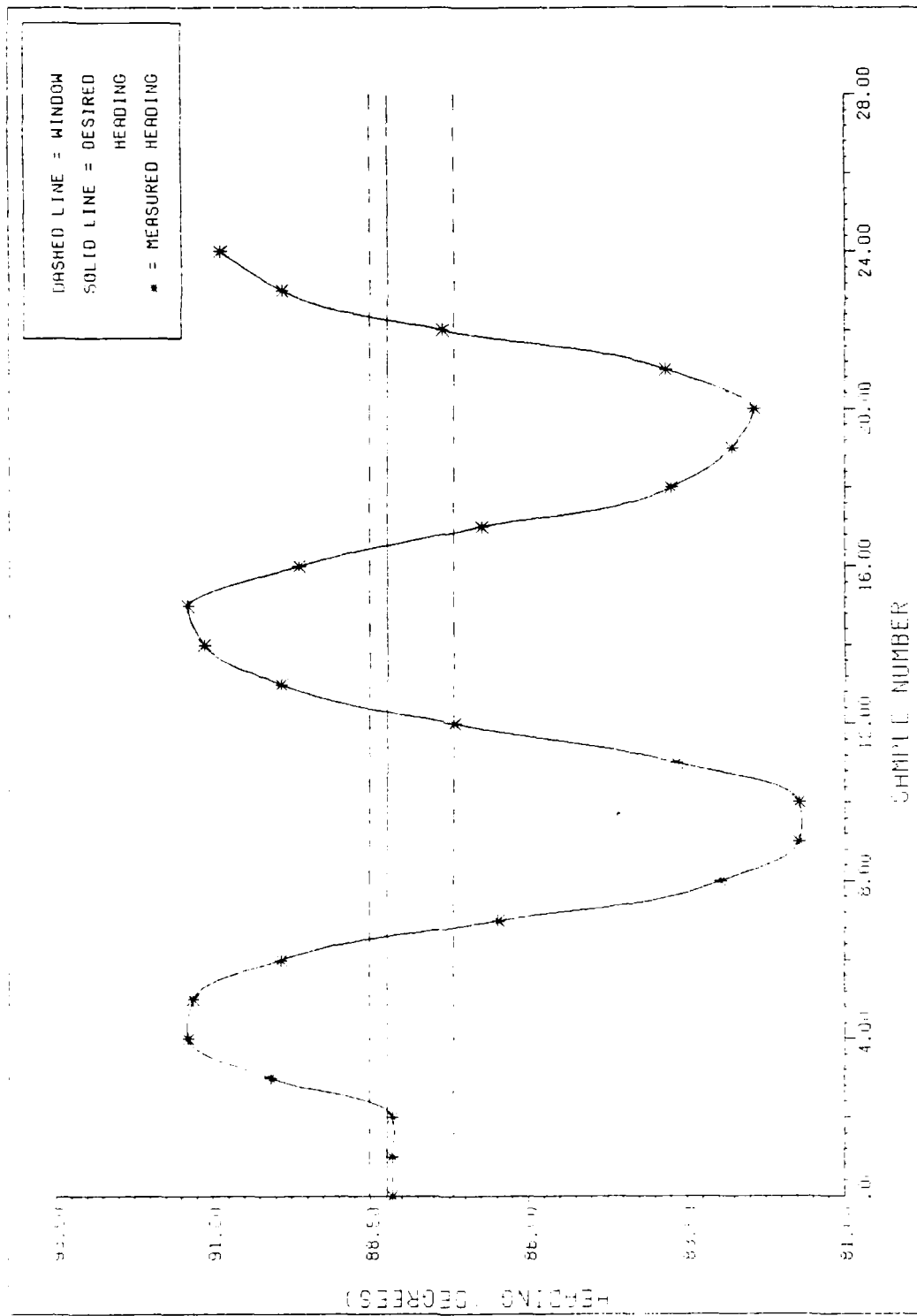


Figure 5.6. GYRAC Navigation Test #1, Automatic control of MARRS-1 Heading.

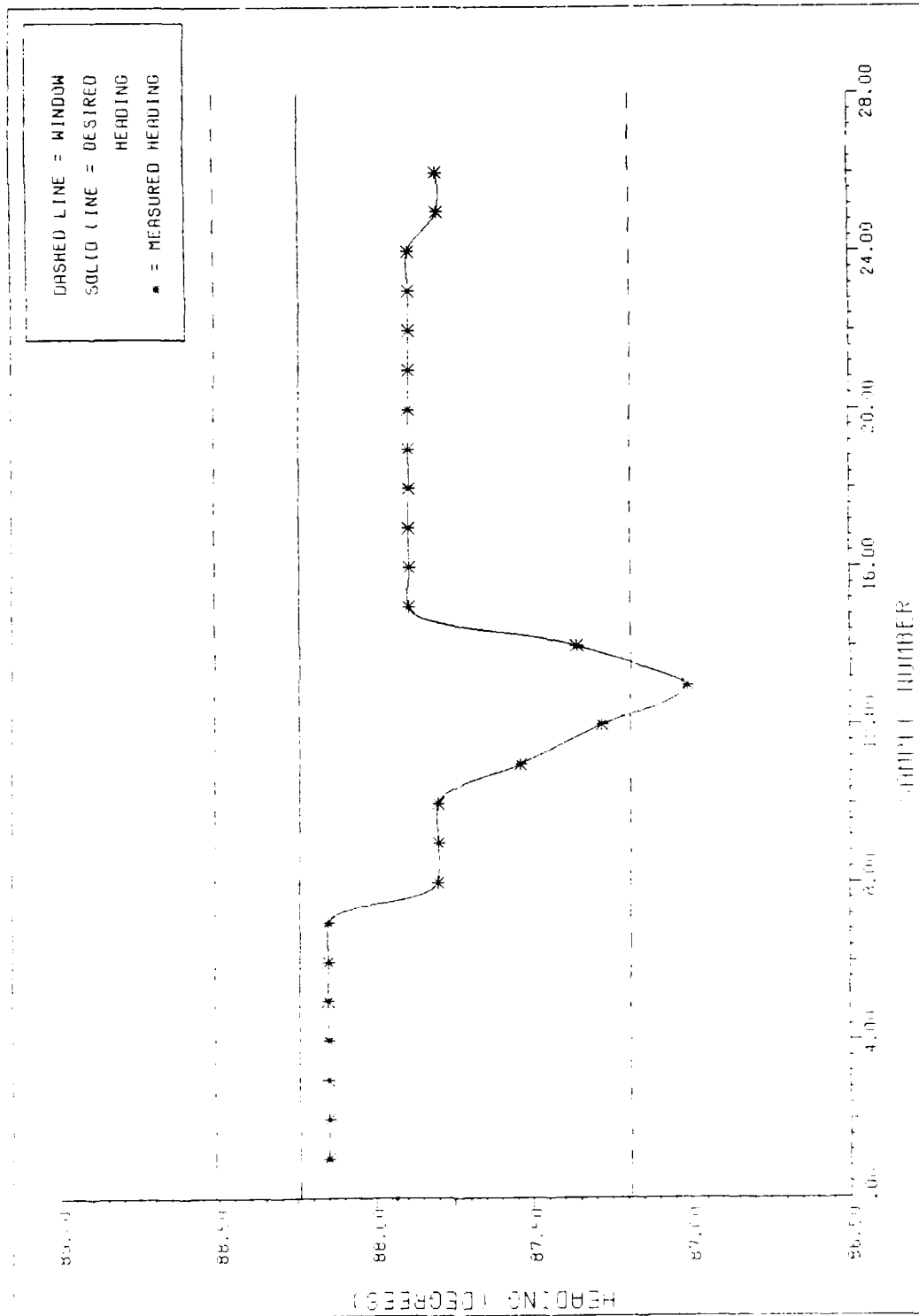


Figure 5.7. GYRAC Navigation Test #2, Automatic Control of MARRS-1 Heading.

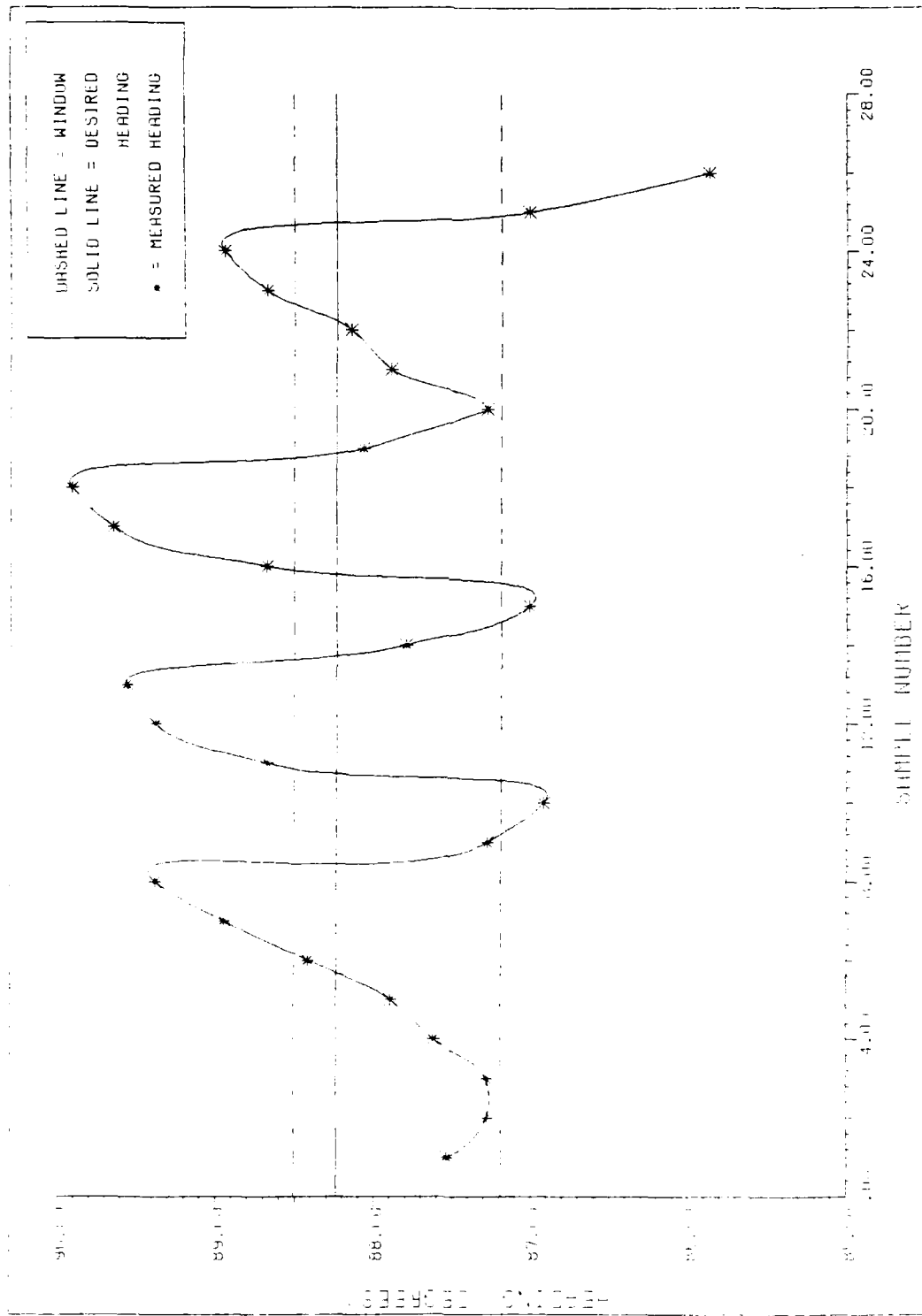


Figure 5.8. GYRAC Navigation Test #3, Automatic Control of MARRS-1 Heading.

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 nemesis 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.

Miscellaneous

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 susceptible to bias and scale factor errors. Also, a new integrator circuit should be designed with a much higher impedance 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 approach 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 implemented 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 (February 1985).
12. Crowley, James I. "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 Requirements 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 Intelligent 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 A-2
AD573 A/D Converter Data Sheet A-8
AD544 Operational Amplifier Data Sheet A-15



**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.

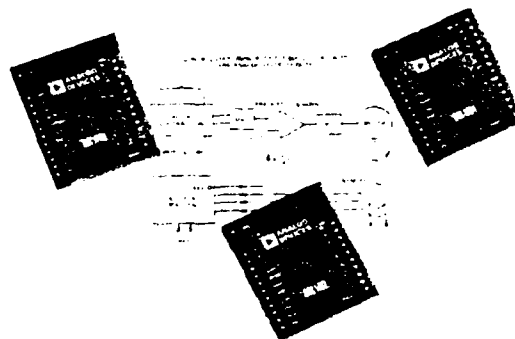
They are intended for use in both Industrial and Military applications.

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 ± 1 LSB 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)

MODEL	SDC1702	SDC1700	SDC1704
ACCURACY ¹ (max error)			
50Hz	±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 kHz	±22 arc minutes	±8.5 arc minutes	±2.0 arc minutes ±1LSB
RESOLUTION	10 Bits (1LSB = 21 arc mins)	12 Bits (1LSB = 5.3 arc mins)	14 Bits (1LSB = 1.4 arc mins)
OUTPUT (in Parallel)	10 Bits (Natural Binary)	12 Bits (Natural Binary)	14 Bits (Natural Binary)
SIGNAL AND REFERENCE FREQUENCY	50Hz, 400Hz, 2 kHz	*	*
SIGNAL VOLTAGE (Line-to-Line)			
Low Level	11.8V rms	*	*
High Level	90V rms	*	*
SIGNAL IMPEDANCES			
Low Level	20kΩ (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)			
50Hz	5 Revolutions Per Second	*	500 ² sec
400Hz	30 Revolutions Per Second	*	12 Revolutions Per Second
2 kHz	75 Revolutions Per Second	*	25 Revolutions Per Second
Accel ²			
Constant K _a			
50Hz	1880 sec ²	*	520 sec ²
400Hz	110,000 sec ²	*	30,000 sec ²
2 kHz	518,000 sec ²	*	170,000 sec ²
STEP RESPONSE (179° Step) (For 1LSB Error)			
50Hz	1.5sec	*	*
400Hz	125ms	*	*
2 kHz	50ms	*	*
POWER LINES	+15V @ 25mA } ±5% -5V @ 70mA }	*	+15V @ 30mA } ±5% -5V @ 85mA }
POWER DISSIPATION	1.1 Watts	*	1.3 Watts
DATA LOGIC OUTPUT ³ TTL Compatible	2TTL Loads SDC17026VZ 4TTL Loads SDC17025VZ	2TTL Loads SDC17006VZ 4TTL Loads SDC17005VZ	2TTL Loads on All Options
BUSY LOGIC OUTPUT POSITIVE PULSE (1 TTL Load)			
50Hz	9 μs	*	9 μs
400Hz	2 μs	*	2 μs } ±10%
2 kHz	2 μs	*	1 μs }
MAX DATA TRANSFER TIME			
50Hz	40 μs	*	15 μs
400Hz	5 μs	*	1 μs
2 kHz	1.8 μs	*	0.8 μs
INHIBIT INPUT (to inhibit)	Logic 0 ⁴ (1 TTL Load)	*	Logic 0 ⁴ (2 TTL Loads)
WARMUP TIME	1sec to Rated Accuracy	*	*
TEMPERATURE RANGE			
Operating	0 to +70°C Standard	*	*
Storage	+55°C to +148°C Extended -55°C to +125°C	*	*
DIMENSIONS	1.125" x 2.025" x 0.14" 74.4 x 50.9 x 3.54 mm	*	*
WEIGHT	1.05 oz (30 grams)	*	*

¹Typical accuracy same as SDC1702
²Typical over the appropriate operating temperature range of the option and for 4% signal and reference amplitude variation, 5% signal and reference harmonic distortion, ±0.5% power supply variation and ±10% variation in reference frequency.
³It is recommended that buffers should be used if the above converters are required to drive over a distance greater than 6'.
⁴Temperature range without notice.

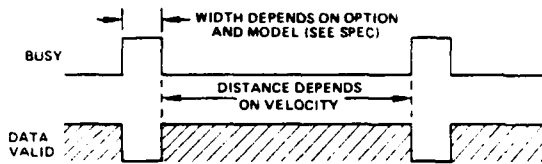
SDC 1700 S/D Converter Data Sheet (continued)

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

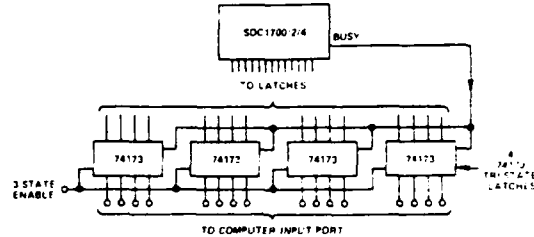
Taking the INHIBIT to a logic "Lo" state prevents the VCO (BUSY) pulses from updating the up-down counter. However, if applied during a BUSY pulse, the INHIBIT will not become effective until the end of the BUSY pulse.

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.

$$\begin{aligned} \text{i.e., } V_1 &= K E_0 \sin \omega t \sin \theta \\ V_2 &= K E_0 \sin \omega t \cos \theta \end{aligned}$$

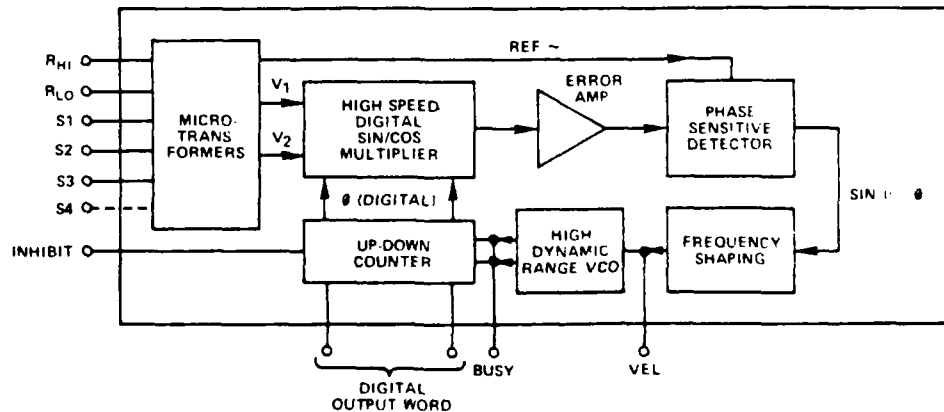
Where θ is the angle of the Synchro Shaft

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 $\cos \phi$ and V_2 is multiplied by $\sin \phi$ to give

$$\begin{aligned} &K E_0 \sin \omega t \sin \theta \cos \phi \\ \text{and } &K E_0 \sin \omega t \cos \theta \sin \phi \end{aligned}$$



Functional Diagram of the SDC 1700 2/4 Converters

SDC 1700 S/D Converter Data Sheet (continued)

These signals are subtracted by the error amplifier to give

$$K E_O \sin \omega t (\sin \theta \cos \phi - \cos \theta \sin \phi)$$

$$\text{or } K E_O \sin \omega t \sin (\theta - \phi)$$

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 $\pm 15V$ and $5V$. They must be connected to the " $\pm 15V$ " and " $5V$ " pins with the common connection to the ground pin GND.

It is suggested that $0.1\mu F$ and $0.8\mu F$ 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:

$$E_{S1 - S3} = E_{RLO} - R_{HI} \sin \omega t \sin \theta$$

$$E_{S3 - S2} = E_{RLO} - R_{HI} \sin \omega t \sin (\theta + 120^\circ)$$

$$E_{S2 - S1} = E_{RLO} - R_{HI} \sin \omega t \sin (\theta + 240^\circ)$$

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

$$E_{S1 - S3} = E_{RLO} - R_{HI} \sin \omega t \sin \theta$$

$$E_{S2 - S4} = E_{RHI} - R_{LO} \sin \omega t \cos \theta$$

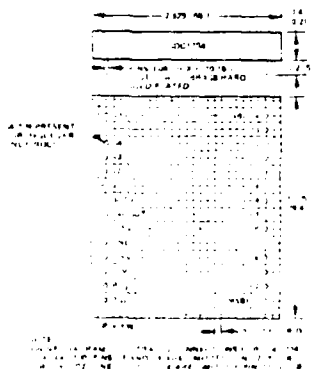
The analog voltage representing velocity is available between " VEL " and " GND ".

The " $BCSY$ " 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

VOL. II, 13-52 SYNCHRO & RESOLVER CONVERTERS

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 $1.11k\Omega$ in series with $S1$, $S2$ and $S3$ per extra volt in the case of the signal, and $2.2k\Omega$ in the case of the reference. In the case of a resolver converter add $2.22k\Omega$ per extra volt in series with $S1$ and $S2$ for the signal and $2.2k\Omega$ 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:

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

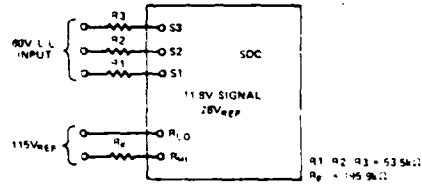
Therefore each resistor needs to have a value of $48.2 \times 1.11 = 53.5k\Omega$. In the case of the reference, the extra voltage capability required is:

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

Therefore the resistor needs to have a value of:

$$89.0 \times 2.2 = 195.8k\Omega$$

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 $R4$ IS NOT CRITICAL.

BIT WEIGHT TABLE

Bit Number	Weight in Degrees
1 (MSB)	150.0000
2	75.0000
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

SDC 1700 S/D Converter Data Sheet (continued)

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.

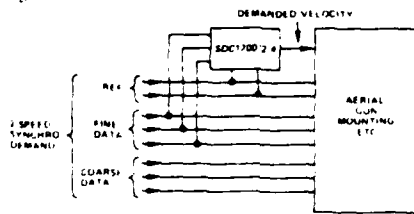


Diagram showing a velocity feed forward application. The SDC is used to produce the demanded velocity from Synchro form inputs.

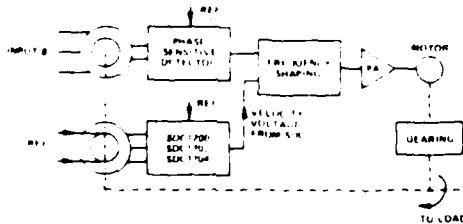
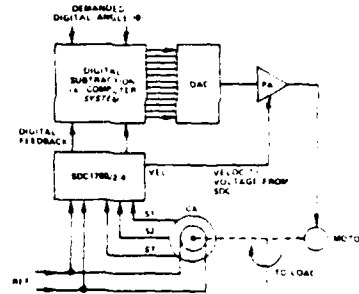


Diagram showing the velocity voltage being used to stabilize an electro-mechanical control loop.

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.

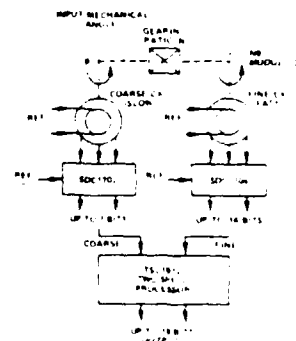


Diagram Showing Coarse-Fine Synchro Processor System

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.

SDC 1700 S/D Converter Data Sheet (continued)

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

$$\frac{\theta_0}{\theta_1} = \frac{8.8 \times 10^7 (1 + 0.8 \times 10^{-3}s)}{s^3 + 8.04 \times 10^2 s^2 + 6.1 \times 10^5 s + 8.8 \times 10^7}$$

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.

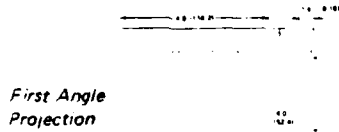
The latches have a tri-state output to facilitate ease of use.

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

Pin Number	Function	Pin Letter	Function
1	REF	A	Reference Voltage
2	REF	B	Reference Voltage
3	REF	C	Reference Voltage
4	REF	D	Reference Voltage
5	REF	E	Reference Voltage
6	REF	F	Reference Voltage
7	REF	G	Reference Voltage
8	REF	H	Reference Voltage
9	REF	J	Reference Voltage
10	REF	K	Reference Voltage
11	REF	L	Reference Voltage
12	REF	M	Reference Voltage
13	REF	N	Reference Voltage
14	REF	P	Reference Voltage
15	REF	Q	Reference Voltage
16	REF	R	Reference Voltage
17	REF	S	Reference Voltage
18	REF	T	Reference Voltage
19	REF	U	Reference Voltage
20	REF	V	Reference Voltage
21	REF	W	Reference Voltage
22	REF	X	Reference Voltage

ORDERING INFORMATION

Parts should be ordered by the appropriate part number (i.e.,

VOL. II, 13-54 SYNCHRO & RESOLVER CONVERTERS

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 SBDC1752 and SBDC1753 are converters with a BCD instead of a binary output based upon the SDC1700. They have outputs of 0-360.0 degrees and 0 to 360.0 degrees respectively.

*60Hz Operation

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



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 - 20 μ s typ
- No Missing Codes Over Temperature
- Operates on +5V and -12V to -15V Supplies
- Low Cost Monolithic Construction

PRODUCT DESCRIPTION

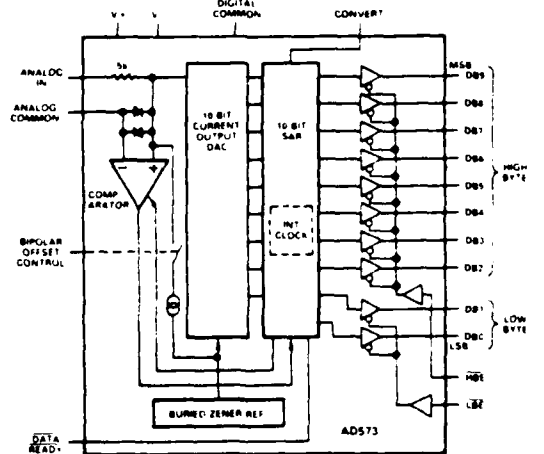
The AD573 is a complete 10-bit successive approximation analog to digital converter consisting of a DAC, voltage reference, clock, comparator, successive approximation register (SAR) and 3 state output buffers—all fabricated on a single chip. No external components are required to perform a full accuracy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

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 20 μ s conversion cycle. DATA READY indicates completion of the conversion. HIGH BYTE ENABLE (HBE) and LOW BYTE ENABLE (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 ± 1 LSB relative accuracy and no missing codes from -55 C to +125 C.

AD573 FUNCTIONAL BLOCK DIAGRAM



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; 4,400,619; and 4,400,690.

AD573 A/D Converter Data Sheet (continued)

SPECIFICATIONS ($T_A = 25^\circ\text{C}$, $V_+ = +5\text{V}$, $V_- = -12\text{V}$ or -15V , all voltages measured with respect to digital common, unless otherwise indicated)

Model	AD573J			AD573K			AD573S			Units
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
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 to +5.5V	± 2			± 1			± 2			LSB
Negative Supply										
-15V to $V_- = -14.25\text{V}$	± 2			± 1			± 2			LSB
-12.5V to $V_- = -11.4\text{V}$	± 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	k Ω
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										
$V_{OL} = 0.4\text{V max}$, T_{min} to T_{max}	3.2			3.2			3.2			mA
Output Source Current ⁵										
$V_{OH} = 2.4\text{V max}$, 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	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			N20A			

NOTES

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.

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

Full scale is defined as 10 volts minus 1 LSB, or 9.999 volts.

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

⁴ Range from -15°C to $+25^\circ\text{C}$ to T_{max} or T_{min} .

The data output lines have active pull-ups to source 0.5mA. The DATA READY line is open collector with a nominal 6k Ω 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. 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 (I²L) 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 ±1LSB (0.05% of full scale).

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

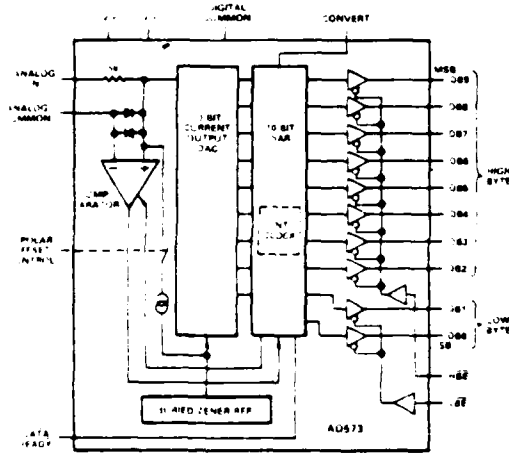


Figure 1 AD573 Functional Block Diagram

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 ±1LSB) 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 signal plus an

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 -18V, 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 pinouts 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).

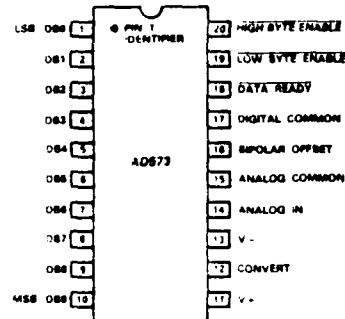


Figure 2 AD573 Pin Connections

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 500Ω 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 11111111 10 and 11111111 11. 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.

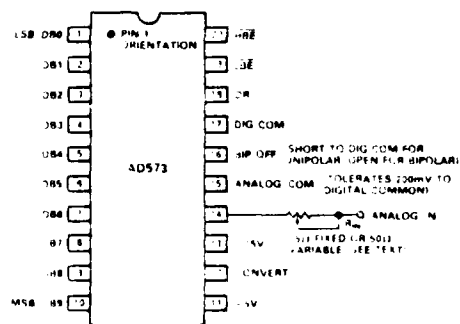


Figure 3 Standard AD573 Connections

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.

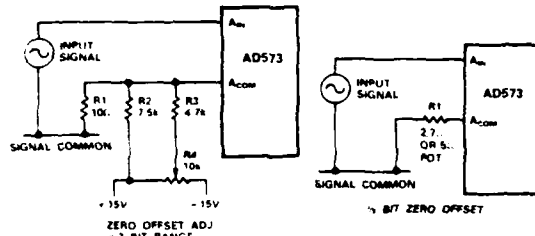


Figure 4a.

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.

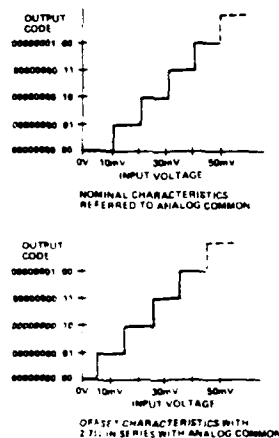


Figure 5 AD573 Transfer Curve - Unipolar Operation (Approximate Bit Weights Shown for Illustration, Nominal Bit Weights = 9.766mV)

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 2.7k resistor in series with the terminal will result in approximately the desired 1-bit offset of the transfer characteristics. The nominal 2mA Analog Common current is not closely controlled in manufacture. If high accuracy is required, a 5k 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 ± 1 LSB 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.

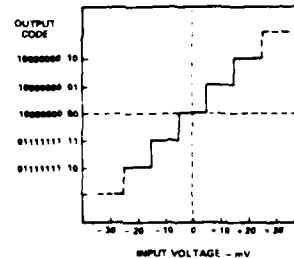


Figure 6. AD573 Transfer Curve - Bipolar Operation

Note that in the bipolar mode, the code transitions are offset $\frac{1}{2}$ LSB such that an input voltage of 0 volts ± 5 mV 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.

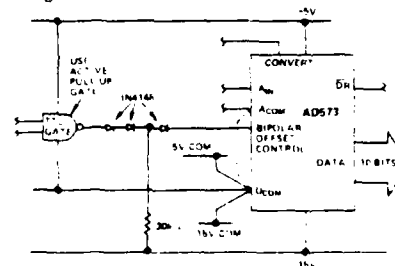


Figure 7. Bipolar Offset Controlled by Logic Gate Gate Output = 1 Unipolar 0 - 10V Input Range Gate Output = 0 Bipolar ± 5 V Input Range

SAMPLE-HOLD AMPLIFIER CONNECTION TO THE AD573

Many situations in high-speed acquisition systems or digitizing rapidly changing signals require a sample-and-hold amplifier (SHA) in front of the A-D converter. The SHA can acquire and hold a

AD573 A/D Converter Data Sheet (continued)

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 10 μ s with a droop rate less than 100 μ V/ms.

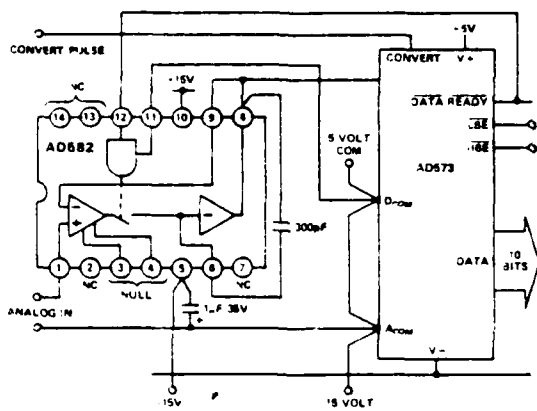


Figure 8. Sample-Hold Interface to the AD573

\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 10 μ s 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 ± 20 mV 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.

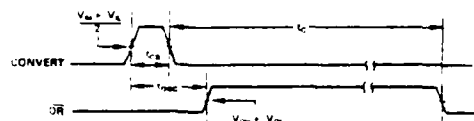


Figure 9. Convert Timing

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.

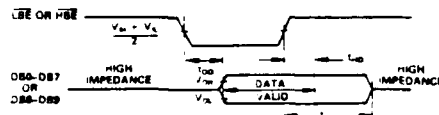


Figure 10. Read Timing

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

Parameter	Symbol	Min	Typ	Max	Units
CONVERT Pulse Width	t_{CS}	500	—	—	ns
\overline{DR} Delay from CONVERT	t_{DSC}	—	1	1.5	μ s
Conversion Time	t_c	10	20	30	μ s
Data Access Time	t_{OD}	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 \overline{RD} to enable the output buffers. The use of a memory address and memory \overline{WR} and \overline{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.

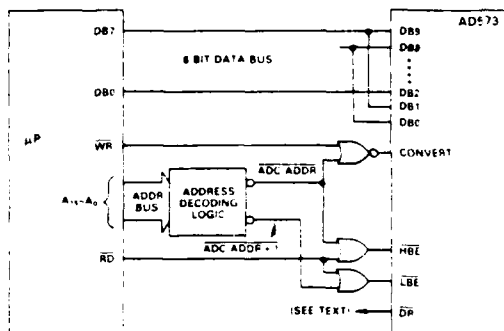


Figure 11. General AD573 Interface to 8-Bit Microprocessor

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 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 DR to signal an interrupt to the processor at the end of a conversion.

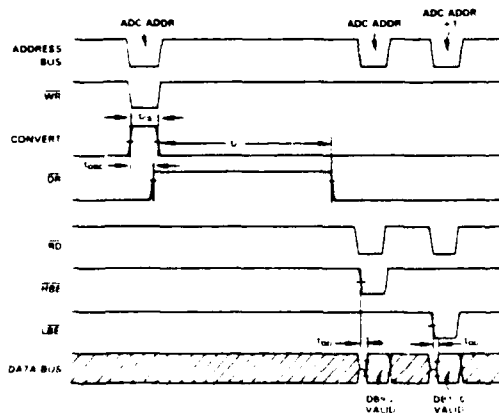


Figure 12. Typical AD573 Interface Timing Diagram

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 WR pulse sets the CONVERT line high through a flip-flop. The rising edge of 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.

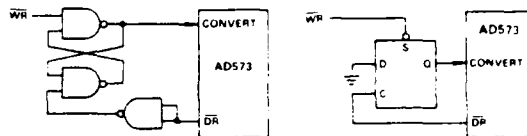


Figure 13a. Using 74LS00 Figure 13b. Using 74LS74

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 HBE (pin 20) and the 2 LSBs (DB1, DB0 - pins 2 and 1) are enabled by 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, 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.

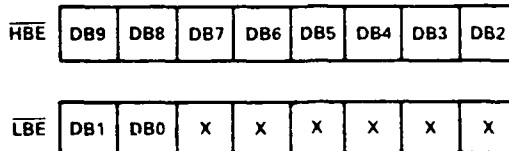


Figure 14. AD573 Output Data Format

In systems where all 10 bits are desired at the same time, HBE and 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 DR output is wired to the HBE and 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 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)

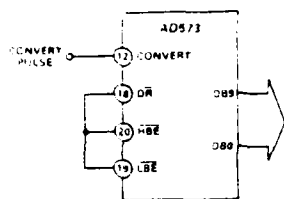


Figure 15. AD573 in "Stand-Alone" Mode
(Output Data Valid 500ns After DR Goes Low)

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.

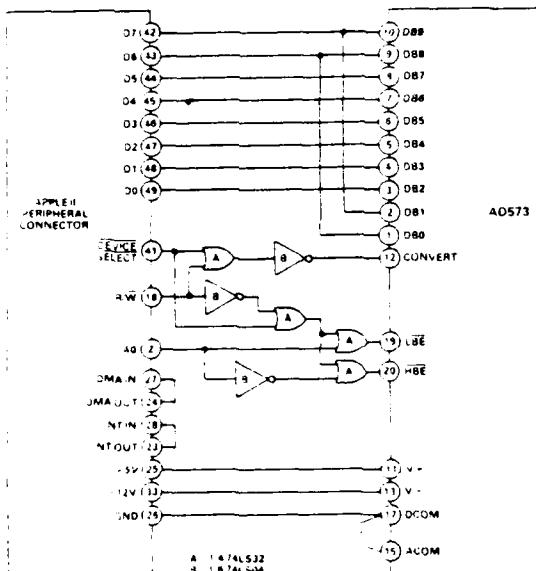


Figure 16. AD573 Interface to Apple II

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 3FCA8, 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 \overline{RW} 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.

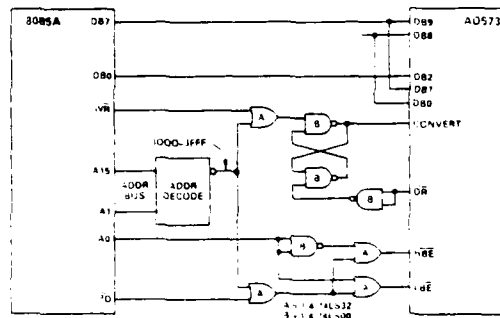


Figure 17. AD573-8085A Interface Connections

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 00 ;MASK LOWER 6 BITS
      MOV E,A ;STORE CLEAN LOW BYTE IN E
      INR E ;LOAD HIGH BYTE ADDRESS
      MOV D,M ;MOVE HIGH BYTE TO D
      RET ;EXIT
    
```

AD544 Operational Amplifier Data Sheet



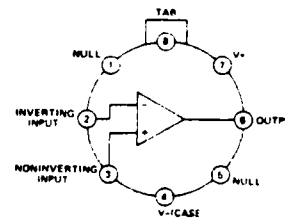
High Speed Implanted FET-Input Op Amp

AD544

FEATURES

- Low Bias Current: 25pA max, warmed-up
- Low Offset Voltage: 500 μ V max
- Low Offset Voltage Drift: 5 μ V/ $^{\circ}$ C max
- Low Input Voltage Noise: 2 μ V p-p
- Low Quiescent Current: 2.5mA max
- High Slew Rate: 13V/ μ s
- Fast Settling to \pm 0.01%: 3 μ s
- 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 (5 μ V/ $^{\circ}$ C max for "L", 10 μ V/ $^{\circ}$ 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 (2 μ V 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 $^{\circ}$ C temperature range and the "S" over the -55 $^{\circ}$ C to +125 $^{\circ}$ 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 3 μ V/ $^{\circ}$ C per mV of offset nulled.)
5. Low voltage noise (2 μ V, p-p), and low offset voltage drift (5 μ V/ $^{\circ}$ C) enhance the AD544's performance as a precision op amp.
6. The high slew rate (13.0V/ μ s) and fast settling time to 0.01% (3.0 μ s) make the AD544 ideal for D/A, A/D, sample-and-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			Units
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
OPEN LOOP GAIN¹ $V_{OL} = \pm 10V, R_L \geq 2k\Omega$ T_{min} to $T_{max}, R_L = 2k\Omega$	30,000 20,000			50,000 40,000			50,000 40,000			50,000 20,000			V/V V/V
OUTPUT CHARACTERISTICS Voltage @ $R_L = 2k\Omega, T_{min}$ to T_{max} Voltage @ $R_L = 10k\Omega, T_{min}$ to T_{max} Short Circuit Current	± 10 ± 12	± 12 ± 13		± 10 ± 12	± 12 ± 13		± 10 ± 12	± 12 ± 13		± 10 ± 12	± 12 ± 13		V V mA
FREQUENCY RESPONSE Unity Gain Small Signal Full Power Response Slew Rate, Unity Gain Settling Time to 0.01% Total Harmonic Distortion		2.0 200			2.0 200			2.0 200			2.0 200		MHz kHz V/μs μs %
INPUT OFFSET VOLTAGE² Initial Offset Input Offset Voltage vs. Temp. or T_{min} to T_{max} Input Offset Voltage vs. Supply, T_{min} to T_{max}			2.0			1.0			0.5			1.0	mV μV/°C μV/V
INPUT BIAS CURRENT³ Either Input Offset Current		10 5	50		10 2	25		10 2	25		10 2	25	pA pA
INPUT IMPEDANCE Differential ⁴ Common Mode		10^{12} $10^{12.3}$			10^{12} $10^{12.3}$			$10^{12.6}$ $10^{12.3}$			$10^{12.6}$ $10^{12.3}$		MΩpF MΩpF
INPUT VOLTAGE RANGE Differential Common Mode Common Mode Rejection			± 20 ± 12			± 20 ± 12			± 20 ± 12			± 20 ± 12	V V dB
INPUT NOISE Voltage 0.1Hz to 10Hz $f = 10Hz$ $f = 100Hz$ $f = 1kHz$ $f = 10kHz$		2 35 22 18 16			2 35 22 18 16			2 35 22 18 16			2 35 22 18 16		μV _{rms} μV _{rms} /√Hz nV _{rms} /√Hz nV _{rms} /√Hz nV _{rms} /√Hz
POWER SUPPLY Rated Performance Operating Quiescent Current		± 5 1.8	± 15 2.5		± 5 1.8	± 15 2.5		± 5 1.8	± 15 2.5		± 5 1.8	± 15 2.5	V V mA
TEMPERATURE RANGE Operating, Rated Performance Storage		0 -65	+70 +150		0 -65	+70 +150		0 -65	+70 +150		-55 -65	+125 +150	°C °C
PACKAGE⁵ TO-99 Style (H08B)	AD544JH			AD544KH			AD544LH			AD544SH			

NOTES

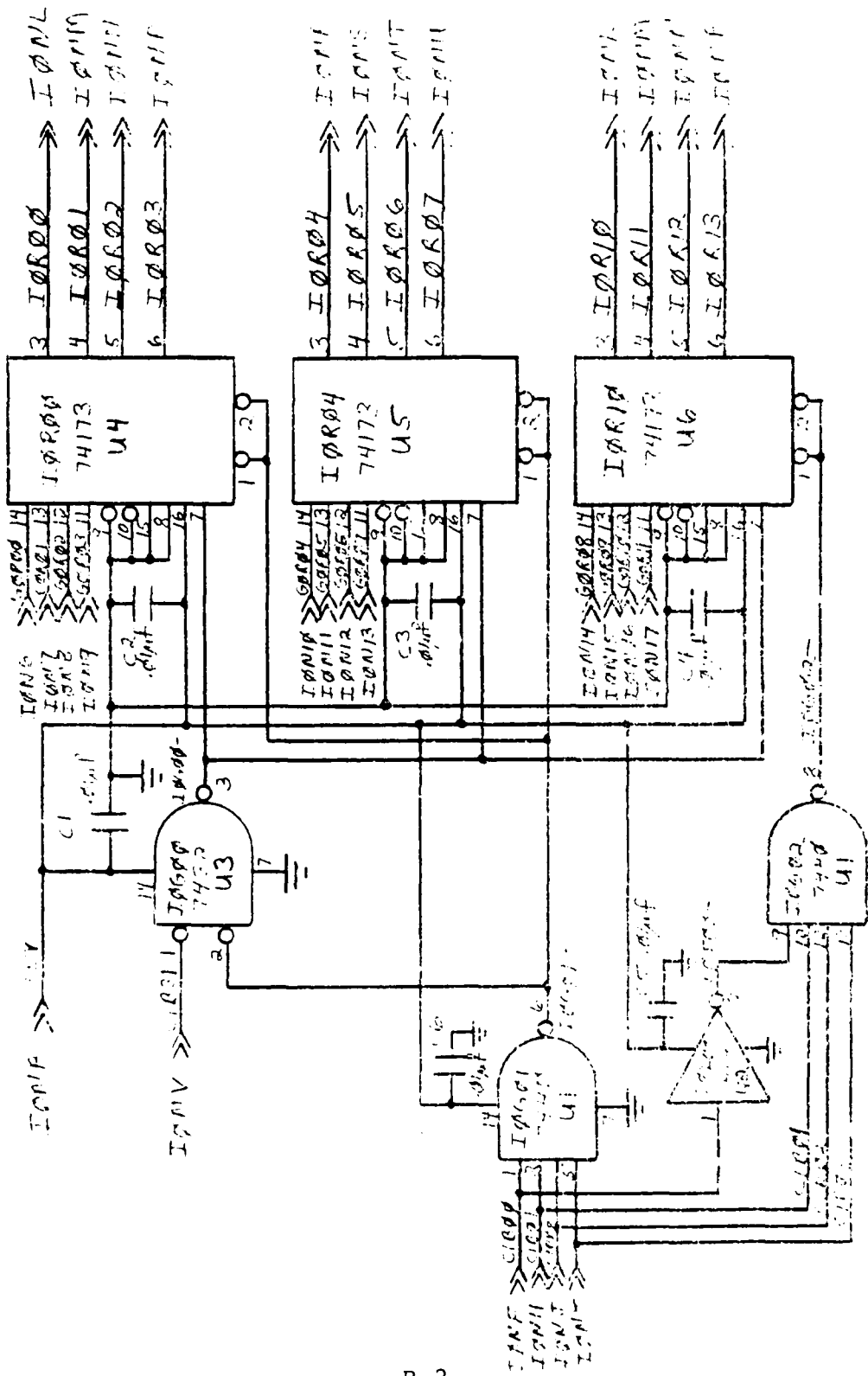
¹Open Loop Gain is specified with V_{OL} both nulled and un-nulled
²Input Offset Voltage specifications are guaranteed after 5 minutes of operation at $T_A = +25^\circ C$
³Bias Current specifications are guaranteed at maximum at either input after 5 minutes of operation at $T_A = +25^\circ C$. For higher temperatures, the current doubles every 10°C.
⁴Defined as voltage between inputs, such that neither exceeds $\pm 10V$ from ground.

⁵See Section 19 for package outline information.
 Specifications subject to change without notice.

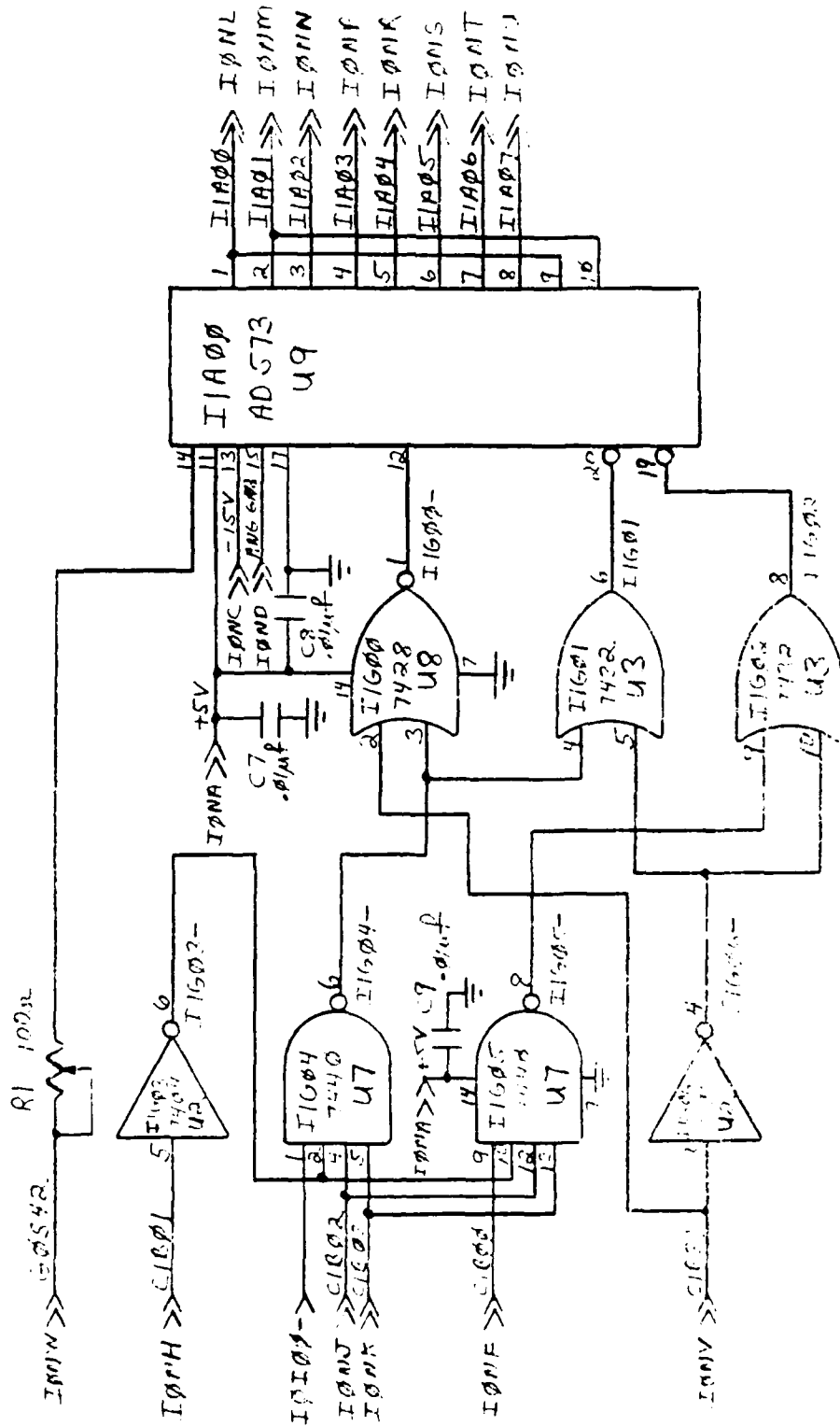
Specifications shown in boldface are tested on all production units at final electrical test. Results from those tests are used to calculate outgoing quality levels. All min and max specifications are guaranteed, although only those shown in boldface are tested on all production units.

APPENDIX B

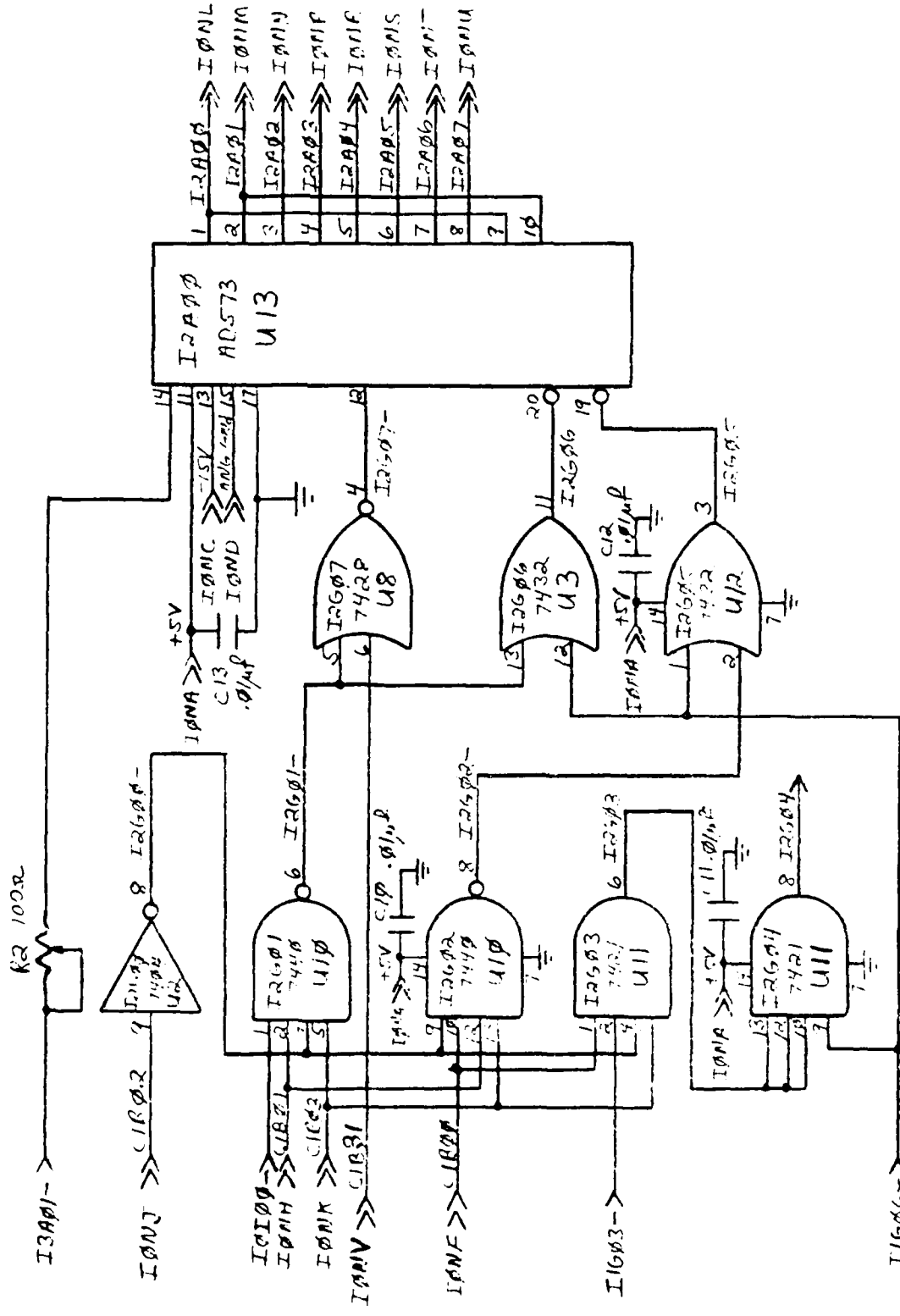
Digital Interface Schematic Diagrams B-2
Circuit Card I: Digital Interface Parts List B-7
Digital Interface Device Layout B-10



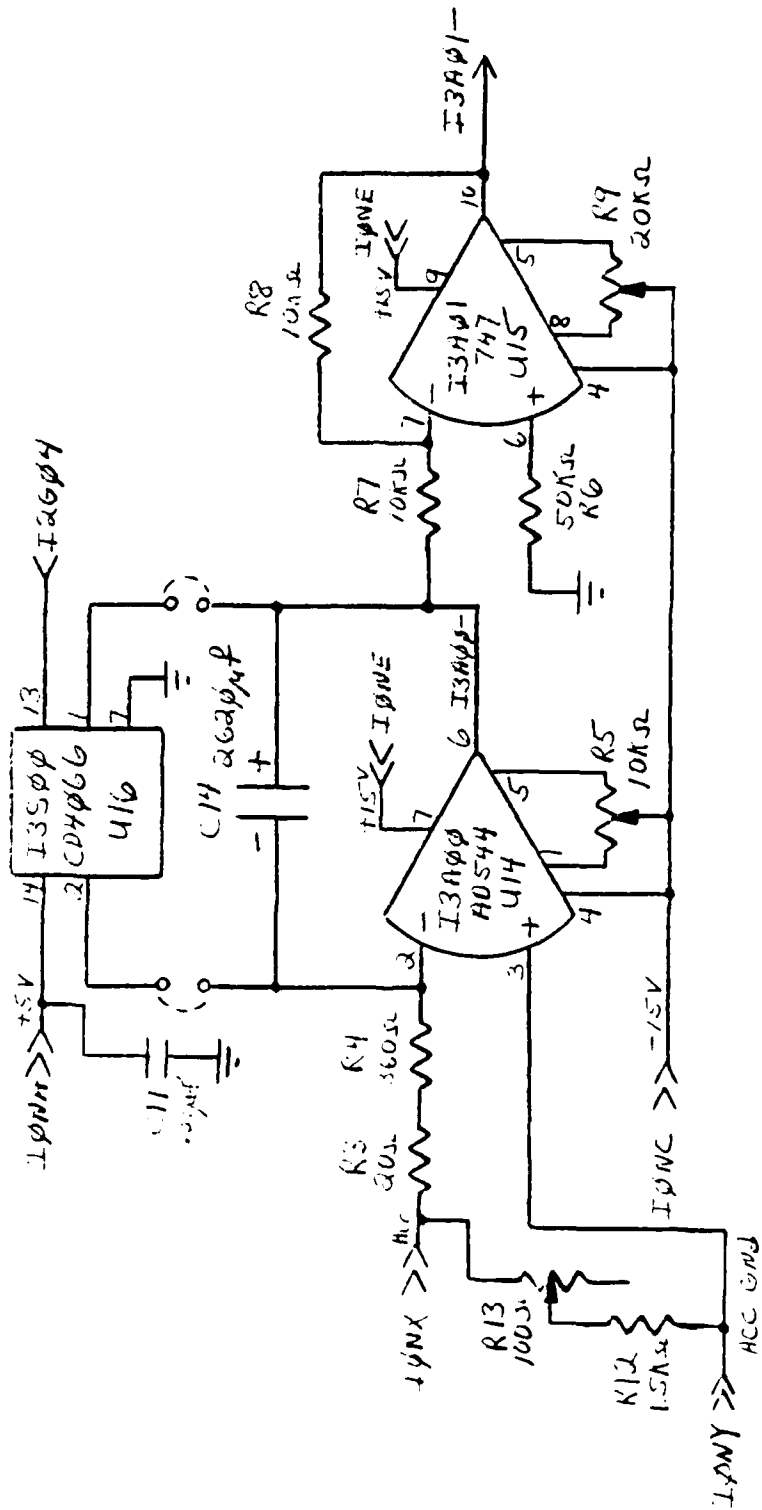
DIGITAL INTERFACE SCHEMATIC DIAGRAMS



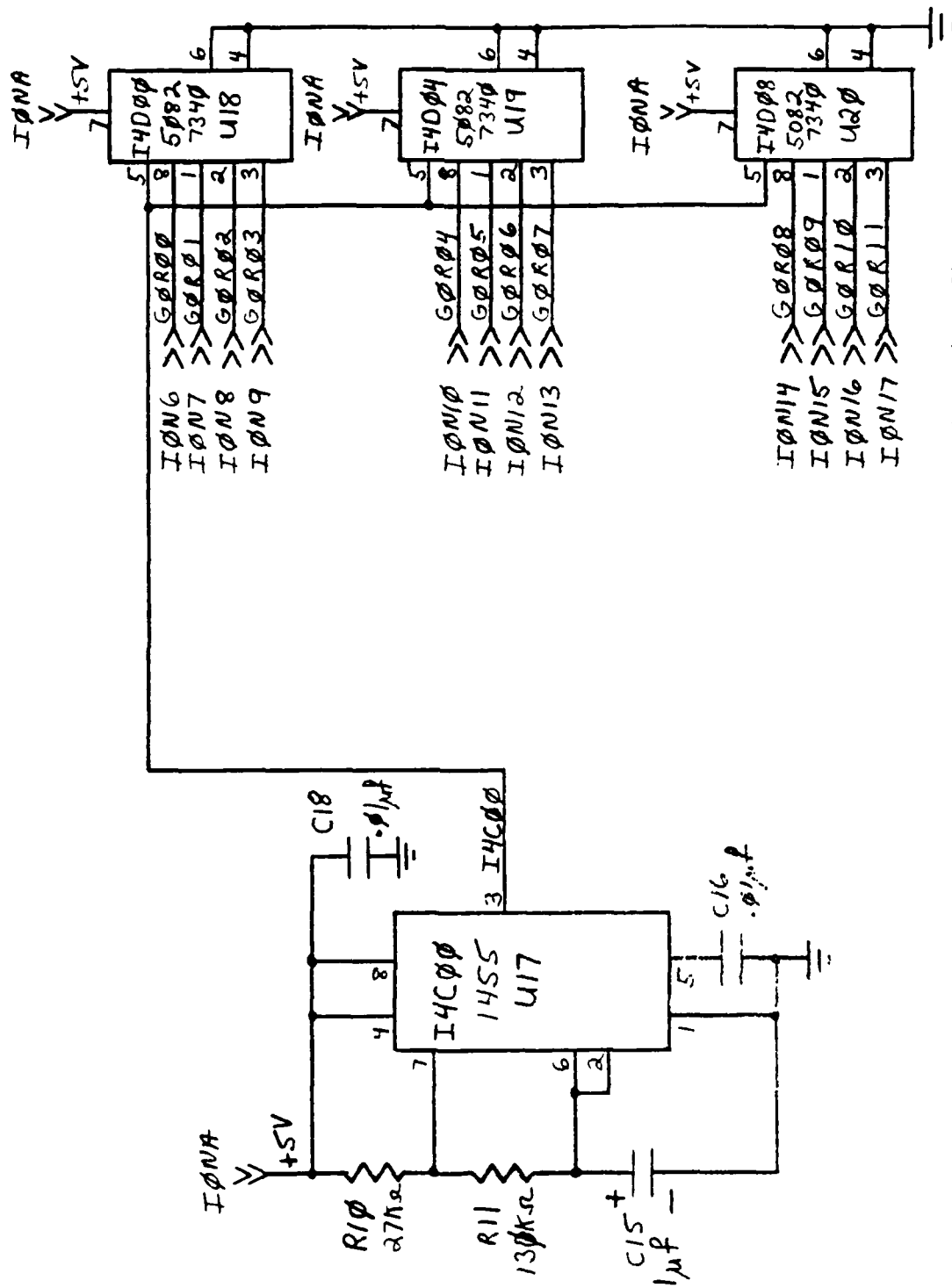
DIGITAL INTERFACE SCHEMATIC DIAGRAMS (continued)



DIGITAL INTERFACE SCHEMATIC DIAGRAMS (continued)



DIGITAL INTERFACE SCHEMATIC DIAGRAMS (continued)



DIGITAL INTERFACE SCHEMATIC DIAGRAMS (continued)

Circuit Card I: Digital Interface
Parts List

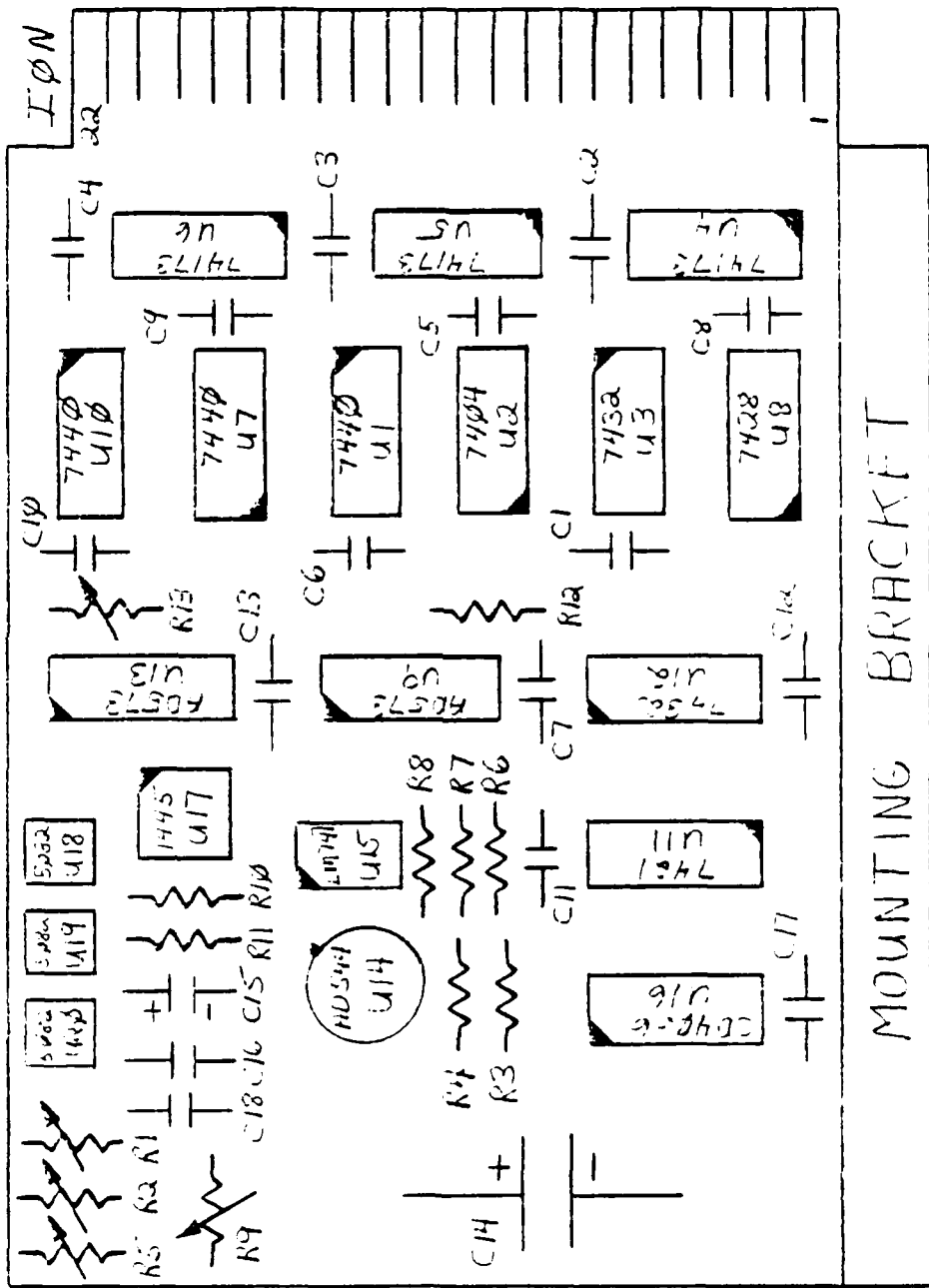
Part Number	Type	Schematic Reference #
7440	Dual 4-Input Positive NAND Buffers	U1 U7 U10
7404	Hex Inverters	U2
7432	Quadruple 2-Input Positive-OR Gates	U3 U12
74173	4-Bit D-Type Register With 3-State Outputs	U4 U5 U6
7428	Quadruple 2-Input Positive-NOR Gates	U8
AD 573	10-Bit Analog to Digital Converter	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

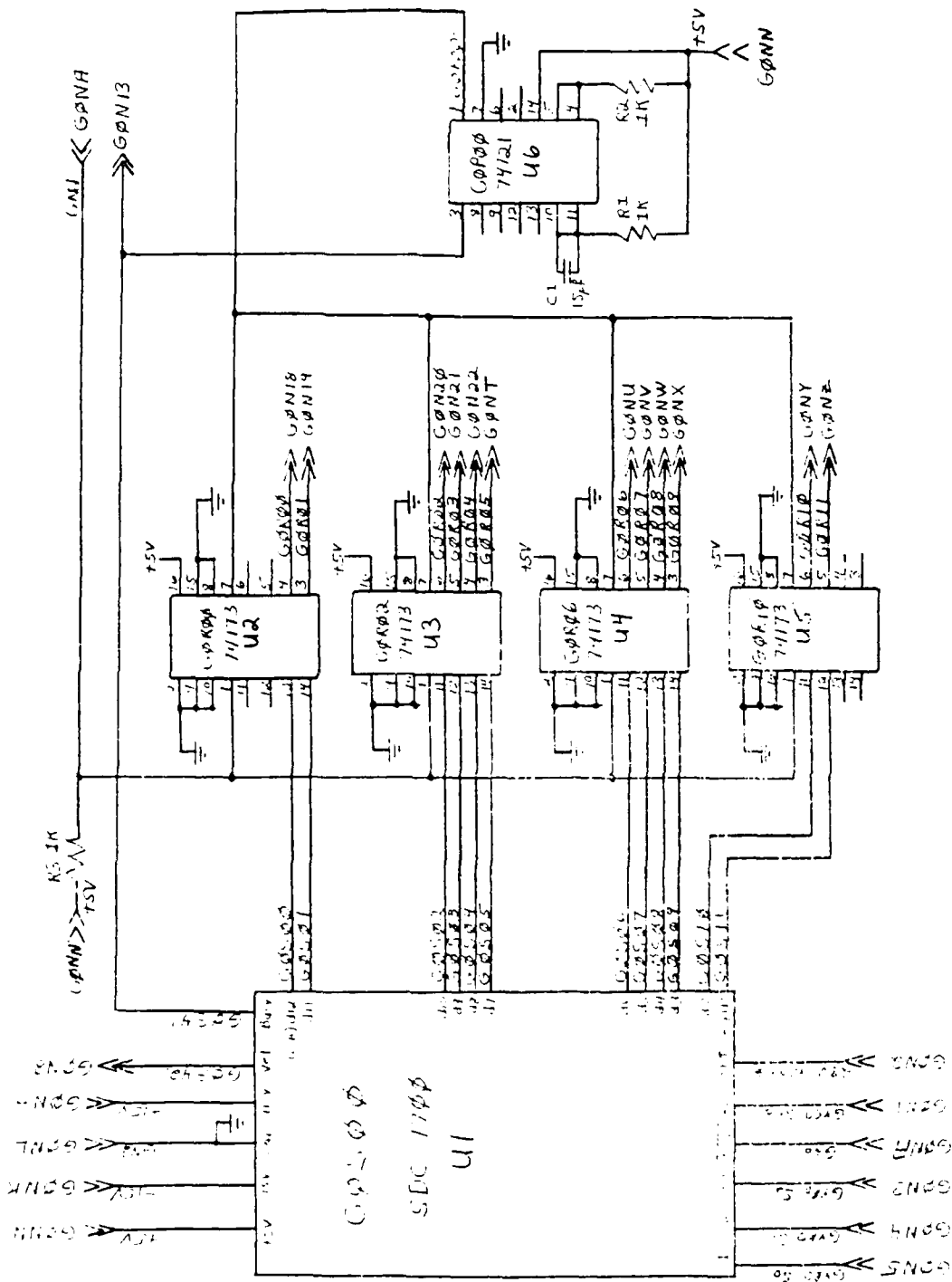


DIGITAL INTERFACE DEVICE LAYOUT

MOUNTING BRACKET

APPENDIX C

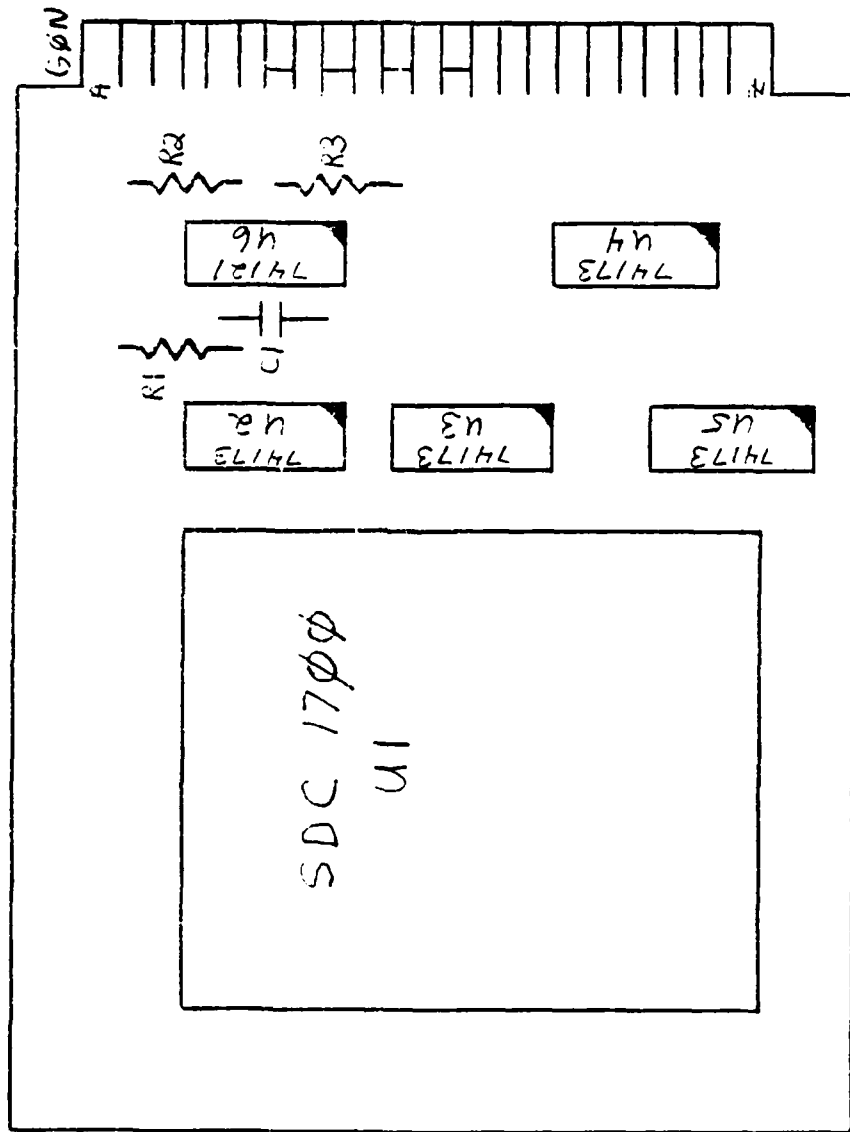
Syncro to Digital Schematic Diagram C-2
Circuit Card G: Syncro to Digital Parts List C-3
Syncro to Digital Device Layout C-4



SYNCR0 TO DIGITAL SCHEMATIC DIAGRAM

Circuit Card G: Syncro to Digital
Parts List

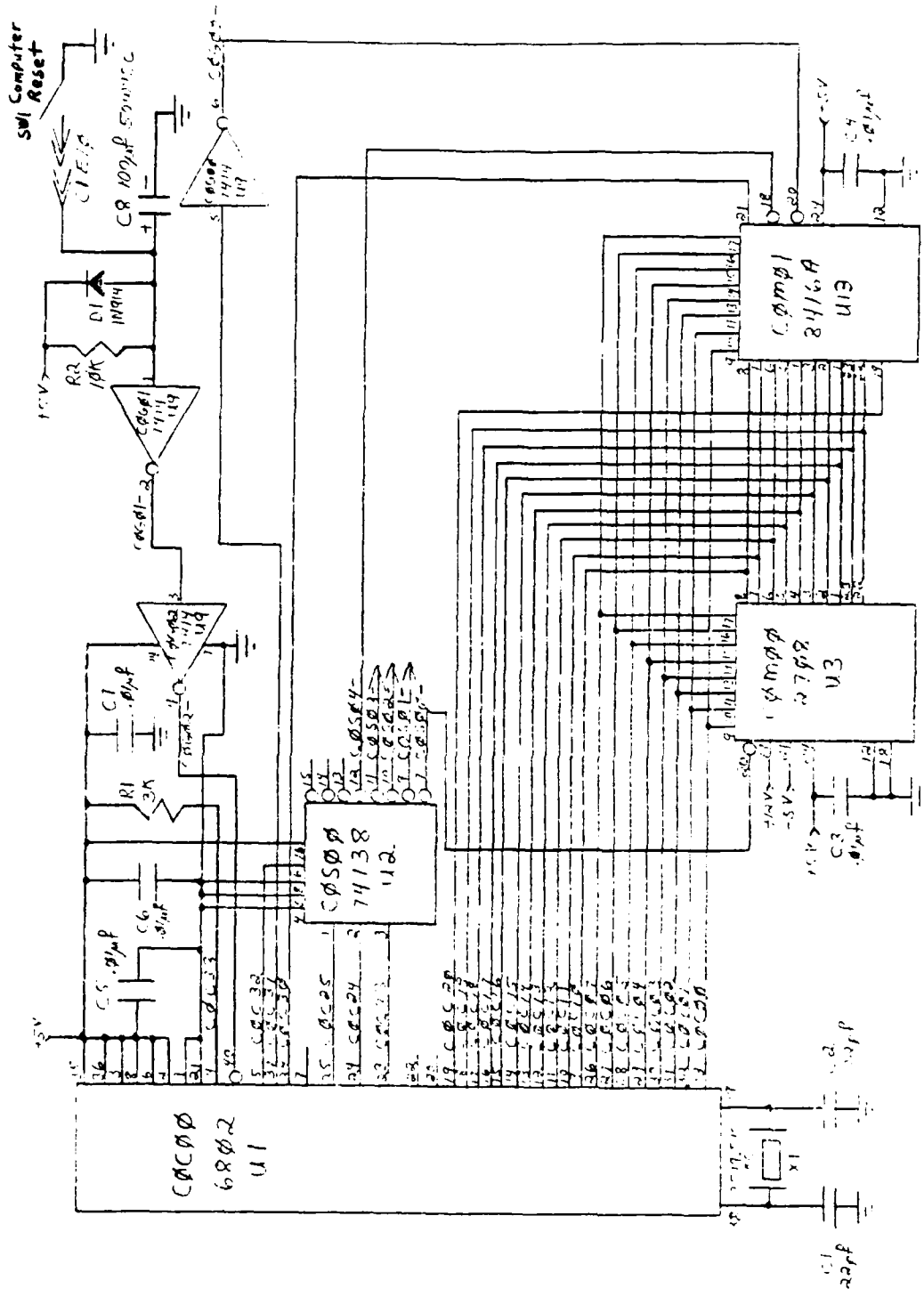
Part Number	Type	Schematic Reference #
SDC 1700	Syncro to Digital Converter	U1
74173	4-Bit D-Type Register With 3-State Outputs	U2 U3 U4 U5
74121	Monostable Multivibrator	U6
1K	Resistor (Ohms)	R1 R2 R3
15 pico	Capacitor (Farads)	C1



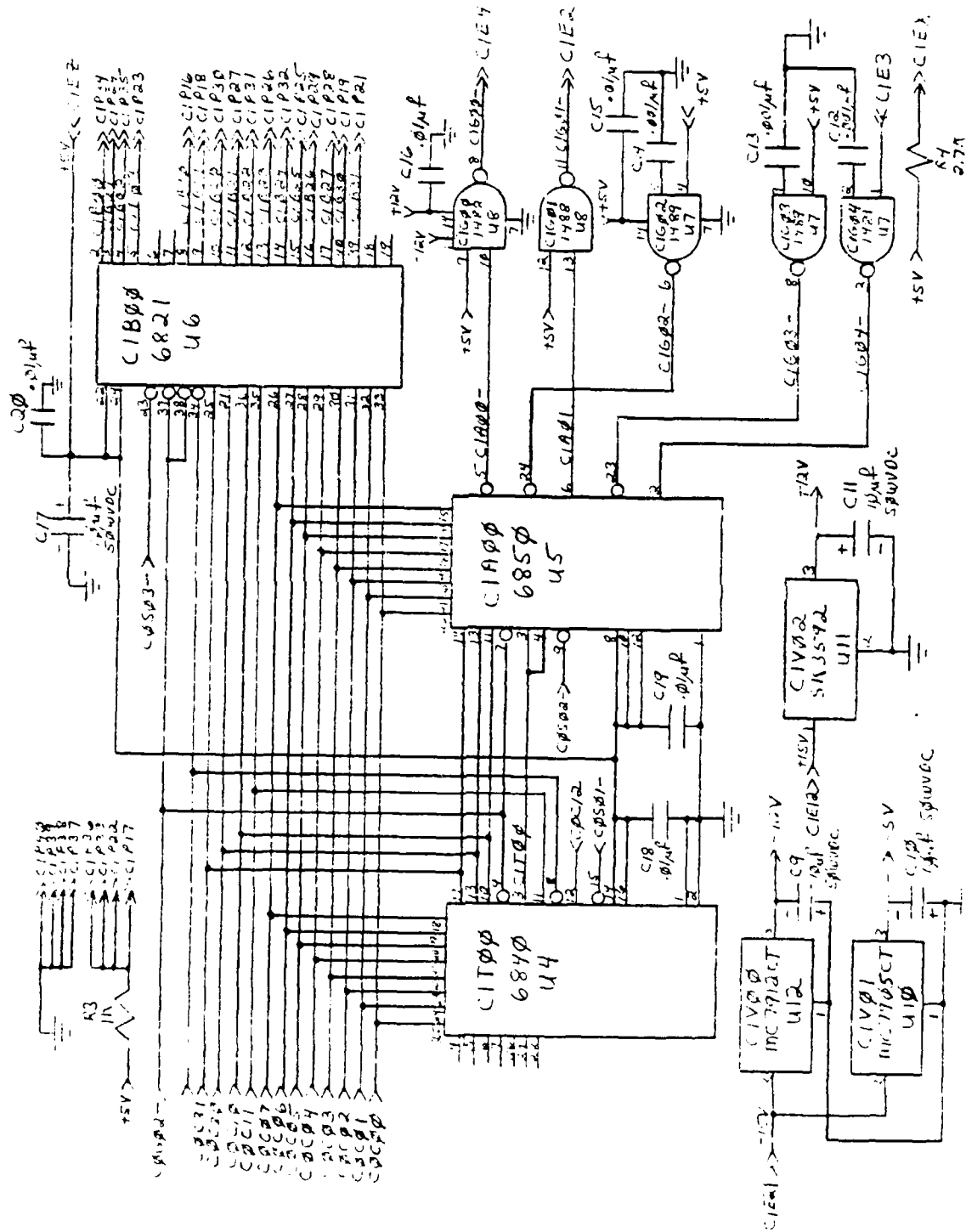
SYNCRO TO DIGITAL DEVICE LAYOUT

APPENDIX D

GYRAC Computer Schematic Diagrams D-2
Circuit Card C: Computer Controller Parts List D-4
GYRAC Computer Memory Map D-7
GYRAC Computer Device Layout D-9



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)

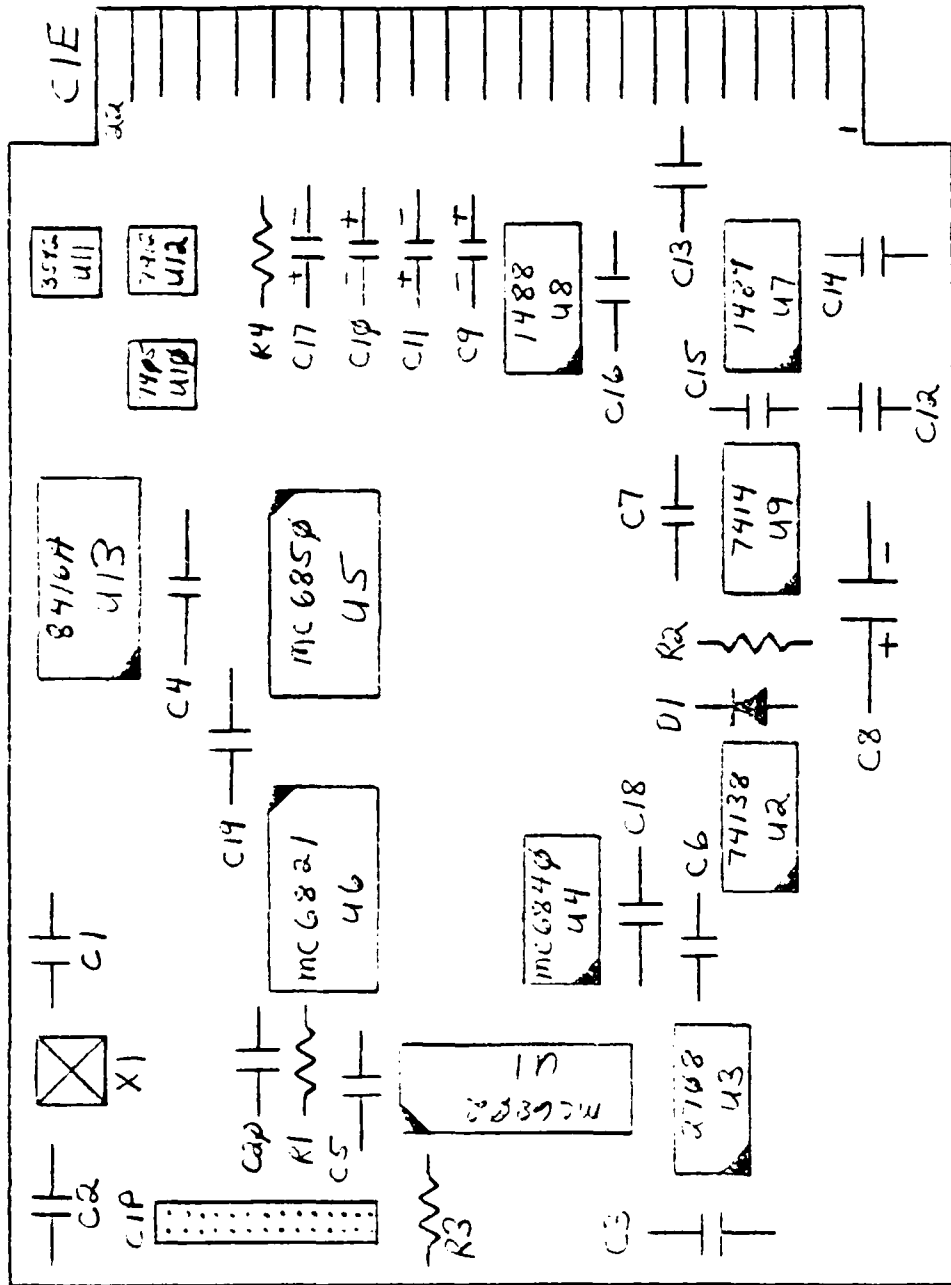
Part Number	Type	Schematic Reference #
10 Micro	Capacitor (Farads) 50 WVDC	C9 C10 C11 C17
0.001 Micro	Capacitor (Farads)	C12 C13 C14

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

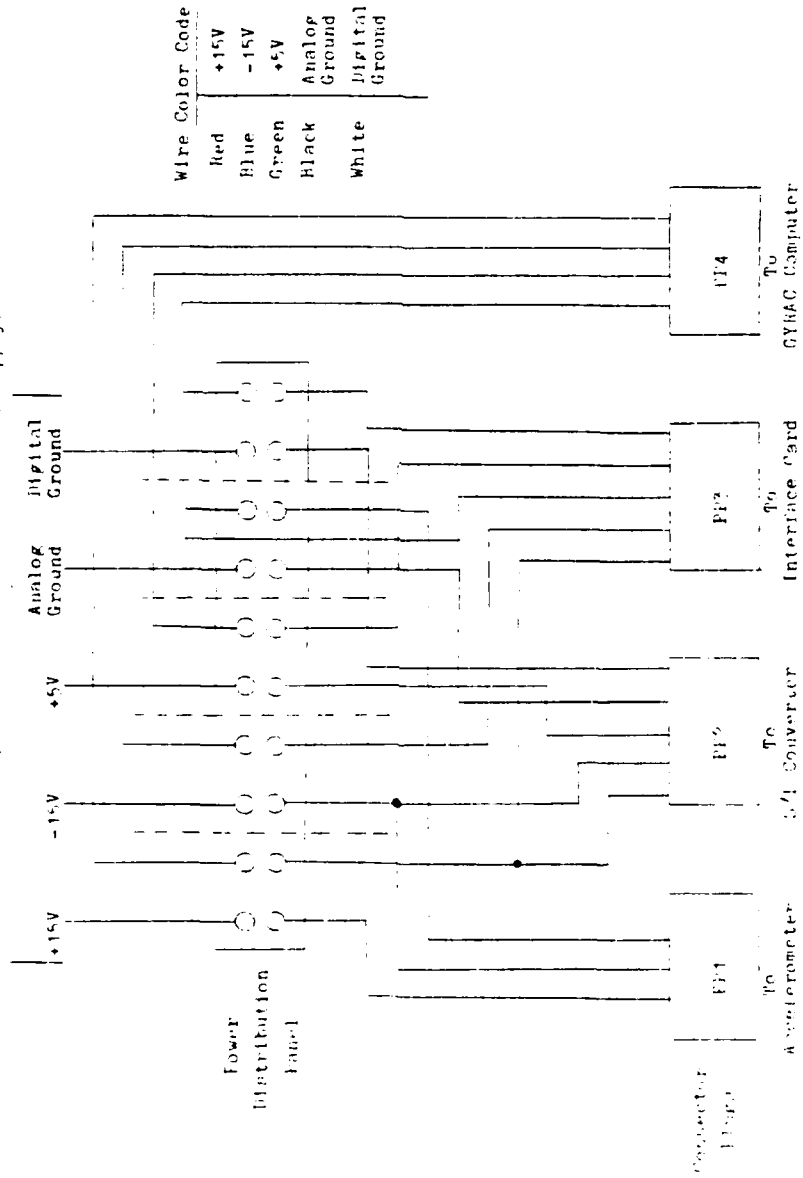


GYRAC COMPUTER DEVICE LAYOUT

APPENDIX E

GYRAC Power Panel	E-2
GON: Syncro to Digital Edge Connector	E-3
C1P: GYRAC Computer Sensor Bus Connector	E-4
C1E: GYRAC Computer Edge Connector	E-5
ION: Digital Interface Edge Connector	E-6
H89 to GYRAC RS-232 Cable	E-7
H89 to NAV T or Drive Computer RS-232 Cable	E-8
GYRAC to NAV L Computer RS-232 Cable	E-9
Drive Computer to NAV X RS-232 Cable	E-10
GYRAC Wiring Harness	E-11

From Gyro Base Assembly (except +5V in from external Supply)



Wire Color	Code
Red	+15V
Blue	-15V
Green	+5V
Black	Analog Ground
White	Digital Ground

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

Nav T/Drive Computer

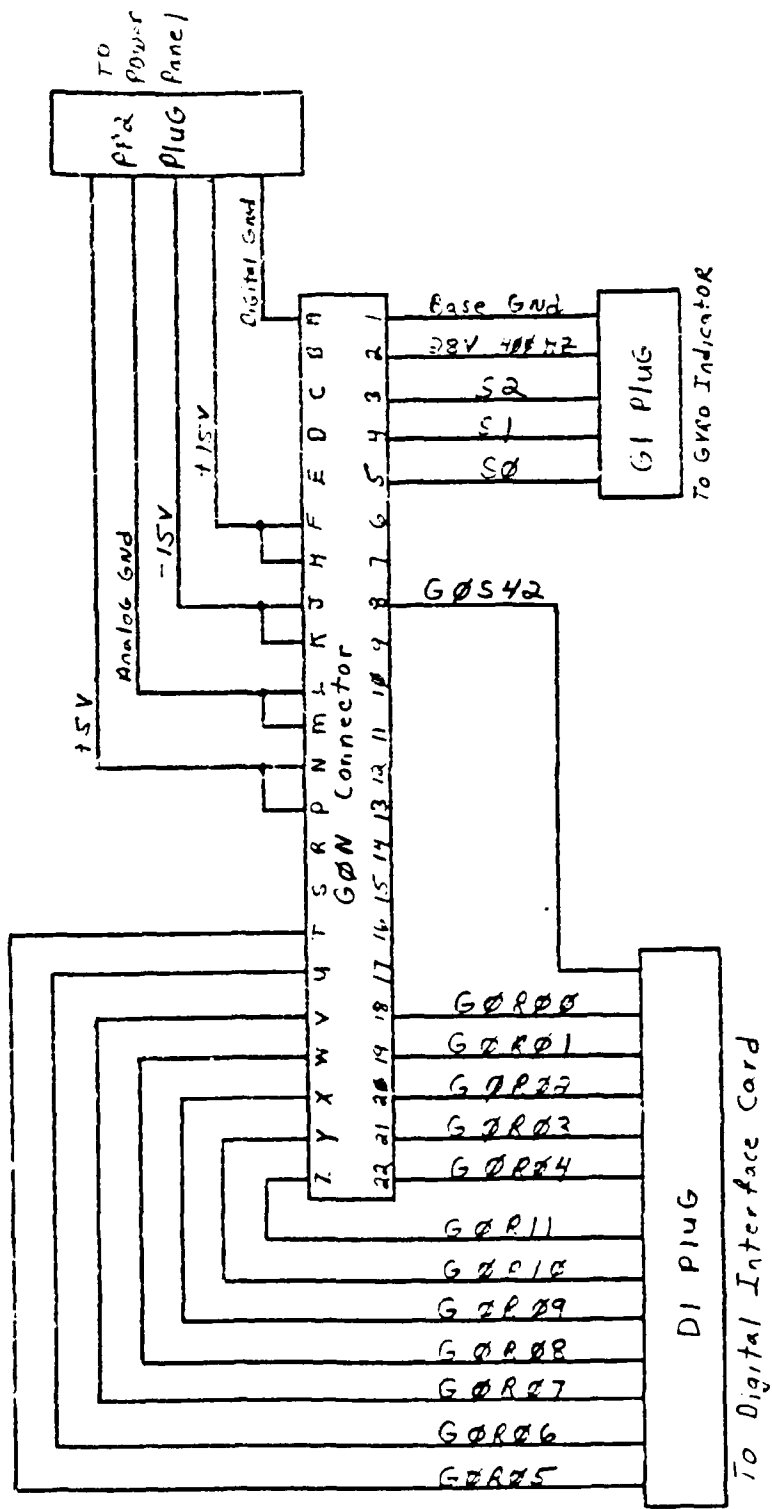
H89 Connector Pin Number	Signal Description	Connector Pin Number
2	Tx Data	3
3	Rx Data	2
7	Ground	7

GYRAC to Nav L Computer RS-232 Cable

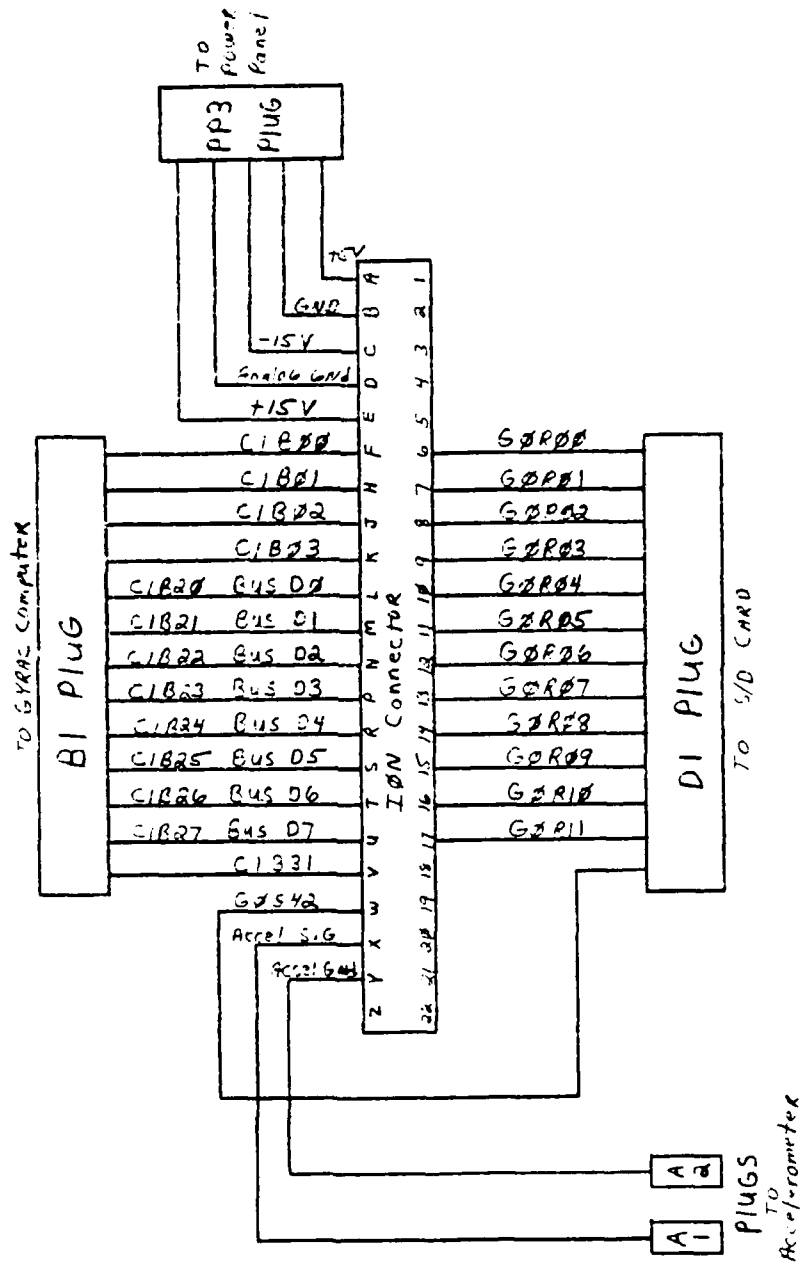
GYRAC Connector Pin Number	Signal Description	Nav L Connector Pin Number
2	Tx Data	3
3	Rx Data	2
7	Ground	7

Drive Computer to Nav X RS-232 Cable

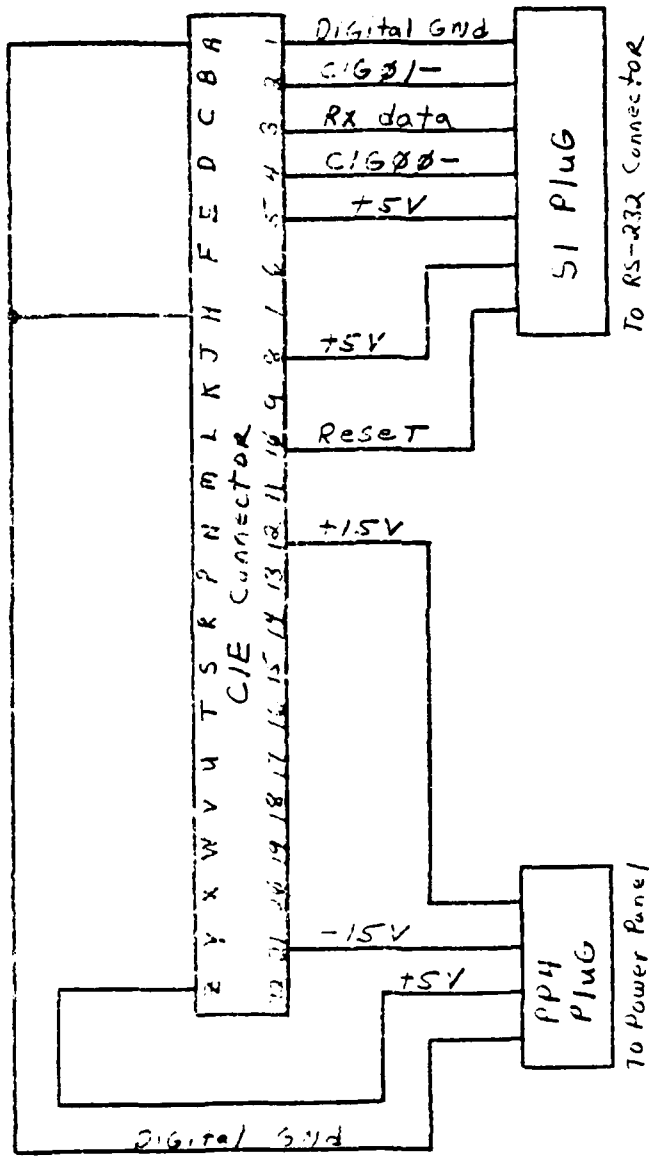
Drive Computer Connector Pin Number	Signal Description	Nav X Connector Pin Number
2	Tx Data	3
3	Rx Data	2
7	Ground	7



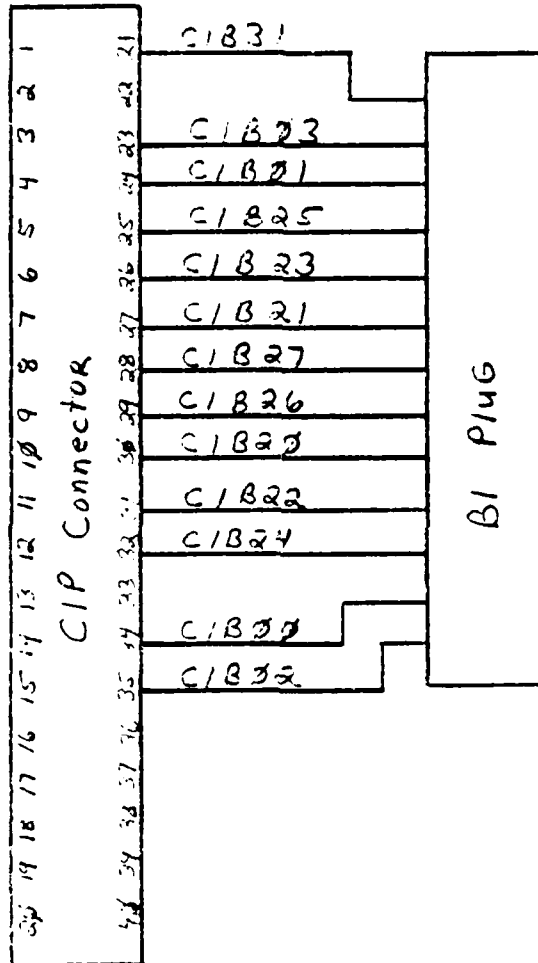
GYRAC WIRING HARNESS



GYRAC WIRING HARNESS (continued)



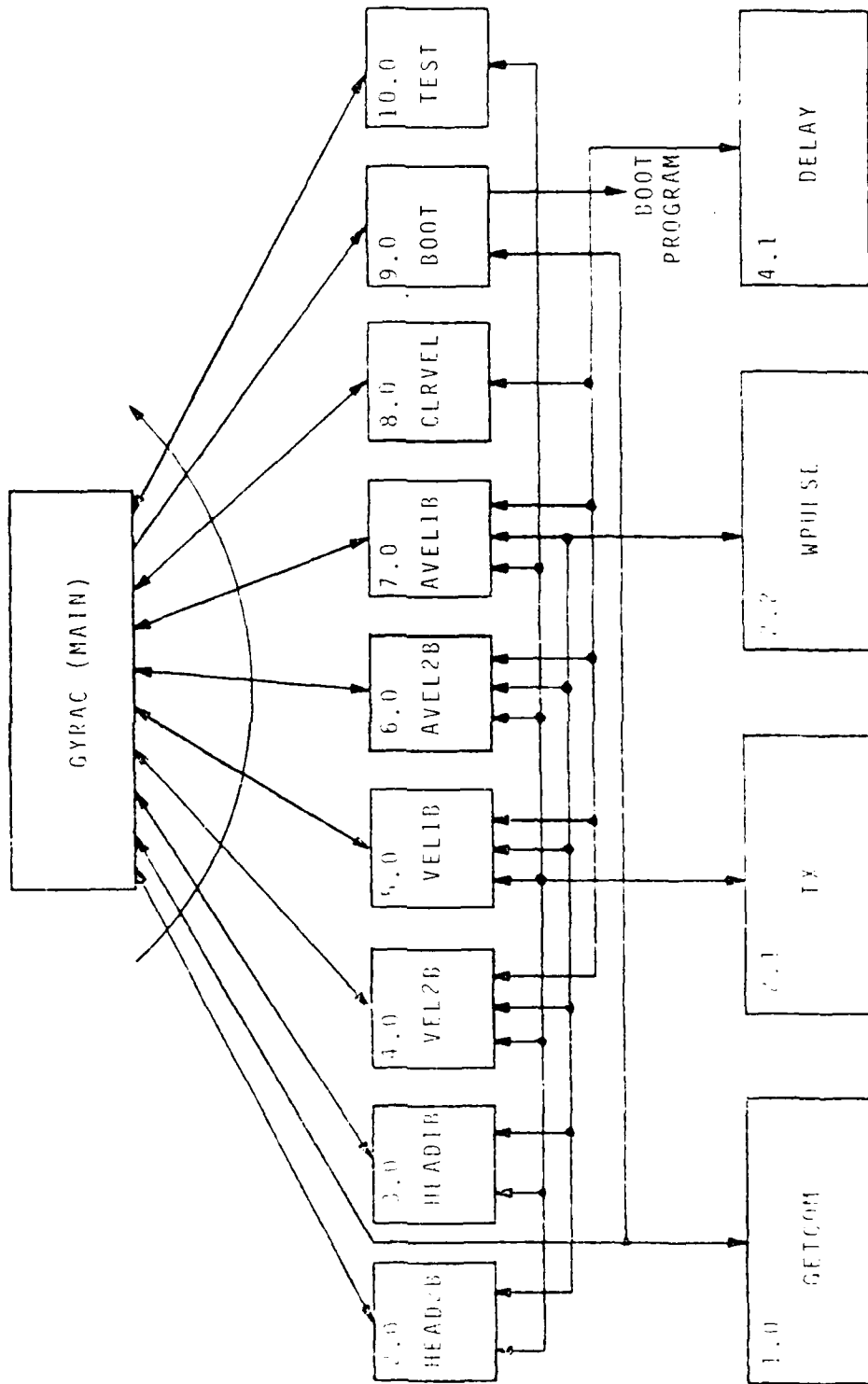
GYRAC WIRING HARNESS (continued)



GYRAC WIRING HARNESS (continued)

APPENDIX F

GYRAC.A Structure Chart F-2
GYRAC.A Program Listing F-3
GYRAC.A Operating Instructions F-21



GYRAC.A STRUCTURE CHART

GYRAC.A PROGRAM LISTING

DATE: REVISED
 10/11/71
 TITLE: GYRO CONTROL PROGRAM
 PROGRAM: GYRAC
 AUTHOR: CAPT WILLIAM G. BAKER JR.
 PROJECT: GYRO AND ACCELEROMETER LABED NAVIGATION SYSTEM FOR A
 MOBILE AIRCRAFT ROCKET (THESIS)
 OPERATING SYSTEM: IBM 360/15M (11.41 MAGNOLIA MICROSYSTEM 11.1)
 LANGUAGE: RUCS-A ASSEMBLER WITH VIRTUAL DEVICES 11.11
 USE: This program is assembled and then burned into an EPROM
 for operation and control of the GYRAC subsystem.
 CONTENTS: MAIN
 getin
 head0t
 head1t
 head2t
 head3t
 ave0tb
 ave1tb
 ave2tb
 ave3tb
 delay
 FUNCTION: This program provides an RE-202 test, a RA4 boot
 loader, the capability to read 1 or 2 bytes of
 heading, angular velocity, and velocity and transient
 one tape over an RE-202 interface, and can clear the
 velocity to start in the accelerometer integrator.

000	000	000	000	000	000
001	001	001	001	001	001
002	002	002	002	002	002
003	003	003	003	003	003
004	004	004	004	004	004
005	005	005	005	005	005
006	006	006	006	006	006
007	007	007	007	007	007
008	008	008	008	008	008
009	009	009	009	009	009
010	010	010	010	010	010
011	011	011	011	011	011
012	012	012	012	012	012
013	013	013	013	013	013
014	014	014	014	014	014
015	015	015	015	015	015
016	016	016	016	016	016
017	017	017	017	017	017
018	018	018	018	018	018
019	019	019	019	019	019
020	020	020	020	020	020
021	021	021	021	021	021
022	022	022	022	022	022
023	023	023	023	023	023
024	024	024	024	024	024
025	025	025	025	025	025
026	026	026	026	026	026
027	027	027	027	027	027
028	028	028	028	028	028
029	029	029	029	029	029
030	030	030	030	030	030
031	031	031	031	031	031
032	032	032	032	032	032
033	033	033	033	033	033
034	034	034	034	034	034
035	035	035	035	035	035
036	036	036	036	036	036
037	037	037	037	037	037
038	038	038	038	038	038
039	039	039	039	039	039
040	040	040	040	040	040
041	041	041	041	041	041
042	042	042	042	042	042
043	043	043	043	043	043
044	044	044	044	044	044
045	045	045	045	045	045
046	046	046	046	046	046
047	047	047	047	047	047
048	048	048	048	048	048
049	049	049	049	049	049
050	050	050	050	050	050
051	051	051	051	051	051
052	052	052	052	052	052
053	053	053	053	053	053
054	054	054	054	054	054
055	055	055	055	055	055
056	056	056	056	056	056
057	057	057	057	057	057
058	058	058	058	058	058
059	059	059	059	059	059
060	060	060	060	060	060
061	061	061	061	061	061
062	062	062	062	062	062
063	063	063	063	063	063
064	064	064	064	064	064
065	065	065	065	065	065
066	066	066	066	066	066
067	067	067	067	067	067
068	068	068	068	068	068
069	069	069	069	069	069
070	070	070	070	070	070
071	071	071	071	071	071
072	072	072	072	072	072
073	073	073	073	073	073
074	074	074	074	074	074
075	075	075	075	075	075
076	076	076	076	076	076
077	077	077	077	077	077
078	078	078	078	078	078
079	079	079	079	079	079
080	080	080	080	080	080
081	081	081	081	081	081
082	082	082	082	082	082
083	083	083	083	083	083
084	084	084	084	084	084
085	085	085	085	085	085
086	086	086	086	086	086
087	087	087	087	087	087
088	088	088	088	088	088
089	089	089	089	089	089
090	090	090	090	090	090
091	091	091	091	091	091
092	092	092	092	092	092
093	093	093	093	093	093
094	094	094	094	094	094
095	095	095	095	095	095
096	096	096	096	096	096
097	097	097	097	097	097
098	098	098	098	098	098
099	099	099	099	099	099
100	100	100	100	100	100

THE FOLLOWING
 Reproduced from
 best available copy. 6
 PAGES

GYRAC.A PROGRAM LISTING (continued)

```

B007      pm10b      equ      00150H      ;Time to divide by 10 before 10 bit or later
B008      stack     equ      0700H      ;Initial stack pointer
B009      read      equ      04H      ;PIA read port address
B010      write     equ      04H      ;PIA write port address
B011      ottdont   equ      01H      ;OT may be written, use enable block,
                                ;initial OT mode, continuous operating
                                ;mode, output enabled
B012      pdont     equ      0010H      ;Timer divide count to 5 = (0x10/0x05)
                                ;i.e. 60/10 = 6, to 14 = 140/10 = 14
B013      a11reset  equ      01H      ;Reset A01A command
B014      a11cmd    equ      05H      ;Divide by 10 mode, 8 bits data plus
                                ;1 stop bit, all interrupts disabled
B015      tone      equ      02H      ;A01A transmit data register empty
B016      rdont     equ      01H      ;A01A receive data register full
B017      headadl   equ      01H      ;Address of GYRAC heading low byte
B018      headadh   equ      02H      ;Address of GYRAC heading high byte
B019      veladl    equ      03H      ;Address of GYRAC velocity low byte
B020      veladh    equ      04H      ;Address of GYRAC velocity high byte
B021      angveladl equ      05H      ;Address of GYRAC ang vel low byte
B022      angveladh equ      06H      ;Address of GYRAC ang vel high byte
B023      v10scad   equ      07H      ;Address of GYRAC vel core clear device
B024      temp      equ      0800H      ;Address of RAM scratch pad
B025      rambot    equ      0900H      ;Address of bottom of 2K ram block
B026      blocks   equ      0001H      ;Address of number of blocks of data
B027      bytes    equ      0001H      ;Address of number of bytes in a block
B028      char     equ      0001H      ;Address of current ASCII character
B029      charcnt  equ      0004H      ;Address of current character count
B030      tdel     equ      05H      ;Timing delay loop variable

B031      begin     ;Beginning of main program
B032      set       ;Disable interrupts
B033      init      ;Initialize stack pointer to top of Ram
B034      4F        ;Init bank in A reg
B035      07 00 01  ;Clear PIA control register A
B036      07 00 01  ;Clear PIA control register B
B037      07 00 00  ;Make PIA port B an input port
B038      40        ;PIA
B039      07 00 00  ;Make PIA port A an output port
B040      40        ;read
B041      07 00 01  ;Prepare port A for data out & read era
B042      07 00 00  ;Prepare port B for data input

B043      07 00 01  ;Get ATM timer 2 command
B044      07 00 01  ;Write ATM timer 2 command
B045      0B 00 02  ;Get timer 2 count
B046      00 00 04  ;Get timer to divide by a byte of 147140
B047      4F        ;PIA
B048      07 00 00  ;start timer

B049      07 00 01  ;a11a
B050      07 00 00  ;a11a
B051      07 00 01  ;a11a
B052      07 00 00  ;a11a
B053      07 00 00  ;a11a

B054      00 00 01  ;Clear velocity scratch
B055      00 00 00  ;Set command to 00000000 = 0000
B056      07 00 01  ;PIA
B057      11        ;PIA
B058      07 00 01  ;PIA

```

GYRAC.A PROGRAM LISTING (continued)

M0140	01 40	1fab	#0100	:Put command C in B register
M0141	01 41	0fa		:Is command C in A register?
M0142	01 42	0ed	0100	:If yes go execute command C
M0143	01 43	1fab	#0101	:Put command D in B register
M0144	01 44	0fa		:Is command D in A register?
M0145	01 45	0ed	0101	:If yes go execute command D
M0146	01 46	1fab	#0102	:Put command E in B register
M0147	01 47	0fa		:Is command E in A register?
M0148	01 48	0ed	0102	:If yes go execute command E
M0149	01 49	1fab	#0103	:Put command F in B register
M0150	01 50	0fa		:Is command F in A register?
M0151	01 51	0ed	0103	:If yes go execute command F
M0152	01 52	1fab	#0104	:Put command G in B register
M0153	01 53	0fa		:Is command G in A register?
M0154	01 54	0ed	0104	:If yes go execute command G
M0155	01 55	1fab	#0105	:Put command H in B register
M0156	01 56	0fa		:Is command H in A register?
M0157	01 57	0ed	0105	:If yes go execute command H
M0158	01 58	1fab	#0106	:Put command I in B register
M0159	01 59	0fa		:Is command I in A register?
M0160	01 60	0ed	0106	:If yes go execute command I
M0161	01 61	1fab	#0107	:Put command J in B register
M0162	01 62	0fa		:Is command J in A register?
M0163	01 63	0ed	0107	:If yes go execute command J
M0164	01 64	1fab	#0108	:Put command K in B register
M0165	01 65	0fa		:Is command K in A register?
M0166	01 66	0ed	0108	:If yes go execute command K
M0167	01 67	1fab	#0109	:Put command L in B register
M0168	01 68	0fa		:Is command L in A register?
M0169	01 69	0ed	0109	:If yes go execute command L
M0170	01 70	1fab	#0110	:Put command M in B register
M0171	01 71	0fa		:Is command M in A register?
M0172	01 72	0ed	0110	:If yes go execute command M
M0173	01 73	1fab	#0111	:Put command N in B register
M0174	01 74	0fa		:Is command N in A register?
M0175	01 75	0ed	0111	:If yes go execute command N
M0176	01 76	1fab	#0112	:Put command O in B register
M0177	01 77	0fa		:Is command O in A register?
M0178	01 78	0ed	0112	:If yes go execute command O
M0179	01 79	0fa	0100	:If all set commands are set to 1
M0180	01 80	0ed	0112	:If yes go execute command O
M0181	01 81	0ed	0113	:If yes go execute command O
M0182	01 82	0ed	0114	:If yes go execute command O
M0183	01 83	0ed	0115	:If yes go execute command O
M0184	01 84	0ed	0116	:If yes go execute command O
M0185	01 85	0ed	0117	:If yes go execute command O
M0186	01 86	0ed	0118	:If yes go execute command O
M0187	01 87	0ed	0119	:If yes go execute command O
M0188	01 88	0ed	0120	:If yes go execute command O
M0189	01 89	0ed	0121	:If yes go execute command O
M0190	01 90	0ed	0122	:If yes go execute command O
M0191	01 91	0ed	0123	:If yes go execute command O
M0192	01 92	0ed	0124	:If yes go execute command O
M0193	01 93	0ed	0125	:If yes go execute command O
M0194	01 94	0ed	0126	:If yes go execute command O
M0195	01 95	0ed	0127	:If yes go execute command O
M0196	01 96	0ed	0128	:If yes go execute command O
M0197	01 97	0ed	0129	:If yes go execute command O
M0198	01 98	0ed	0130	:If yes go execute command O
M0199	01 99	0ed	0131	:If yes go execute command O
M0200	01 100	0ed	0132	:If yes go execute command O

GYRAC.A PROGRAM LISTING (continued)

```

B070 10 B1 17      jsr  avel1b  ;Read and T=1 byte of ang vel
B071 7E B1 17      jrb                ;Get next command

B072 10 B1 FA cndC  jsr  head2b  ;Read and T=2 bytes of heading
B073 7E B1 FA      jrb                ;Get next command

B074 10 B1 11 cndD  jsr  head1b  ;Read and T=1 byte of heading
B075 7E B1 11      jrb                ;Get next command

B076 10 B1 41 cndE  jsr  vel1b   ;Read and T=2 bytes of velocity
B077 7E B1 41      jrb                ;Get next command

B078 10 B1 74 cndF  jsr  vel1b   ;Read and T=1 byte of velocity
B079 7E B1 74      jrb                ;Get next command

B080 10 B1 10 cndG  jsr  d1vel1  ;Clear velocity constant
B081 7E B1 10      jrb                ;Get next command

B082 10 B1 34 cndH  jsr  ave12b  ;Read and T=2 bytes of ang vel
B083 7E B1 34      jrb                ;Get next command

B084 10 B1 89 cndI  jsr  ave11b  ;Read and T=1 byte of ang vel
B085 7E B1 89      jrb                ;Get next command

B086 10 B1 FA cndJ  jsr  head2b  ;Read and T=2 bytes of heading
B087 10 B1 41      jsr  vel1b   ;Read and T=2 bytes of velocity
B088 7E B1 41      jrb                ;Get next command

B089 10 B1 19 cndK  jsr  head1b  ;Read and T=1 byte of heading
B090 10 B1 74      jsr  vel1b   ;Read and T=1 byte of velocity
B091 7E B1 19      jrb                ;Get next command

B092 7E B1 08 cndL  op    init    ;Reset and re-initialize GYRAC

B093 7E B1 82 cndM  op    boot    ;Load Raw heading from 80-202 & velocity

B094 10 B1 91 cndN  jsr  test    ;T=printable char to 80-202
B095 7E B1 91      jrb                ;Get next command

B096 10 B1 10 cndO  jsr  head1b  ;Read and T=1 byte of heading
B097 10 B1 41      jsr  vel1b   ;Read and T=2 bytes of velocity
B098 7E B1 10      jrb                ;Get next command

```

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 5/10/66
VERSION: 1.0
NAME: getcom
MODULE NUMBER: 110
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.
RAISED VARIABLES: None
RETURN: 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 W. RAME JR.
HISTORY: NONE
    
```

```

E000 01 A0 00 getcom ldaa aciastat ;Get ACIA (RS-232) status
E001 04 01 anda #ndrf ;Is receive data register full
E004 07 00 beq getcom ;If no loop to getcom
E005 05 A0 01 ldaa aciadata ;Else read command byte from ACIA
E009 09 00 rts
    
```

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 05 25/80
VERSION: 1.1
NAME: head2b
MODULE NUMBER: 204
DESCRIPTION: This module sends a write pulse to the Gyrac
             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.
PULSED 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:
             wpulse
CALLING MODULES: main
AUTHOR: CAPT WILLIAM J. RAMEY JR.
HISTORY: Version 1.0 original
             Version 1.1 change heading data format
    
```

```

E0FA 06 0F      head2b  ldaa #headadd1 ;Get address of LSB of heading
E0FB 07 00 00      staa piadataA ;Write LSB heading address to GYRAC
E0FC 00 00 00      .sr      wpulse ;Send GYRAC write pulse to latch data
E0FD 08 0E      ldaa #headaddh ;Get address of MSB of heading
E104 07 00 00      staa piadataA ;Write MSB heading address to GYRAC
E107 08 00 00      ldsb piadataB ;Read MSB of heading from GYRAC
E10A 00 00 00      .sr      tr ;Send MSB of heading to ACIA
E10D 06 0F      ldaa #headadd1 ;Get address of LSB of heading
E10F 07 00 00      staa piadataA ;Write LSB heading address to GYRAC
E111 08 00 00      ldsb piadataB ;Read LSB of heading from GYRAC
E115 00 00 00      .sr      tr ;Send LSB of heading to ACIA
E116 09      .sta
    
```

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 05/14/85
VERSION: 1.00
NAME: 00
MODULE NUMBER: 001
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.
PARAMETER VARIABLES: Output data byte in the B register
RETURN: Data byte output to ACIA
LOCAL VARIABLES USED: None
GLOBAL VARIABLES CHANGED: None
REGISTERS USED: A, B
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: None
CALLING MODULES: readit
                  readb
                  v112b
                  v111b
                  v110b
                  v111c
                  v111d
AUTHOR: CAPT WILLIAM D. FAMEY JR.
HISTORY: NONE
    
```

```

E010 06 00 00 00      11aa  acrststat ;Read ACIA (RS-232) status
E011 04 02           anda  #00000000 ;Is ACIA transmit data register empty?
E012 07 09           bcc   #0         ;If no, loop to 06 and recheck status
E013 07 00 01       stab  acldata    ;Else transmit data in B reg to ACIA
E014 01
    
```

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 10/10/68
VERSION: 1.1
NAME: wpulse
MODULE NUMBER: 01
DESCRIPTION: This module sends a write pulse to the GYRAC.
PULSED VARIABLES: None
RETURN: None
LOCAL VARIABLES USED: None
LOCAL VARIABLES CHANGED: None
REGISTERS USED: 0
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: None
CALLING MODULES: head2b
                  headb
                  ve12b
                  ve11b
                  ave12b
                  ave11b
AUTHOR: CAPT WILLIAM H. RAMEY JR.
HISTORY: NONE
    
```

```

E023 06 14 wpulse 1dab #write :Get GYRAC write command
E025 F7 30 01 stab piactr1A :Send GYRAC write level to latch data
E028 06 10 1dab #read :Get GYRAC read command
E034 F7 30 01 stab piactr1A :Send GYRAC read level to form a pulse
E035 09
    
```

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 05/24/85
VERSION: 1.1
NAME: headb
MODULE NUMBER: 0.0
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.
RAISED VARIABLES: None
RETURNS: One byte of heading data written to ACIA (RS-232)
GLOBAL VARIABLES USED: temp
LOCAL VARIABLES CHANGED: temp
REGISTERS USED: A, B
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: mpulse
CALLING MODULES: main
AUTHOR: CAPT WILLIAM J. RAMEY JR.
HISTORY: Version 1.0 original
        Version 1.1 change heading data format
    
```

```

E119 56 0F headb ldaa #headaddl ;Get address of LEB of heading
E118 57 50 00 staa piadataA ;Write address of LEB of head to GYRAC
E11E 5D E2 03 jsr mpulse ;Send GYRAC write pulse to latch data
E121 56 4E ldaa #headaddh ;Get address of MEB of heading
E123 57 50 00 staa piadataA ;Write address of MEB of head to GYRAC
E126 56 50 02 ldaa piadataB ;Read MEB of heading data
E129 59 aslb ;Move lower 4 bits to upper 4 bits
E12A 59 aslb ;and shift 4 bytes into the lower
E12B 59 aslb ;4 bits--i.e., 00000000. This will
E12C 59 aslb ;form the MS nibble of the heading byte
E12E 56 4F ldaa #headaddl ;Get address of LEB of heading
E12F 57 50 02 staa piadataA ;Write address of LEB of head to GYRAC
E132 56 50 02 ldaa piadataB ;Read LEB of heading data
E135 44 lraa ;Move upper 4 bits to lower 4 bits
E136 44 lraa ;and shift 4 bytes into the upper
E137 44 lraa ;4 bits--i.e., 00000000. This will
E138 44 lraa ;form the LB nibble of the heading byte
E139 57 00 staa temp ;Save the LB nibble of the heading
E13A 5A 00 clab temp ;Form 8 bits of heading (1 byte)
E13D 5D E2 13 jsr t ;Output 1 byte of heading to ACIA
E140 39 rts
    
```

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 04/11/77
VERSION: 1.1
NAME: velcb
MODULE NUMBER: 110
DESCRIPTION: This module sends a write pulse to the Gyrac
velocity data register to latch the data, reads
the bytes of velocity data, performs and sends
the data, MSB first, to the ACIA (PS-200) output.
Only the least significant 16 bits of the data is
significant.
PARAMETER VARIABLES: None
RETURNS: Two bytes of velocity data written to ACIA (PS-200)
GLOBAL VARIABLES USED: None
GLOBAL VARIABLES CHANGED: None
REGISTERS USED: A, B
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: tr
                pulse
                delay
CALLING MODULES: main
AUTHOR: CAPT WILLIAM J. RAMEY JR.
HISTORY: Version 1.0 original
         Version 1.1 change velocity data format
    
```

```

E141 88 0A   velcb   lcaa  #veladdh ;Get address of MSB of velocity
E142 B7 00 00      staa  @dataA ;Write MSB velocity address to GYRAC
E143 ED ED 01      lsr   pulse   ;Send GYRAC write pulse to latch data
E144 D0 ED 0E      lsr   delay   ;Allow for A to D conversion time
E145 F1 00 02      ldat  @dataA ;Read MSB of heading from GYRAC
E14F D7 00      stab  temp   ;Save raw MSB of data
E151 94          lsrb                ;Form correct MSB
E152 94          lsrb                ;
E153 94          lsrb                ;
E154 94          lsrb                ;
E155 94          lsrb                ;
E156 94          lsrb                ;
E157 D0 E2 10      lsr   tr           ;Send MSB of heading to ACIA
E158 88 10      lcaa  #veladdl ;Get address of LSB of velocity
E159 D7 00 00      staa  @dataA ;Write LSB velocity address to GYRAC
E15A F1 00 02      ldat  @dataA ;Read LSB of velocity from GYRAC
E15B 94          lsrb                ;Adjust raw LSB to correct format
E15C 94          lsrb                ;
E15D 94          lsrb                ;
E15E 94          lsrb                ;
E15F 94          lsrb                ;
E160 94          lsrb                ;
E161 94          lsrb                ;
E162 94          lsrb                ;
E163 94          lsrb                ;
E164 94          lsrb                ;
E165 94          lsrb                ;
E166 94          lsrb                ;
E167 94          lsrb                ;
E168 94          lsrb                ;
E169 94          lsrb                ;
E16A 94          lsrb                ;
E16B 94          lsrb                ;
E16C 94          lsrb                ;
E16D 94          lsrb                ;
E16E 94          lsrb                ;
E16F 94          lsrb                ;
E170 94          lsrb                ;
E171 94          lsrb                ;
    
```

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 11/20/75
VERSION: 1.1
NAME: delay
MODULE NUMBER: 40
DESCRIPTION: This module provides a time delay * 1.117417 *
              (1/1000) in seconds.
PAGES INVOLVED: 0001
REFS: None
LOCAL VARIABLES USED: None
GLOBAL VARIABLES USED: None
REGISTERS USED: A
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: None
CALLING MODULES: ve10b
                  ve10c
                  aie10b
                  sve10c
                  ilve1
AUTHOR: CAPT WILLIAM J. RAMEY JR.
HISTORY: NONE
    
```

```

000E 36 05      delay 10aa #del      ;Get time delay variable
0010 71                next      010
0012 44                decd      010
0014 26 FC                bne      next      ;if not done, do another loop delay
0016 07                rts

0018                org      vector
0019 8W 00 00 00 00      fdb      begin, begin, begin, begin ;1st 40 vector table
001C 8W 00 00 00 00

0400                END
    
```


GYRAC.A PROGRAM LISTING (continued)

```

DATE: 04/24/87
VERSION: 1.1
NAME: velib
MODULE NUMBER: 5.7
DESCRIPTION: This module sends a write pulse to the Gyrac
              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 (latched to ACIA (RS-232))
GLOBAL VARIABLES USED: None
GLOBAL VARIABLES CHANGED: None
REGISTERS USED: A, B
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: 0x
                 0pulse
                 delay
CALLING MODULES: main
AUTHOR: CAPT WILLIAM J. RAMEY JR.
HISTORY: Version 1.0 original
         Version 1.1 change velocity data format
    
```

```

B174 06 04 velib ldaa #veladdr ;Get address of MSB of velocity
B175 17 01 00 staa piadataA ;Write address of MSB of vel to GYRAC
B176 00 02 01 jsr 0pulse ;Send GYRAC write pulse to latch data
B177 00 02 02 jsr delay ;Allow for A to D conversion time
B178 00 00 01 load piadataB ;Read MSB of velocity data
B179 00 02 03 jsr 0x ;Output 1 byte of velocity to ACIA
B180 01
    
```

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 11/21/85
VERSION: 1.1
NAME: ave101
MODULE NUMBER: 101
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, reformat 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:
    pulse
    delay
CALLING MODULES: main
AUTHOR: CAPT WILLIAM J. RAMEY JR.
HISTORY: Version 1.0 original
        version 1.1 change angular velocity data format
    
```

```

E111 06 00 ave101: lda #ave1addh ;Get address of MSB of ang vel
E112 07 10 00 staa #ave1addh ;Write MSB ang vel address to GYRAC
E113 08 02 00 jr pulse ;Send GYRAC write pulse to latch data
E114 09 00 00 jr delay ;Allow for A to D conversion time
E115 0A 10 01 stat #ave1addl ;Read MSB of ang vel from GYRAC
E116 0B 00 staa temp ;Save raw MSB of data
E117 0C 04 lart ;Fix incorrect MSB of data
E118 0D 04 lart
E119 0E 04 lart
E120 0F 04 lart
E121 10 04 lart
E122 11 04 lart
E123 12 04 lart
E124 13 04 lart
E125 14 04 lart
E126 15 04 lart
E127 16 04 lart
E128 17 04 lart
E129 18 04 lart
E130 19 04 lart
E131 1A 04 lart
E132 1B 04 lart
E133 1C 04 lart
E134 1D 04 lart
E135 1E 04 lart
E136 1F 04 lart
E137 20 04 lart
E138 21 04 lart
E139 22 04 lart
E140 23 04 lart
E141 24 04 lart
E142 25 04 lart
E143 26 04 lart
E144 27 04 lart
E145 28 04 lart
E146 29 04 lart
E147 2A 04 lart
E148 2B 04 lart
E149 2C 04 lart
E150 2D 04 lart
E151 2E 04 lart
E152 2F 04 lart
E153 30 04 lart
E154 31 04 lart
E155 32 04 lart
E156 33 04 lart
E157 34 04 lart
E158 35 04 lart
E159 36 04 lart
E160 37 04 lart
E161 38 04 lart
E162 39 04 lart
E163 3A 04 lart
E164 3B 04 lart
E165 3C 04 lart
E166 3D 04 lart
E167 3E 04 lart
E168 3F 04 lart
E169 40 04 lart
E170 41 04 lart
E171 42 04 lart
E172 43 04 lart
E173 44 04 lart
E174 45 04 lart
E175 46 04 lart
E176 47 04 lart
E177 48 04 lart
E178 49 04 lart
E179 4A 04 lart
E180 4B 04 lart
E181 4C 04 lart
E182 4D 04 lart
E183 4E 04 lart
E184 4F 04 lart
E185 50 04 lart
E186 51 04 lart
E187 52 04 lart
E188 53 04 lart
E189 54 04 lart
E190 55 04 lart
E191 56 04 lart
E192 57 04 lart
E193 58 04 lart
E194 59 04 lart
E195 5A 04 lart
E196 5B 04 lart
E197 5C 04 lart
E198 5D 04 lart
E199 5E 04 lart
E200 5F 04 lart
E201 60 04 lart
E202 61 04 lart
E203 62 04 lart
E204 63 04 lart
E205 64 04 lart
E206 65 04 lart
E207 66 04 lart
E208 67 04 lart
E209 68 04 lart
E210 69 04 lart
E211 6A 04 lart
E212 6B 04 lart
E213 6C 04 lart
E214 6D 04 lart
E215 6E 04 lart
E216 6F 04 lart
E217 70 04 lart
E218 71 04 lart
E219 72 04 lart
E220 73 04 lart
E221 74 04 lart
E222 75 04 lart
E223 76 04 lart
E224 77 04 lart
E225 78 04 lart
E226 79 04 lart
E227 7A 04 lart
E228 7B 04 lart
E229 7C 04 lart
E230 7D 04 lart
E231 7E 04 lart
E232 7F 04 lart
E233 80 04 lart
E234 81 04 lart
E235 82 04 lart
E236 83 04 lart
E237 84 04 lart
E238 85 04 lart
E239 86 04 lart
E240 87 04 lart
E241 88 04 lart
E242 89 04 lart
E243 8A 04 lart
E244 8B 04 lart
E245 8C 04 lart
E246 8D 04 lart
E247 8E 04 lart
E248 8F 04 lart
E249 90 04 lart
E250 91 04 lart
E251 92 04 lart
E252 93 04 lart
E253 94 04 lart
E254 95 04 lart
E255 96 04 lart
E256 97 04 lart
E257 98 04 lart
E258 99 04 lart
E259 9A 04 lart
E260 9B 04 lart
E261 9C 04 lart
E262 9D 04 lart
E263 9E 04 lart
E264 9F 04 lart
E265 A0 04 lart
E266 A1 04 lart
E267 A2 04 lart
E268 A3 04 lart
E269 A4 04 lart
E270 A5 04 lart
E271 A6 04 lart
E272 A7 04 lart
E273 A8 04 lart
E274 A9 04 lart
E275 AA 04 lart
E276 AB 04 lart
E277 AC 04 lart
E278 AD 04 lart
E279 AE 04 lart
E280 AF 04 lart
E281 B0 04 lart
E282 B1 04 lart
E283 B2 04 lart
E284 B3 04 lart
E285 B4 04 lart
E286 B5 04 lart
E287 B6 04 lart
E288 B7 04 lart
E289 B8 04 lart
E290 B9 04 lart
E291 BA 04 lart
E292 BB 04 lart
E293 BC 04 lart
E294 BD 04 lart
E295 BE 04 lart
E296 BF 04 lart
E297 C0 04 lart
E298 C1 04 lart
E299 C2 04 lart
E300 C3 04 lart
E301 C4 04 lart
E302 C5 04 lart
E303 C6 04 lart
E304 C7 04 lart
E305 C8 04 lart
E306 C9 04 lart
E307 CA 04 lart
E308 CB 04 lart
E309 CC 04 lart
E310 CD 04 lart
E311 CE 04 lart
E312 CF 04 lart
E313 D0 04 lart
E314 D1 04 lart
E315 D2 04 lart
E316 D3 04 lart
E317 D4 04 lart
E318 D5 04 lart
E319 D6 04 lart
E320 D7 04 lart
E321 D8 04 lart
E322 D9 04 lart
E323 DA 04 lart
E324 DB 04 lart
E325 DC 04 lart
E326 DD 04 lart
E327 DE 04 lart
E328 DF 04 lart
E329 E0 04 lart
E330 E1 04 lart
E331 E2 04 lart
E332 E3 04 lart
E333 E4 04 lart
E334 E5 04 lart
E335 E6 04 lart
E336 E7 04 lart
E337 E8 04 lart
E338 E9 04 lart
E339 EA 04 lart
E340 EB 04 lart
E341 EC 04 lart
E342 ED 04 lart
E343 EE 04 lart
E344 EF 04 lart
E345 F0 04 lart
E346 F1 04 lart
E347 F2 04 lart
E348 F3 04 lart
E349 F4 04 lart
E350 F5 04 lart
E351 F6 04 lart
E352 F7 04 lart
E353 F8 04 lart
E354 F9 04 lart
E355 FA 04 lart
E356 FB 04 lart
E357 FC 04 lart
E358 FD 04 lart
E359 FE 04 lart
E360 FF 04 lart
    
```

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 11/17/88
VERSION: 1.1
AUTHOR: ave11b
MODULE NUMBER: 7.3
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.
PAIRED 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:
    wpulse
    delay
CALLING MODULES: main
AUTHOR: CAPT WILLIAM J. RAMEY US.
HISTORY: Version 1.0 original
         Version 1.1 change angular velocity data format
    
```

```

B107 36 3C ave11b 1aaa #ave1addr :Get address of MSB of ang vel
B108 37 3D 00 staa #stacdataA :write address of MSB of ave1 to 00FFH
B109 3D 3D 20 .sr #wpulse :Send GYRAC write pulse to latch data
B10A 3D 3D 20 .sr #delay :Allow for A to B conversion time
B10B 37 3D 20 .stb #stacdataB :Read MSB of angular velocity data
B10C 3D 3D 10 .jsr #tr :Output 1 byte of ang vel to ACIA
B10D 3F
    
```

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 04/26/85
VERSION: 1.1
NAME: clear
MODULE NUMBER: 8.0
DESCRIPTION: This module clears, sets # on the integration
constant in the acceleration integrator. This is
equivalent to zero velocity, initial condition.
PLACING THE ADDRESS in 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. FAMEY JR.
HISTORY: Version 1.0 original
        Version 1.1 add delay subroutine
    
```

```

M111 26 09   clear  10aa #vconsadd :Get address of vel const clear device
M112 27 09 08   staa  piadataA :Start clearing velocity constant
M113 28 14     11ab #write  :Get write command
M114 29 15 01   stab  piactr1A :Send GYRAC write level to start clear
M115 30 22 1E   jsr   delay  :Allow time for capacitor to discharge
M116 31 10     11ab #read   :Get read command
M117 32 09 01   stab  piactr1A :Send GYRAC read level to stop clear
M118 33     11aa #vcons  :Clear velocity constant
M119 34 15 08   staa  piadataA :Stop clearing velocity constant
M120 35
    
```

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...

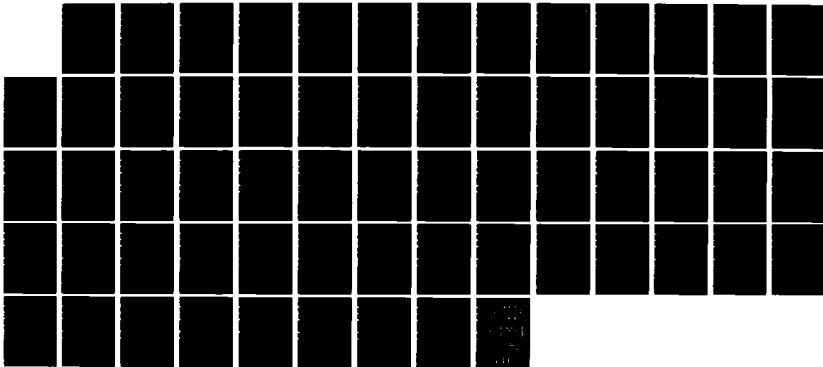
3/3

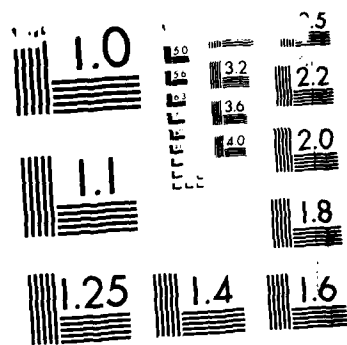
UNCLASSIFIED

R J BLOOM ET AL 02 DEC 85

F/G 17/7

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 05/10/85
VERSION: 1.0
NAME: boot
MODULE NUMBER: 0.0
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, X
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: getcom
CALLING MODULES: main
AUTHOR: CAPT WILLIAM J. RAMEY JR.
HISTORY: NONE
    
```

```

E1E2 50 E0 EF boot   jsr  getcom   ;Get the number of blocks to load
E1E3 17 01          staa  blocks   ;Save # of blocks in memory
E1E7 60 E0 EF      jsr  getcom   ;Get the number of bytes in a block
E1E8 07 00          staa  bytes   ;Save # of bytes in a block in memory
E1E9 FE 60 00      ldw  rambot   ;Load pointer to RAM memory
E1EF 05 02          ldab  bytes   ;Get number of bytes
E1F1 60 E0 EF bytloop jsr  getcom   ;Get byte of data
E1F4 A7 00          staa  @,x     ;Load data byte into RAM
E1F6 0E           incw           ;Increment to next RAM pointer
E1F7 5A           decb           ;Decrement block byte count
E1F8 01 F7        bne  bytloop   ;If byte count not 0, get next byte
E1FA 74 00 01     dec  blocks   ;Else block done, decrement blk count
E1FB 26 F0        bne  blkloop   ;If all block are not done, go another
E1FD 7E 00 00     jno  rambot   ;Else go execute loaded program
    
```

GYRAC.A PROGRAM LISTING (continued)

```

DATE: 05/10/65
VERSION: 1.0
NAME: test
MODULE NUMBER: 1019
DESCRIPTION: This module will, on demand send to the ACIA
              (93-222) output 70 printable ASCII characters.
              It will then return control to the calling
              program.
PASSED VARIABLES: None
RETURNS: 70 characters to ACIA
GLOBAL VARIABLES USED: None
GLOBAL VARIABLES CHANGED: None
REGISTERS USED: A, B
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: cx
CALLING MODULES: main
AUTHOR: CAPT WILLIAM J. RAMEY JR.
HISTORY: NONE
    
```

```

E002 16 21 test ldao #21H ;Get first ASCII character
E004 07 03 stab char ;Save character in memory
E006 16 0D ldaa #03 ;Get number of characters to be sent
E008 07 04 staa charcnt ;Save character count
E00A 01 03 charloop ljab char ;Get next ASCII character
E00C 0D 02 13 jsr tx ;Send ASCII character
E00E 0C 0A 03 inc char ;Increment to next ASCII character
E010 0A 03 04 dec charcnt ;Decrement character count
E012 16 03 bne charloop ;If all characters have not been sent
E014 00 00 ;do send the next one
E017 09 rts ;Else return to calling program
    
```


GYRAC.A PROGRAM LISTING (continued)

ac1a	E001	ac1aon1	E015	ac1adata	E011	ac1aasse	E000
ac1astat	E000	ave1ib	E109	ave1ibc	E181	ave1adddh	E000
ave1addd1	E000	begin	E000	bitloop	E1E0	bitcris	E001
bit	E1E0	bytes	E000	butloop	E1F1	char	E000
charone	E004	charloop	E004	cinvel	E000	ck1A	E000
ck1C	E000	ck1C	E000	ck1D	E004	ck1E	E004
ck1F	E000	ck1G	E000	ck1H	E000	ck1I	E000
ck1J	E000	ck1J	E001	ck1L	E004	ck1M	E000
ck1N	E1E0	ck1O	E0E0	ck1A	E041	ck1B	E042
ck1P	E042	ck1Q	E044	ck1E	E045	ck1F	E040
ck1R	E047	ck1H	E043	ck1I	E049	ck1N	E044
ck1S	E047	ck1L	E040	ck1M	E040	ck1O	E04E
ck1O	E04F	delay	E00E	getcom	E0EF	head1b	E119
head1b	E0FA	headadddh	E00E	headaddd1	E00F	init	E000
in1uc	E000	next	E000	pl1a	E004	pl1aon1A	E001
pl1aon1B	E002	pl1adataA	E000	pl1adataB	E002	on	E01A
pt1ac	E004	pt1ad	E000	pt1ont	E002	pt1onh11	E000
pt1onh12	E001	pt1nt	E000	rabot	E000	rd1r	E001
read	E000	stack	E0FF	sd1l	E005	td1r	E001
temp	E000	test	E002	tx	E016	vb1rsadd	E000
vector	E1F0	ve1ib	E174	ve1ibc	E141	ve1adddh	E00A
ve1addd1	E00E	wpulse	E003	write	E004		

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 hexadecimal 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 off 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 G-2
GTEST.A Test Program Overlay G-3
GTEST.A Operating Instructions G-11

PUTD
OVERLAY

PUTC
OVERLAY

PUTB
OVERLAY

GTEST.A TEST PROGRAM OVERLAY

THE Following
Reproduced from
best available copy. 
PAGES

DATE: 7/10/75
VERSION: 1.1
TITLE: G-RAC DATA COLLECTION PROGRAM
FILENAME: GTEST.A
COORDINATOR: CAPT WILLIAM J. RAMEY, Jr.
PROJECT: GYRO AND GYROSCOPE METER BASED NAVIGATION SYSTEM FOR A
MOBILE AUTONOMOUS ROBOT (THESEI)
OPERATING SYSTEM: DEC/1001 VAX/VMS LOGICAL MAGNOLIA MICROSYSTEM 1000
LANGUAGE: ROSS-86 ASSEMBLER V1.1: VIRTUAL DEVICES 1969
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 MOD in terminal (T)
mode with the following command: L,DCFA,05E9 followed by
transmitting the file gtest.hex. All of the original
commands still work: D to begin transmitting data or HIR,
T to pause data Tn C to resume data Tn, and C to
terminate 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
putd
puti

FUNCTION: This program overlays three existing subroutines in
the nav computer's RAM memory to allow the nav
computer to request data from the GVRAC (over an
RS-232 link) and then transmit the GVRAC's data
to an external computer over an RS-232 link. It
replaces the subroutines that send sonar S, G, and
D data with subroutines that send velocity, heading,
and angular velocity respectively. The data is
represented as four hexadecimal digits.

0004	begin	418	019AH	start address of t_start_sonar
0008	beginb	419	019AH	start address of t_start_b
000C	begina	41A	019AH	start address of t_start_a
0010	gvrac	41B	019AH	GVRAC command-- to all 3 bytes
0014	gvrac	41C	019AH	GVRAC command-- to all 3 bytes
0018	gvrac	41D	019AH	GVRAC command-- to heading 2 bytes
001C	gvrac	41E	019AH	GVRAC command-- to heading 1 byte
0020	gvrac	41F	019AH	GVRAC command-- to angular velocity
0024	gvrac	420	019AH	GVRAC command-- to angular velocity
0028	gvrac	421	019AH	GVRAC command-- to heading 2 bytes
002C	gvrac	422	019AH	GVRAC command-- to heading 1 byte
0030	gvrac	423	019AH	GVRAC command-- to heading 2 bytes
0034	gvrac	424	019AH	GVRAC command-- to heading 1 byte
0038	gvrac	425	019AH	GVRAC command-- heading 1 byte
003C	gvrac	426	019AH	GVRAC command-- heading 2 bytes
0040	gvrac	427	019AH	GVRAC command-- heading 2 bytes
0044	gvrac	428	019AH	GVRAC command-- heading 1 byte
0048	gvrac	429	019AH	GVRAC command-- heading 1 byte
004C	gvrac	42A	019AH	GVRAC command-- heading 1 byte
0050	gvrac	42B	019AH	GVRAC command-- heading 1 byte
0054	gvrac	42C	019AH	GVRAC command-- heading 1 byte
0058	gvrac	42D	019AH	GVRAC command-- heading 1 byte
005C	gvrac	42E	019AH	GVRAC command-- heading 1 byte
0060	gvrac	42F	019AH	GVRAC command-- heading 1 byte
0064	gvrac	430	019AH	GVRAC command-- heading 1 byte
0068	gvrac	431	019AH	GVRAC command-- heading 1 byte
006C	gvrac	432	019AH	GVRAC command-- heading 1 byte
0070	gvrac	433	019AH	GVRAC command-- heading 1 byte
0074	gvrac	434	019AH	GVRAC command-- heading 1 byte
0078	gvrac	435	019AH	GVRAC command-- heading 1 byte
007C	gvrac	436	019AH	GVRAC command-- heading 1 byte
0080	gvrac	437	019AH	GVRAC command-- heading 1 byte
0084	gvrac	438	019AH	GVRAC command-- heading 1 byte
0088	gvrac	439	019AH	GVRAC command-- heading 1 byte
008C	gvrac	43A	019AH	GVRAC command-- heading 1 byte
0090	gvrac	43B	019AH	GVRAC command-- heading 1 byte
0094	gvrac	43C	019AH	GVRAC command-- heading 1 byte
0098	gvrac	43D	019AH	GVRAC command-- heading 1 byte
009C	gvrac	43E	019AH	GVRAC command-- heading 1 byte
00A0	gvrac	43F	019AH	GVRAC command-- heading 1 byte
00A4	gvrac	440	019AH	GVRAC command-- heading 1 byte
00A8	gvrac	441	019AH	GVRAC command-- heading 1 byte
00AC	gvrac	442	019AH	GVRAC command-- heading 1 byte
00B0	gvrac	443	019AH	GVRAC command-- heading 1 byte
00B4	gvrac	444	019AH	GVRAC command-- heading 1 byte
00B8	gvrac	445	019AH	GVRAC command-- heading 1 byte
00BC	gvrac	446	019AH	GVRAC command-- heading 1 byte
00C0	gvrac	447	019AH	GVRAC command-- heading 1 byte
00C4	gvrac	448	019AH	GVRAC command-- heading 1 byte
00C8	gvrac	449	019AH	GVRAC command-- heading 1 byte
00CC	gvrac	44A	019AH	GVRAC command-- heading 1 byte
00D0	gvrac	44B	019AH	GVRAC command-- heading 1 byte
00D4	gvrac	44C	019AH	GVRAC command-- heading 1 byte
00D8	gvrac	44D	019AH	GVRAC command-- heading 1 byte
00DC	gvrac	44E	019AH	GVRAC command-- heading 1 byte
00E0	gvrac	44F	019AH	GVRAC command-- heading 1 byte
00E4	gvrac	450	019AH	GVRAC command-- heading 1 byte
00E8	gvrac	451	019AH	GVRAC command-- heading 1 byte
00EC	gvrac	452	019AH	GVRAC command-- heading 1 byte
00F0	gvrac	453	019AH	GVRAC command-- heading 1 byte
00F4	gvrac	454	019AH	GVRAC command-- heading 1 byte
00F8	gvrac	455	019AH	GVRAC command-- heading 1 byte
00FC	gvrac	456	019AH	GVRAC command-- heading 1 byte
0100	gvrac	457	019AH	GVRAC command-- heading 1 byte
0104	gvrac	458	019AH	GVRAC command-- heading 1 byte
0108	gvrac	459	019AH	GVRAC command-- heading 1 byte
010C	gvrac	45A	019AH	GVRAC command-- heading 1 byte
0110	gvrac	45B	019AH	GVRAC command-- heading 1 byte
0114	gvrac	45C	019AH	GVRAC command-- heading 1 byte
0118	gvrac	45D	019AH	GVRAC command-- heading 1 byte
011C	gvrac	45E	019AH	GVRAC command-- heading 1 byte
0120	gvrac	45F	019AH	GVRAC command-- heading 1 byte
0124	gvrac	460	019AH	GVRAC command-- heading 1 byte
0128	gvrac	461	019AH	GVRAC command-- heading 1 byte
012C	gvrac	462	019AH	GVRAC command-- heading 1 byte
0130	gvrac	463	019AH	GVRAC command-- heading 1 byte
0134	gvrac	464	019AH	GVRAC command-- heading 1 byte
0138	gvrac	465	019AH	GVRAC command-- heading 1 byte
013C	gvrac	466	019AH	GVRAC command-- heading 1 byte
0140	gvrac	467	019AH	GVRAC command-- heading 1 byte
0144	gvrac	468	019AH	GVRAC command-- heading 1 byte
0148	gvrac	469	019AH	GVRAC command-- heading 1 byte
014C	gvrac	46A	019AH	GVRAC command-- heading 1 byte
0150	gvrac	46B	019AH	GVRAC command-- heading 1 byte
0154	gvrac	46C	019AH	GVRAC command-- heading 1 byte
0158	gvrac	46D	019AH	GVRAC command-- heading 1 byte
015C	gvrac	46E	019AH	GVRAC command-- heading 1 byte
0160	gvrac	46F	019AH	GVRAC command-- heading 1 byte
0164	gvrac	470	019AH	GVRAC command-- heading 1 byte
0168	gvrac	471	019AH	GVRAC command-- heading 1 byte
016C	gvrac	472	019AH	GVRAC command-- heading 1 byte
0170	gvrac	473	019AH	GVRAC command-- heading 1 byte
0174	gvrac	474	019AH	GVRAC command-- heading 1 byte
0178	gvrac	475	019AH	GVRAC command-- heading 1 byte
017C	gvrac	476	019AH	GVRAC command-- heading 1 byte
0180	gvrac	477	019AH	GVRAC command-- heading 1 byte
0184	gvrac	478	019AH	GVRAC command-- heading 1 byte
0188	gvrac	479	019AH	GVRAC command-- heading 1 byte
018C	gvrac	47A	019AH	GVRAC command-- heading 1 byte
0190	gvrac	47B	019AH	GVRAC command-- heading 1 byte
0194	gvrac	47C	019AH	GVRAC command-- heading 1 byte
0198	gvrac	47D	019AH	GVRAC command-- heading 1 byte
019C	gvrac	47E	019AH	GVRAC command-- heading 1 byte
01A0	gvrac	47F	019AH	GVRAC command-- heading 1 byte
01A4	gvrac	480	019AH	GVRAC command-- heading 1 byte
01A8	gvrac	481	019AH	GVRAC command-- heading 1 byte
01AC	gvrac	482	019AH	GVRAC command-- heading 1 byte
01B0	gvrac	483	019AH	GVRAC command-- heading 1 byte
01B4	gvrac	484	019AH	GVRAC command-- heading 1 byte
01B8	gvrac	485	019AH	GVRAC command-- heading 1 byte
01BC	gvrac	486	019AH	GVRAC command-- heading 1 byte
01C0	gvrac	487	019AH	GVRAC command-- heading 1 byte
01C4	gvrac	488	019AH	GVRAC command-- heading 1 byte
01C8	gvrac	489	019AH	GVRAC command-- heading 1 byte
01CC	gvrac	48A	019AH	GVRAC command-- heading 1 byte
01D0	gvrac	48B	019AH	GVRAC command-- heading 1 byte
01D4	gvrac	48C	019AH	GVRAC command-- heading 1 byte
01D8	gvrac	48D	019AH	GVRAC command-- heading 1 byte
01DC	gvrac	48E	019AH	GVRAC command-- heading 1 byte
01E0	gvrac	48F	019AH	GVRAC command-- heading 1 byte
01E4	gvrac	490	019AH	GVRAC command-- heading 1 byte
01E8	gvrac	491	019AH	GVRAC command-- heading 1 byte
01EC	gvrac	492	019AH	GVRAC command-- heading 1 byte
01F0	gvrac	493	019AH	GVRAC command-- heading 1 byte
01F4	gvrac	494	019AH	GVRAC command-- heading 1 byte
01F8	gvrac	495	019AH	GVRAC command-- heading 1 byte
01FC	gvrac	496	019AH	GVRAC command-- heading 1 byte
0200	gvrac	497	019AH	GVRAC command-- heading 1 byte
0204	gvrac	498	019AH	GVRAC command-- heading 1 byte
0208	gvrac	499	019AH	GVRAC command-- heading 1 byte
020C	gvrac	49A	019AH	GVRAC command-- heading 1 byte
0210	gvrac	49B	019AH	GVRAC command-- heading 1 byte
0214	gvrac	49C	019AH	GVRAC command-- heading 1 byte
0218	gvrac	49D	019AH	GVRAC command-- heading 1 byte
021C	gvrac	49E	019AH	GVRAC command-- heading 1 byte
0220	gvrac	49F	019AH	GVRAC command-- heading 1 byte
0224	gvrac	4A0	019AH	GVRAC command-- heading 1 byte
0228	gvrac	4A1	019AH	GVRAC command-- heading 1 byte
022C	gvrac	4A2	019AH	GVRAC command-- heading 1 byte
0230	gvrac	4A3	019AH	GVRAC command-- heading 1 byte
0234	gvrac	4A4	019AH	GVRAC command-- heading 1 byte
0238	gvrac	4A5	019AH	GVRAC command-- heading 1 byte
023C	gvrac	4A6	019AH	GVRAC command-- heading 1 byte
0240	gvrac	4A7	019AH	GVRAC command-- heading 1 byte
0244	gvrac	4A8	019AH	GVRAC command-- heading 1 byte
0248	gvrac	4A9	019AH	GVRAC command-- heading 1 byte
024C	gvrac	4AA	019AH	GVRAC command-- heading 1 byte
0250	gvrac	4AB	019AH	GVRAC command-- heading 1 byte
0254	gvrac	4AC	019AH	GVRAC command-- heading 1 byte
0258	gvrac	4AD	019AH	GVRAC command-- heading 1 byte
025C	gvrac	4AE	019AH	GVRAC command-- heading 1 byte
0260	gvrac	4AF	019AH	GVRAC command-- heading 1 byte
0264	gvrac	4B0	019AH	GVRAC command-- heading 1 byte
0268	gvrac	4B1	019AH	GVRAC command-- heading 1 byte
026C	gvrac	4B2	019AH	GVRAC command-- heading 1 byte
0270	gvrac	4B3	019AH	GVRAC command-- heading 1 byte
0274	gvrac	4B4	019AH	GVRAC command-- heading 1 byte
0278	gvrac	4B5	019AH	GVRAC command-- heading 1 byte
027C	gvrac	4B6	019AH	GVRAC command-- heading 1 byte
0280	gvrac	4B7	019AH	GVRAC command-- heading 1 byte
0284	gvrac	4B8	019AH	GVRAC command-- heading 1 byte
0288	gvrac	4B9	019AH	GVRAC command-- heading 1 byte
028C	gvrac	4BA	019AH	GVRAC command-- heading 1 byte
0290	gvrac	4BB	019AH	GVRAC command-- heading 1 byte
0294	gvrac	4BC	019AH	GVRAC command-- heading 1 byte
0298	gvrac	4BD	019AH	GVRAC command-- heading 1 byte
029C	gvrac	4BE	019AH	GVRAC command-- heading 1 byte
02A0	gvrac	4BF	019AH	GVRAC command-- heading 1 byte
02A4	gvrac	4C0	019AH	GVRAC command-- heading 1 byte
02A8	gvrac	4C1	019AH	GVRAC command-- heading 1 byte
02AC	gvrac	4C2	019AH	GVRAC command-- heading 1 byte
02B0	gvrac	4C3	019AH	GVRAC command-- heading 1 byte
02B4	gvrac	4C4	019AH	GVRAC command-- heading 1 byte
02B8	gvrac	4C5	019AH	GVRAC command-- heading 1 byte
02BC	gvrac	4C6	019AH	GVRAC command-- heading 1 byte
02C0	gvrac	4C7	019AH	GVRAC command-- heading 1 byte
02C4	gvrac	4C8	019AH	GVRAC command-- heading 1 byte
02C8	gvrac	4C9	019AH	GVRAC command-- heading 1 byte
02CC	gvrac	4CA	019AH	GVRAC command-- heading 1 byte
02D0	gvrac	4CB	019AH	GVRAC command-- heading 1 byte
02D4	gvrac	4CC	019AH	GVRAC command-- heading 1 byte
02D8	gvrac	4CD	019AH	GVRAC command-- heading 1 byte
02DC	gvrac	4CE	019AH	GVRAC command-- heading 1 byte
02E0	gvrac	4CF	019AH	GVRAC command-- heading 1 byte
02E4	gvrac	4D0	019AH	GVRAC command-- heading 1 byte
02E8	gvrac	4D1	019AH	GVRAC command-- heading 1 byte
02EC	gvrac	4D2	019AH	GVRAC command-- heading 1 byte
02F0	gvrac	4D3	019AH	GVRAC command-- heading 1 byte
02F4	gvrac	4D4	019AH	GVRAC command-- heading 1 byte
02F8	gvrac	4D5	019AH	GVRAC command-- heading 1 byte
02FC	gvrac	4D6	019AH	GVRAC command-- heading 1 byte
0300	gvrac	4D7	019AH	GVRAC command-- heading 1 byte
0304	gvrac	4D8	019AH	GVRAC command-- heading 1 byte
0308	gvrac	4D9	019AH	GVRAC command-- heading 1 byte
030C	gvrac	4DA	019AH	GVRAC command-- heading 1 byte
0310	gvrac	4DB	019AH	GVRAC command-- heading 1 byte
0314	gvrac	4DC	019AH	GVRAC command-- heading 1 byte
0318	gvrac	4DD	019AH	GVRAC command-- heading 1 byte
031C	gvrac	4DE	019AH	GVRAC command-- heading 1 byte
0320	gvrac	4DF	019AH	GVRAC command-- heading 1 byte
0324	gvrac	4E0	019AH	GVRAC command-- heading 1 byte
0328	gvrac	4E1	019AH	GVRAC command-- heading 1 byte
032C	gvrac	4E2	019AH	GVRAC command-- heading 1 byte
0330	gvrac	4E3	019AH	GVRAC command-- heading 1 byte
0334	gvrac	4E4	019AH	GVRAC command-- heading 1 byte
0338	gvrac	4E5	019AH	GVRAC command-- heading 1 byte
033C	gvrac	4E6	019AH	GVRAC command-- heading 1 byte
0340	gvrac	4E7	019AH	GVRAC command-- heading 1 byte
0344	gvrac	4E8	019AH	GVRAC command-- heading 1 byte
0348	gvrac	4E9	019AH	GVRAC command-- heading 1 byte
034C	gvrac	4EA	019AH	GVRAC command-- heading 1 byte
0350	gvrac	4EB	019AH	GVRAC command-- heading 1 byte
0354	gvrac	4EC	019AH	GVRAC command-- heading 1 byte
0358	gvrac	4ED	019AH	GVRAC command-- heading 1 byte
035C	gvrac	4EE	019AH	GVRAC command-- heading 1 byte
0360	gvrac	4EF	019AH	GVRAC command-- heading 1 byte
0364	gvrac	4F0	019AH	GVRAC command-- heading 1 byte
0368	gvrac	4F1	019AH	GVRAC command-- heading 1 byte
036C	gvrac	4F2	019AH	GVRAC command-- heading 1 byte
0370	gvrac	4F3	019AH	GVRAC command-- heading 1 byte
0374	gvrac	4F4	019AH	GVRAC command-- heading 1 byte
0378	gvrac	4F5	019AH	GVRAC command-- heading 1 byte
037C	gvrac	4F6	019AH	GVRAC command-- heading 1 byte
0380	gvrac	4F7	019AH	GVRAC command-- heading 1 byte
0384	gvrac	4F8	019AH	GVRAC command-- heading 1 byte
0388	gvrac	4F9	019AH	GVRAC command-- heading 1 byte
038C	gvrac	4FA	019AH	GVRAC command-- heading 1 byte
0390	gvrac	4FB	019AH	GVRAC command-- heading 1 byte
0394	gvrac	4FC	019AH	GVRAC command-- heading 1 byte
0398	gvrac	4FD	019AH	GVRAC command-- heading 1 byte
039C	gvrac	4FE		

GTEST.A TEST PROGRAM OVERLAY (continued)

```

DATE: 07/19/85
VERSION: 1.0
NAME: gtest
MODULE NUMBER: 110
DESCRIPTION: This module reads 16 bytes of velocity data from
the GPRAC, converts it to four hexadecimal
characters and places the characters in the output
buffer.
PARSED VARIABLES: NONE
RETURN: Four hex velocity characters in the data buffer.
GLOBAL VARIABLES USED: None
GLOBAL VARIABLES CHANGED: None
REGISTERS USED: A, D, X
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: putChex
CALLING MODULES: gtest
AUTHOR: CAPT WILLIAM L. RAMEY, US.
HISTORY: NONE
    
```

```

00FA 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00FB 0E 0E 40 subb 000 beginv ;Start of putb subroutine
00FC 06 45 load #0000 ;0. #intreading ;Point to velocity data buffer
00FD 06 00 10 tbb 00a acia1stat ;Get GPRAC 2 byte vel command
00FE 04 02 anda #000 ;Get GPRAC ACIA status
00FF 07 00 beq 000 ;Is transmit data reg empty?
0100 07 00 11 stb acia1data ;If no loop to tbb and recheck status
0101 07 00 11 ;Else send GPRAC command
0102 08 00 10 tbb 00a acia1stat ;Get GPRAC ACIA status
0103 04 01 anda #000 ;Is receive data register full?
0104 07 00 beq 000 ;If no loop to tbb & recheck status
0105 08 00 11 00a acia1data ;Else get the LED of velocity data
0106 08 02 00 jnr putline ;Convert data to hex & save in buffer
0107 08 00 10 tbb 00a acia1stat ;Get GPRAC ACIA status
0108 04 01 anda #000 ;Is receive data register full?
0109 07 00 beq 000 ;If no loop to tbb & recheck status
010A 08 00 11 00a acia1data ;Else get the LED of velocity data
010B 08 02 00 jnr putChex ;Convert data to hex & save in buffer
010C 08 00 10 tbb 000 ;Blank out remainder of old program
010D 07 00 00 jnr 000
010E 07 00 00 jnr 000
010F 07 00 00 jnr 000
0110 07 00 00 jnr 000
0111 07 00 00 jnr 000
0112 07 00 00 jnr 000
0113 07 00 00 jnr 000
0114 07 00 00 jnr 000
0115 07 00 00 jnr 000
0116 07 00 00 jnr 000
0117 07 00 00 jnr 000
0118 07 00 00 jnr 000
0119 07 00 00 jnr 000
011A 07 00 00 jnr 000
011B 07 00 00 jnr 000
011C 07 00 00 jnr 000
011D 07 00 00 jnr 000
011E 07 00 00 jnr 000
011F 07 00 00 jnr 000
0120 07 00 00 jnr 000
0121 07 00 00 jnr 000
0122 07 00 00 jnr 000
0123 07 00 00 jnr 000
0124 07 00 00 jnr 000
0125 07 00 00 jnr 000
0126 07 00 00 jnr 000
0127 07 00 00 jnr 000
0128 07 00 00 jnr 000
0129 07 00 00 jnr 000
012A 07 00 00 jnr 000
012B 07 00 00 jnr 000
012C 07 00 00 jnr 000
012D 07 00 00 jnr 000
012E 07 00 00 jnr 000
012F 07 00 00 jnr 000
0130 07 00 00 jnr 000
    
```

GTEST.A TEST PROGRAM OVERLAY (continued)

0307	01	END
0308	01	END
0309	01	END
0310	01	END
0311	01	END
0312	01	END
0313	01	END
0314	01	END
0315	01	END
0316	01	END
0317	01	END
0318	01	END
0319	01	END
0320	01	END
0321	01	END
0322	01	END
0323	01	END
0324	01	END
0325	01	END
0326	01	END
0327	01	END
0328	01	END
0329	01	END
0330	01	END
0331	01	END
0332	01	END
0333	01	END
0334	01	END
0335	01	END
0336	01	END
0337	01	END
0338	01	END
0339	01	END
0340	01	END
0341	01	END
0342	01	END
0343	01	END
0344	01	END
0345	01	END
0346	01	END
0347	01	END
0348	01	END
0349	01	END
0350	01	END

GTEST.A TEST PROGRAM OVERLAY (continued)

```

DATE: 10/21/85
VERSION: 1.1
NAME: gtest
MODULE NUMBER: 1.
DESCRIPTION: This module reads two bytes of heading data from
the GYRAC, converts it to four hex ASCII
characters and places the characters in the output
buffer. The most significant nibble is masked
out.
RAISED VARIABLES: None
RETURNS: Four hex heading characters in the data buffer.
GLOBAL VARIABLES USED: None
LOCAL VARIABLES CHANGED: None
REGISTERS USED: A, D, X
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: gputchar
CALLING MODULES: gtest
AUTHOR: CAPT WILLIAM J. FANEY JR.
HISTORY: Version 1.0 original program
Version 1.1 add mask to MS nibble of the heading
    
```

```

004A 0E 45 50 auto    ld  #inheading ;Point to heading data buffer
004C 08 43          ldab #com1     ;Get GYRAC 2 byte heading command
004E 05 08 10 bcd    ldaa acialstat ;Get GYRAC ACIA status
0050 34 02          anda #drf     ;Is transmit data register full?
0052 17 F9          beq  bcd     ;If no loop to bcd and recheck status
0054 F7 09 11       stab acialdata ;Else send GYRAC command
0056

0059 34 08 10 auto    ldaa acialstat ;Get GYRAC ACIA status
005B 34 01          anda #drf     ;Is receive data register full?
005D 17 F9          beq  rcd     ;If no loop to rcd & recheck status
005F 31 08 11       ldas acialdata ;Else get the MSB of heading data
0061 34 0F          anda #mask    ;Mask out most significant nibble
0063 10 02 50       lsr  outthe   ;Convert data to hex & save in buffer
0065

0068 35 08 10 auto    ldaa acialstat ;Get GYRAC ACIA status
006A 34 01          anda #drf     ;Is receive data register full?
006C 17 F9          beq  rcd     ;If no loop to rcd & recheck status
006E 31 08 11       ldas acialdata ;Else get the LSB of heading data
0070 3F 02 50       lsr  outthe   ;Convert data to hex & save in buffer
0072 00              ;Blank out remainder of out buffer
0074 01              ;
0076 01              ;
0078 01              ;
007A 01              ;
007C 01              ;
007E 01              ;
0080 01              ;
0082 01              ;
0084 01              ;
0086 01              ;
0088 01              ;
008A 01              ;
008C 01              ;
008E 01              ;
0090 01              ;
0092 01              ;
0094 01              ;
0096 01              ;
0098 01              ;
009A 01              ;
009C 01              ;
009E 01              ;
00A0 01              ;
00A2 01              ;
00A4 01              ;
00A6 01              ;
00A8 01              ;
00AA 01              ;
00AC 01              ;
00AE 01              ;
00B0 01              ;
00B2 01              ;
00B4 01              ;
00B6 01              ;
00B8 01              ;
00BA 01              ;
00BC 01              ;
00BE 01              ;
00C0 01              ;
00C2 01              ;
00C4 01              ;
00C6 01              ;
00C8 01              ;
00CA 01              ;
00CC 01              ;
00CE 01              ;
00D0 01              ;
00D2 01              ;
00D4 01              ;
00D6 01              ;
00D8 01              ;
00DA 01              ;
00DC 01              ;
00DE 01              ;
00E0 01              ;
00E2 01              ;
00E4 01              ;
00E6 01              ;
00E8 01              ;
00EA 01              ;
00EC 01              ;
00EE 01              ;
00F0 01              ;
00F2 01              ;
00F4 01              ;
00F6 01              ;
00F8 01              ;
00FA 01              ;
00FC 01              ;
00FE 01              ;
0100 01              ;
    
```

GTEST.A TEST PROGRAM OVERLAY (continued)

0007	01	REP
0010	01	REP
0007	01	REP
0014	01	REP
0011	01	REP
0010	01	REP
0011	01	REP
0010	01	REP
0017	01	REP
0010	01	REP
0011	01	REP
0010	01	REP
0011	01	REP
0014	01	REP
0015	01	REP
0012	01	REP
0017	01	REP
0010	01	REP
0010	01	REP

GTEST.A TEST PROGRAM OVERLAY (continued)

```

DATE: 07 10 85
VERSION: 1.0
NAME: gtd
MODULE NUMBER: 1.0
DESCRIPTION: This module reads two bytes of ang velocity data
             from the GVRAC, 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, D, X
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: putHex
CALLING MODULES: send
AUTHOR: CAPT WILLIAM J. RAMEY JR.
HISTORY: NONE
    
```

```

03FA 0E 05 5A puts      ldv  #InReading ;Point to ang velocity data buffer
03FB 03 48          ldab #000H      ;Get GVRAC 2 byte ang vel command
03FC 05 00 10 bwd      ldaa acialstat ;Get GVRAC ACIA status
03FD 04 02          andca #0dne   ;Is transmit data reg empty?
03FE 07 F0          beq 0x3     ;If no loop to bwd and recheck status
03FF 07 00 11          stb  acialdata ;Else send GVRAC command

04A9 05 00 10 rwd      ldaa acialstat ;Get GVRAC ACIA status
04AA 04 01          andca #0dnf   ;Is receive data register full?
04AB 07 F0          beq 0x0H    ;If no loop to rwd & recheck status
04AC 05 00 11          ldaa acialdata ;Else get the LBE of ang velocity data
04AD 0D 02 53          lsr  put2hex  ;Convert data to hex & save in buffer

0508 05 00 10 rwd      ldaa acialstat ;Get GVRAC ACIA status
0509 04 01          andca #0dnf   ;Is receive data register full?
050A 07 F0          beq 0x0H    ;If no loop to rwd & recheck status
050B 05 00 11          ldaa acialdata ;Else get the LBE of ang velocity data
050C 0D 02 53          lsr  put2hex  ;Convert data to hex & save in buffer
050D          rts          ;Blank out remainder of old program
050E          rts
050F          rts
0510          rts
0511          rts
0512          rts
0513          rts
0514          rts
0515          rts
0516          rts
0517          rts
0518          rts
0519          rts
051A          rts
051B          rts
051C          rts
051D          rts
051E          rts
051F          rts
0520          rts
0521          rts
0522          rts
0523          rts
0524          rts
0525          rts
0526          rts
0527          rts
0528          rts
0529          rts
052A          rts
052B          rts
052C          rts
052D          rts
052E          rts
052F          rts
0530          rts
0531          rts
0532          rts
0533          rts
0534          rts
0535          rts
0536          rts
0537          rts
0538          rts
0539          rts
053A          rts
053B          rts
053C          rts
053D          rts
053E          rts
053F          rts
0540          rts
0541          rts
0542          rts
0543          rts
0544          rts
0545          rts
0546          rts
0547          rts
0548          rts
0549          rts
054A          rts
054B          rts
054C          rts
054D          rts
054E          rts
054F          rts
0550          rts
0551          rts
0552          rts
0553          rts
0554          rts
0555          rts
0556          rts
0557          rts
0558          rts
0559          rts
055A          rts
055B          rts
055C          rts
055D          rts
055E          rts
055F          rts
0560          rts
0561          rts
0562          rts
0563          rts
0564          rts
0565          rts
0566          rts
0567          rts
0568          rts
0569          rts
056A          rts
056B          rts
056C          rts
056D          rts
056E          rts
056F          rts
0570          rts
0571          rts
0572          rts
0573          rts
0574          rts
0575          rts
0576          rts
0577          rts
0578          rts
0579          rts
057A          rts
057B          rts
057C          rts
057D          rts
057E          rts
057F          rts
0580          rts
0581          rts
0582          rts
0583          rts
0584          rts
0585          rts
0586          rts
0587          rts
0588          rts
0589          rts
058A          rts
058B          rts
058C          rts
058D          rts
058E          rts
058F          rts
0590          rts
0591          rts
0592          rts
0593          rts
0594          rts
0595          rts
0596          rts
0597          rts
0598          rts
0599          rts
059A          rts
059B          rts
059C          rts
059D          rts
059E          rts
059F          rts
05A0          rts
05A1          rts
05A2          rts
05A3          rts
05A4          rts
05A5          rts
05A6          rts
05A7          rts
05A8          rts
05A9          rts
05AA          rts
05AB          rts
05AC          rts
05AD          rts
05AE          rts
05AF          rts
05B0          rts
05B1          rts
05B2          rts
05B3          rts
05B4          rts
05B5          rts
05B6          rts
05B7          rts
05B8          rts
05B9          rts
05BA          rts
05BB          rts
05BC          rts
05BD          rts
05BE          rts
05BF          rts
05C0          rts
05C1          rts
05C2          rts
05C3          rts
05C4          rts
05C5          rts
05C6          rts
05C7          rts
05C8          rts
05C9          rts
05CA          rts
05CB          rts
05CC          rts
05CD          rts
05CE          rts
05CF          rts
05D0          rts
05D1          rts
05D2          rts
05D3          rts
05D4          rts
05D5          rts
05D6          rts
05D7          rts
05D8          rts
05D9          rts
05DA          rts
05DB          rts
05DC          rts
05DD          rts
05DE          rts
05DF          rts
05E0          rts
05E1          rts
05E2          rts
05E3          rts
05E4          rts
05E5          rts
05E6          rts
05E7          rts
05E8          rts
05E9          rts
05EA          rts
05EB          rts
05EC          rts
05ED          rts
05EE          rts
05EF          rts
05F0          rts
05F1          rts
05F2          rts
05F3          rts
05F4          rts
05F5          rts
05F6          rts
05F7          rts
05F8          rts
05F9          rts
05FA          rts
05FB          rts
05FC          rts
05FD          rts
05FE          rts
05FF          rts
    
```

GTEST.A TEST PROGRAM OVERLAY (continued)

00D8	01	nop
00D9	01	nop
00DA	01	nop
00DB	01	nop
00DC	01	nop
00DD	01	nop
00DE	01	nop
00DF	01	nop
00E0	01	nop
00E1	01	nop
00E2	01	nop
00E3	01	nop
00E4	01	nop
00E5	01	nop
00E6	01	nop
00E7	01	nop
00E8	01	nop
00E9	01	nop
00EA		END

GTEST.A TEST PROGRAM OVERLAY (continued)

area1dat	0011	area1ata	0010	test0a	004A	test0b	004A
begin1	000A	testA	0041	testC	0042	testD	0041
end1	0004	testE	0045	testF	0041	testG	0041
area2	0000	testI	0042	testH	0042	testJ	004B
areaL	0040	testM	0042	testN	0042	testO	004F
areaH	0000	inbreadd	0030	inbreadd1	0001	inbreadd2	0001
out2hex	0050	putb	00FA	putc	004A	putd	009A
test	0001	putl	0011	putm	0047	putn	001B
test2	0007	putr	00B8	putw	00A7	tdr	000C
test3	00FF	putc	0047	tdi	00FF		

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 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 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 the 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

GTEST.A OPERATING INSTRUCTIONS (continued)

intervals to the H89 computer. This data will be displayed on the CRT and stored in the input buffer.

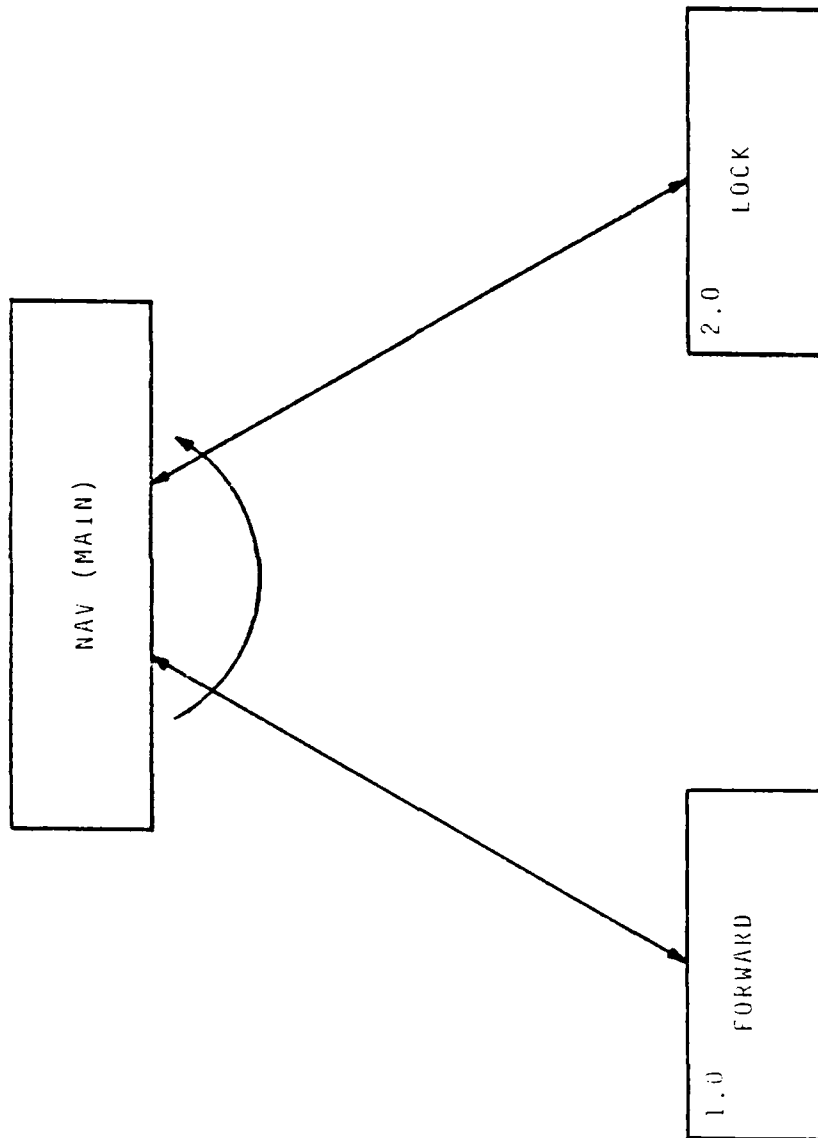
STEP 7: When the robot run is completed (i.e. you have manually stopped it with the MARRS-1 system reset button) the data stored in the input buffer may be written to disk. To do this press the Nav Computer reset button on MARRS-1 and/or type "control C" on the H89. Next type "control shift " followed 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 off the power to the H89 system. Turn off power to the robot, GYRAC, and external power supplies.

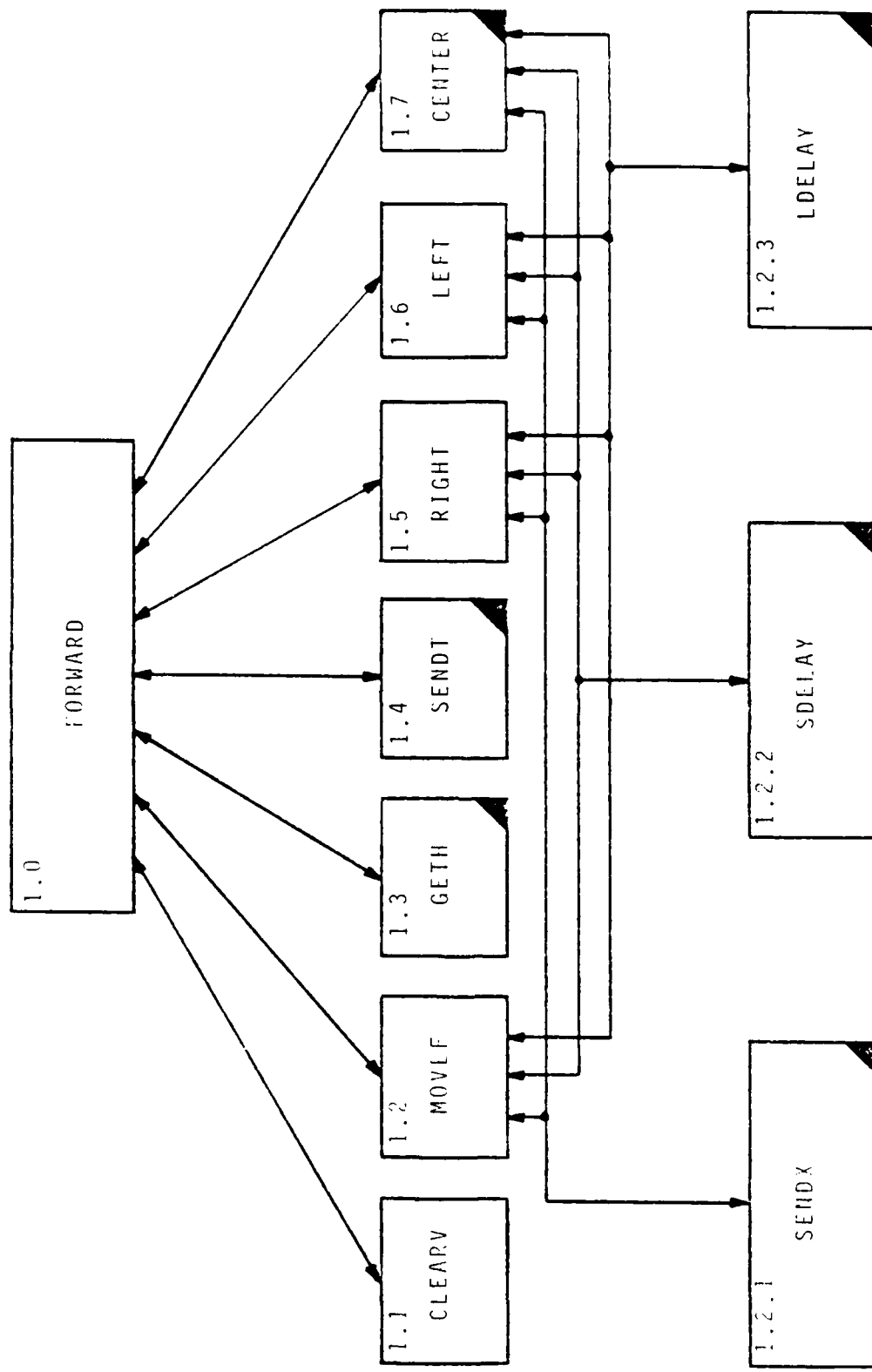
NOTE: All references to "control" and "shift" in the H89 command lines refer to the control and shift keys and not the words control and shift. Care must be taken to ensure the various cables to MARRS-1 do not become tangled during movement. In addition, it is assumed that the robot has been "pointed" 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.

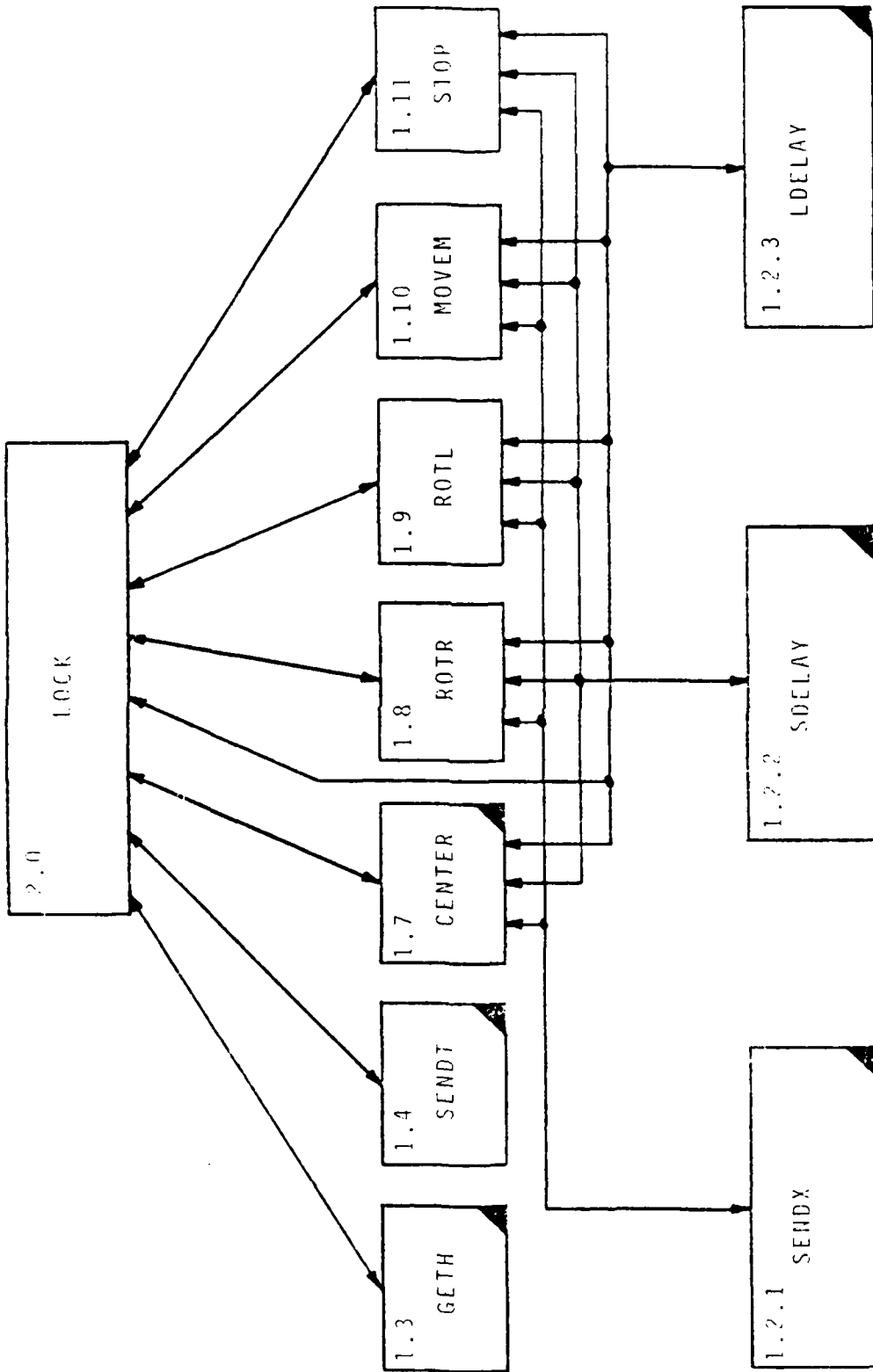
APPENDIX H

NAV.A Structure Charts	H-2
NAV.A Program Listing	H-5
MARRS.NAV Program Listing	H-26
NAV.A Operating Instructions	H-27



NAV.A STRUCTURE CHARTS





NAV.A PROGRAM LISTING

NAV.A PROGRAM LISTING
 DATE: 19/12/85
 VERSION: 1.4
 TITLE: NAVIGATION AND HEADING DATA COLLECTION PROGRAM
 FILENAME: NAV.A
 COORDINATOR: CAPT WILLIAM J. RAHE, USMC
 PROJECT: GYRO AND ACCELEROMETER BASED NAVIGATION SYSTEM FOR A
 MOBILE AUTONOMOUS ROBOT (THESIS)
 OPERATING SYSTEM: IBM PC/XT (IBM VOLTAGE MAGNOLIA MICROSYSTEM 1980)
 LANGUAGE: ROSS-80 ASSEMBLER v1.1 VIRTUAL DEVICE 1984
 USE: This program is assembled and then transferred to the
 robot's navigation computer via an RS-232 link (This must
 be preceded by loading the file MAPRS.NAV to the drive
 computer (MENDS) using M72 in Term mode). NAV.HEX can
 be loaded into the nav computer using M72 in terminal (T)
 mode with the following commands: L,1000,1000 followed by
 transmitting the file NAV.HEX. The heading of travel
 is ghead00, ghead01, and ghead11 which are set at assembly
 time and therefore fixed in NAV.HEX.

DATE: 19/12/85
 VERSION: 1.4
 TITLE: NAVIGATION AND HEADING DATA COLLECTION PROGRAM
 FILENAME: NAV.A
 COORDINATOR: CAPT WILLIAM J. RAHE, USMC
 PROJECT: GYRO AND ACCELEROMETER BASED NAVIGATION SYSTEM FOR A
 MOBILE AUTONOMOUS ROBOT (THESIS)
 OPERATING SYSTEM: IBM PC/XT (IBM VOLTAGE MAGNOLIA MICROSYSTEM 1980)
 LANGUAGE: ROSS-80 ASSEMBLER v1.1 VIRTUAL DEVICE 1984
 USE: This program is assembled and then transferred to the
 robot's navigation computer via an RS-232 link (This must
 be preceded by loading the file MAPRS.NAV to the drive
 computer (MENDS) using M72 in Term mode). NAV.HEX can
 be loaded into the nav computer using M72 in terminal (T)
 mode with the following commands: L,1000,1000 followed by
 transmitting the file NAV.HEX. The heading of travel
 is ghead00, ghead01, and ghead11 which are set at assembly
 time and therefore fixed in NAV.HEX.

CONTENTS: forward
 lock
 clearv
 movev
 getch
 sendt
 sendh
 right
 left
 center
 rptr
 rpr
 moveh
 stop
 sdelay
 ldelay

FUNCTION: This program allows the nav computer to request data
 from the GYRA0 (over an RS-232 link), transmit GYRA0
 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
1001      ghead00    equ      02H        ;MSB L3 nibble given heading in he
1002      ghead01    equ      06H        ;LSB M3 nibble given heading in he
1003      ghead11    equ      0CH        ;LSB L3 nibble given heading in he
1004      c00A       equ      41H        ;0x00A0 command-- to All 0 bytes
1005      c00C       equ      42H        ;0x00C0 command-- to all 1 byte
1006      c00E       equ      43H        ;0x00E0 command-- to heading 0 bytes
1007      c010       equ      44H        ;0x00E0 command-- to heading 1 byte
1008      c012       equ      45H        ;0x00A1 command-- to velocity 0 bytes
1009      c014       equ      46H        ;0x00A1 command-- to velocity 1 byte
100A      c016       equ      47H        ;0x00A0 command-- clear velocity
100B      c018       equ      48H        ;0x00A1 command-- to ang vel 0 bytes
100C      c01A       equ      49H        ;0x00A1 command-- to ang vel 1 byte
100D      c01C       equ      4AH        ;0x00A0 command-- to head 0 vel 0 bytes
100E      c01E       equ      4BH        ;0x00A0 command-- to head 0 vel 1 byte
100F      c020       equ      4CH        ;0x00A0 command-- reset system
1010      c022       equ      4DH        ;0x00A1 command-- to test ad 0 adm
1011      c024       equ      4EH        ;0x00A1 command-- RING test
1012      c026       equ      4FH        ;0x00A1 command-- to head 1 vel 0 bytes
  
```

NAV.A PROGRAM LISTING (continued)

```

1000      aciatstat equ    00000000      ;ACIA serial port status & control reg
1001      aciatdata equ    00000000      ;ACIA serial port data register
1002      acia_start equ    00000000      ;MEND serial port status & control reg
1003      acia_data equ    00000000      ;MEND serial port data register
1004      aciatstat equ    00000000      ;TERMINAL serial port status & control reg
1005      aciatdata equ    00000000      ;TERMINAL serial port data register
1006      tdr equ    00000000      ;ACIA transmit data register empty
1007      rinf equ    00000000      ;ACIA receive data register full
1008      rmask equ    00000000      ;MEO mask for heading data

1009      ;-----
1010      ;Starting address of nav program
1011      ;Rotate robot until facing given head
1012      ;Move forward along given heading
1013      ;-----
1014      DD 10 42 main      ;r   back
1015      DD 10 60          ;r   forward
1016      DD 10 51          ;bra main

```

Reproduced from
 best available copy.

NAV.A PROGRAM LISTING (continued)

```

DATE: 10/12/68
VERSION: 1.1
NAME: forward
MODULE NUMBER: 1.1
DESCRIPTION: This module will steer the drive along a given
            heading (gheadwh, gheadh, & gheadl).
PASSED VARIABLES: None
RETURNING: None
GLOBAL VARIABLES USED: gheadh, gheadl
GLOBAL VARIABLES CHANGED: gheadh, gheadl
REGISTERS USED: A, D
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: clearv
                center
                movef
                geth
                sendh
                right
                left
CALLING MODULES: main
AUTHOR: CAPT WILLIAM J. RAMEY JR.
HISTORY: None
    
```

```

1003 DD 10 A2 forward jsr clearv ;Clear GVARC velocity constant
1004 DD 11 A3 jsr center ;Center the steering wheel
1005 DD 10 AF jsr movef ;Start drive motor fast speed

1011 DD 10 E5 forward jsr geth ;Get D/RAC heading
1012 DD 11 0F jsr sendh ;Send heading to external computer
1013 DD 13 0E ldaa gheadh ;Get MSB of current heading
1014 DD 01 00 cmpa #gheadwh ;If given & current heading are equal
1015 DD 07 beq ifnow ;Then test LSB MSB nibble of heading
1016 DD 10 bpl left ;Else if current > given head turn left
1017 DD 11 00 jsr right ;Else turn right
1018 DD 00 bra forward ;Go get next heading update

1025 DD 13 07 ldaa gheadl ;Get LSB of current heading
1026 DD 44 isra ;Use only the MS nibble
1027 DD 44 isra
1028 DD 44 isra
1029 DD 44 isra
1030 DD 01 0E cmpa #gheadl ;If current > given test LSB MSB of head
1031 DD 07 beq ifnow ;Else if current < given head turn left
1032 DD 11 00 jsr right ;Else turn right
1033 DD 00 bra forward ;Go get next heading update

1041 DD 10 A2 forward jsr center ;Center steering wheel
1042 DD 01 0F jsr movef

1051 DD 11 07 left jsr left ;Turn left
1052 DD 01 0F jsr movef ;Start drive motor fast speed
1053 DD 00 bra forward
    
```

Reproduced from
best available copy.

NAV.A PROGRAM LISTING (continued)

```

DATE: 10/10/75
VERSION: 1.0
NAME: clear
LINE NUMBER: 111
DESCRIPTION: This module clears the GPR velocity constants.
PAIRED VARIABLES: None
RETURN: None
GLOBAL VARIABLES USED: None
GLOBAL VARIABLES CHANGED: None
REGISTERS USED: A, D
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: None
CALLING MODULES: forward
AUTHOR: CAPT WILLIAM C. RAMEY, JR.
EDITOR: None
    
```

```

1040 00 47      clearv      10ab #inn0      ;Get GPRAC clear vel command
1041 06 01 10 00v      10aa acia1stat ;Get GPRAC ACIA status
1042 04 01      ania #dre      ;is transmit data reg empty
1043 17 00      seq 00v      ;if no recheck status
1044 07 00 11      stop acia1data ;Else send GPRAC clear vel command
1045 09          rts
    
```

THE FOLLOWING
 Reproduced from
 best available copy. 
 PAGES

NAV.A PROGRAM LISTING (continued)

```

DATE: 11/10/65
VERSION: 1.0
UNIT: 100
MODULE NUMBER: 100
DESCRIPTION: This file sends a drive motor fast
            command sequence to the drive computer (MENDS).
ENTERED VARIABLES: None
RETURNED: 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: GARY WILLIAM L. RAMEY JR.
DISTR: None
    
```

```

1007 00 41 00000000 1000 # A      ;send to drive computer (MENDS)
1008 00 10 00      ;sr send
1009 00 10 40      ;sr sdelay
            ;sequence

1007 00 44 00000000 1000 # D1
1008 00 10 00      ;sr send
1009 00 10 40      ;sr sdelay

1007 00 39 00000000 1000 # 0
1008 00 10 00      ;sr send
1009 00 10 40      ;sr sdelay

1007 00 31 00000000 1000 # 1
1008 00 10 00      ;sr send
1009 00 10 40      ;sr sdelay

1007 00 30 00000000 1000 # 0
1008 00 10 00      ;sr send
1009 00 10 40      ;sr sdelay

1007 00 29 00000000 1000 # 0
1008 00 10 00      ;sr send
1009 00 10 40      ;sr sdelay

1007 00 10 00000000 1000 # 0
1008 00 10 00      ;sr send
1009 00 10 40      ;sr sdelay

1007 00 10 00000000 1000 # 0
1008 00 10 00      ;sr send
1009 00 10 40      ;sr sdelay
    
```


NAV.A PROGRAM LISTING (continued)

```

DATE: 25/03/78
VERSION: 1.0
NAME: sendr
MODULE NUMBER: 1.1.1
DESCRIPTION: This module sends a type of data to the drive
              driver when MENOS is in RT-STD link.
PAIRED VARIABLES: Output data (passed in D reg)
RETURNS: None
GLOBAL VARIABLES USED: None
GLOBAL VARIABLES CHANGED: None
REGISTERS USED: A, I
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: None
CALLING MODULES: None, right, left, center, rotate, stop
AUTHOR: CAPT WILLIAM J. RAMEY JR.
TITLE: None
    
```

```

1100 D: 00 00 sendr 10aa acia stat ;Get MENOS ACIA status
1140 04 00 acia #dreg ;Is transit data reg empty
1141 01 00 reg sendr ;If no recheck status
1144 07 00 00 stat acia data ;Else send data to MENOS
1147 09 ;Is
    
```

NAV.A PROGRAM LISTING (continued)

```

DATE: 08/21/05
VERSION: 1.00
NAME: sdelg
MODULE NUMBER: 112.12
DESCRIPTION: This module provides for a short fixed time delay
FIXED VARIABLES: None
RETURN: None
GLOBAL VARIABLES USED: None
GLOBAL VARIABLES CHANGED: None
REGISTERS USED: A, D
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: None
CALLING MODULES: None, right, left, center, rotate, stop
AUTHOR: CAPT WILLIAM G. RAMEL, JR.
HISTORY: None
    
```

```

1043 17      sdelay  rshb          ;save registers
1044 18      psba
1045 19  IF      ldaa #ifH          ;Execute fixed time delay
1046 08 FF      sdi      load #ifH
1047 01      sd2      bnp
1048 0A      dcb
1049 11  FC      bne sd2
1050 11      cca
1051 12  FF      bne sd1
1052 12      jula          ;Restore registers
1053 13      pulb
1054 14      rts
    
```

NAV.A PROGRAM LISTING (continued)

```

DATE: 08/11/88
VERSION: 1.0
NAME: tdelay
MODULE NUMBER: 110.0
DESCRIPTION: This module provides a fixed long time delay
POISED VARIABLES: None
EXTERNALS: None
GLOBAL VARIABLES USED: None
LOCAL VARIABLES CHANGED: None
REGISTERS USED: A, Z
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: None
CALLING MODULES: none, right, left, center, rotate, stop, lock
AUTHOR: DART WILLIAM J. FAMELUP
HISTORY: None
    
```

```

0000 07      tdelay  pshb      ;Save registers
0001 08      pshb
0002 09 09      pshb #00FH ;Execute time delay
0003 10 0F      ldi #00FH
0004 11      pop
0005 12      lach
0006 13 00      tne #00
0007 14 00      deca
0008 15 07      tne #01
0009 16      pola
000A 17      push
000B 18      rts

000C      check  r0: 1
000D      check  r1: 1

000E      END
    
```


NAV.A PROGRAM LISTING (continued)

```

DATE: 08/20/88
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.
RAISED 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, ltrn
AUTHOR: CAPT WILLIAM G. RAHEY JR.
HISTORY: None
    
```

```

1108 05 10 62 sendt 1dab cheadm ;Get MSB of current heading
1110 04          1snb          ;Convert upper nibble of MSB to ASCII
1110 04          1snb
1114 04          1snb
1115 04          1snb
1116 04 0F        andb #0FH
1118 00 00        addb #00H
111A 01 00        cnzb #00H
111C 00 00        ors  01
111B 00 00        addc #07H
111D 00 00 @ 01 1daa aciatstat ;Get terminal ACIA status
111E 04 00        anda #00H     ;is transmit data reg empty
111F 07 07        beq  01       ;if no receive status
1120 07 00 01     staa aciatdata ;Else send data

112A 05 11 68     1dab cheadm   ;Get MSB of current heading
112B 04 0F        andb #0FH     ;Convert lower nibble of MSB to ASCII
112C 00 00        addb #00H
112E 01 00        cnzb #00H
112F 00 00        ors  01
1130 00 00        addc #07H
1132 00 00 @ 01 1daa aciatstat ;Get terminal ACIA status
1133 04 00        anda #00H     ;is transmit data reg empty
1134 07 07        beq  01       ;if no receive status
1136 07 00 01     staa aciatdata ;Else send data

1141 05 12 69     1dab cheadl   ;Get LSB of current heading
1142 04          1snb          ;Convert upper nibble of LSB to ASCII
1143 04          1snb
1144 04          1snb
1145 04          1snb
1146 04 0F        andb #0FH
1148 00 00        addb #00H
114A 01 00        cnzb #00H
114C 00 00        ors  01
114B 00 00        addc #07H
114E 00 00 @ 01 1daa aciatstat ;Get terminal ACIA status
114F 04 00        anda #00H     ;is transmit data reg empty
1150 07 07        beq  01       ;if no receive status
1152 07 00 01     staa aciatdata ;Else send data
    
```

NAV.A PROGRAM LISTING (continued)

```

1150  F 10 65      ldat #headl      ;Get LID of current heading
1151  D4 1F      addr #1FH      ;Insert lower nibble of LID to ASCII
1152  D0 70      addr #0FH      ;
1153  D1 70      addr #0FH      ;
1154  D0 70      chr #4      ;
1155  D0 70      addr #0FH      ;
1156  D0 70 00 64  ldaa aciastat    ;Get terminal ASCII status
1157  D4 70      anda #d0e    ;Is transmit data reg empty
1158  D1 70      bcc #4      ;If no receive status
1159  F7 70 01      stac aciastat    ;Else send data

1171  D0 70      ldat #00H      ;Get carriage return
1172  D0 70 00 64  ldaa aciastat    ;Get terminal ASCII status
1173  D4 70      anda #00e    ;Is transmit data reg empty
1174  D1 70      bcc #4      ;If no receive status
1175  F7 70 01      stac aciastat    ;Else send data

1177  D0 00      ldat #0AH      ;Get line feed
1178  D0 70 00 64  ldaa aciastat    ;Get terminal ASCII status
1179  D4 70      anda #00e    ;Is transmit data reg empty
1180  D1 70      bcc #5      ;If no receive status
1181  F7 70 01      stac aciastat    ;Else send data

1182  D0 00      rts

```


NAV.A PROGRAM LISTING (continued)

```

DATE: 10/22/65
VERSION: 1.00
NAME: left
MODULE NUMBER: 1.3
DESCRIPTION: This module sends a left turn relative mode
              command sequence to the driver computer (MEND)
RAISED 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, sendr
CALLING MODULES: forward
AUTHOR: CAPT WILLIAM G. RAMEY JR.
HISTORY: None
    
```

```

1109 16 41 left ltab #1A ;Send to drive computer (MEND)
1110 20 13 30 jsr sendr ;left turn continuous command
1111 20 13 43 jsr sdelay ;sequence

1112 16 44 ltab #1D
1113 20 13 30 jsr sendr
1114 20 13 43 jsr sdelay

1115 16 50 ltab #1E
1116 20 13 30 jsr sendr
1117 20 13 43 jsr sdelay

1118 16 51 ltab #1F
1119 20 13 30 jsr sendr
1120 20 13 43 jsr sdelay

1121 16 52 ltab #1G
1122 20 13 30 jsr sendr
1123 20 13 43 jsr sdelay

1124 16 54 ltab #1H
1125 20 13 30 jsr sendr
1126 20 13 43 jsr sdelay

1127 16 55 ltab #1I
1128 20 13 30 jsr sendr
1129 20 13 43 jsr sdelay

1130 16 56 ltab #1J
1131 20 13 30 jsr sendr
1132 20 13 43 jsr sdelay
    
```


NAV.A PROGRAM LISTING (continued)

```

DATE: 04/11/55
FORM: 11.1
NAME: 11.1
MODULE NUMBER: 11.1
DESCRIPTION: This module will issue the forward sequence to
            cause the steering wheel to move to the right
            will rotate right when the drive motor is turned
            on.
ENTERED VARIABLES: None
RETURN: None
GLOBAL VARIABLES USED: None
GLOBAL VARIABLES CHANGED: None
REGISTERS USED: A, B
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: send, sdelay, delay
CALLING MODULES: None
AUTHOR: CAPT WILLIAM J. RAMEY JR.
REVISION: None
    
```

1141	01 41	notr	tab # A1	;Send to drive computer (MENGES)
1142	00 12 00		sr send	turn 30 deg right steering motor
1143	00 12 40		sr sdelay	command sequence
1144	01 44		tab # D1	
1145	00 12 00		sr send	
1146	00 12 40		sr sdelay	
1147	01 19		tab # W	
1148	00 12 00		sr send	
1149	00 12 40		sr sdelay	
1150	01 11		tab # 1	
1151	00 12 00		sr send	
1152	00 12 40		sr sdelay	
1153	01 11		tab # 11	
1154	00 12 00		sr send	
1155	00 12 40		sr sdelay	
1156	01 01		tab # D1	
1157	00 12 00		sr send	
1158	00 12 40		sr sdelay	
1159	00 12 50		sr delay	
1160	00 12 50		sr delay	
1161	00 12 50		sr delay	
1162	00 12 50		sr delay	
1163	00 12 50		sr delay	
1164	00 12 50		sr delay	
1165	00 12 50		sr delay	
1166	00 12 50		sr delay	
1167	00 12 50		sr delay	
1168	00 12 50		sr delay	
1169	00 12 50		sr delay	
1170	00 12 50		sr delay	
1171	00 12 50		sr delay	
1172	00 12 50		sr delay	
1173	00 12 50		sr delay	
1174	00 12 50		sr delay	
1175	00 12 50		sr delay	
1176	00 12 50		sr delay	
1177	00 12 50		sr delay	
1178	00 12 50		sr delay	
1179	00 12 50		sr delay	
1180	00 12 50		sr delay	
1181	00 12 50		sr delay	
1182	00 12 50		sr delay	
1183	00 12 50		sr delay	
1184	00 12 50		sr delay	
1185	00 12 50		sr delay	
1186	00 12 50		sr delay	
1187	00 12 50		sr delay	
1188	00 12 50		sr delay	
1189	00 12 50		sr delay	
1190	00 12 50		sr delay	
1191	00 12 50		sr delay	
1192	00 12 50		sr delay	
1193	00 12 50		sr delay	
1194	00 12 50		sr delay	
1195	00 12 50		sr delay	
1196	00 12 50		sr delay	
1197	00 12 50		sr delay	
1198	00 12 50		sr delay	
1199	00 12 50		sr delay	

NAV.A PROGRAM LISTING (continued)

```

DATE: 07 11 65
VERSION: 1.1
NAME: stop
MODULE NUMBER: 1.11
DESCRIPTION: This module sends the command sequence to stop
the drive and steering motors.
RAISED VARIABLES: None
RETURNS: None
GLOBAL VARIABLES USED: None
LOCAL VARIABLES CHANGED: None
REGISTERS USED: A, B
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: sendr, sdelay, delay
CALLING MODULES: lock
AUTHOR: CAPT WILLIAM D. RAMEY, JR.
HISTORY: None
    
```

```

1006 06 41 stop 1tab #1A 1send to drive computer (MENDS)
1006 00 13 00 jsr sendr 1stop steering and drive motor command
1006 00 13 48 jsr sdelay 1sequence

100E 06 44 1tab #01
1010 00 13 00 jsr sendr
1010 00 13 48 jsr sdelay

1014 06 38 1tab #0
1014 00 13 00 jsr sendr
1014 00 10 48 jsr sdelay

101E 06 31 1tab #11
101E 00 13 00 jsr sendr
101E 00 13 48 jsr sdelay

1024 06 21 1tab #11
1024 00 13 00 jsr sendr
1024 00 13 48 jsr sdelay

102E 06 10 1tab #01
102E 00 13 00 jsr sendr
102E 00 13 48 jsr sdelay

1034 00 11 50 jsr delay
1034 00 11 90 jsr delay
1034 00
    
```

NAV.A PROGRAM LISTING (continued)

```

DATE: 10/12/68
VERSION: 1.0
NAME: rdt
MODULE NUMBER: 1.0
DESCRIPTION: This module will rotate the robot until it is
              within 1.5 degrees of the given heading.
PASSED VARIABLES: None
RETURNS: None
LOCAL VARIABLES USED: cheadn, cheadl
LOCAL VARIABLES CHANGED: cheadn, cheadl
REGISTERS USED: A, B
FILES READ: None
FILES WRITTEN: None
MODULES CALLED: center
                 moven
                 geth
                 sendt
                 rdtl
                 rdtl
                 stop
                 delay
CALLING MODULES: main
AUTHOR: CAPT WILLIAM G. RAMEY JR.
REVISION: None
    
```

```

1042 00 17 00 0000  lsr  geth      ;Get G FAC heading
1045 00 11 00 0000  lsr  sendt     ;Send heading to external computer
1048 00 10 00 0000  ldaa  cheadn   ;Get MSB of current heading
1048 01 03 0000  cmpa  #theadn   ;If given & current heading are equal
1040 01 07 0000  beq  1044     ;Then test LDB MSB of current heading
1047 00 17 0000  bnl  1051     ;Else if current & given to left
1051 00 11 00 0000  lsr  rdtl     ;Else move steering wheel 30 right
1054 00 17 0000  bra  1045

*
1056 00 10 00 0000  ldaa  cheadl   ;Get LSB of current heading
1057 00 04 0000  lsrn  4        ;Use only the MS nibble
1059 00 04 0000  lsrn  4
1059 00 04 0000  lsrn  4
1059 00 04 0000  lsrn  4
1059 01 03 0000  cmpa  #theadl   ;If current heading = given heading
1059 01 03 0000  beq  1061     ;Then then move forward
1061 00 17 00 0000  lsr  rdtl     ;Else if current & given head to left
1061 00 11 00 0000  lsr  rdtl     ;Else move steering wheel 30 right
1066 00 17 0000  bra  1045

*
1068 00 10 00 0000  lsr  rdtl     ;Move steering wheel 30 deg left
1068 00 17 0000  bra  1045

*
1070 00 10 00 0000  lsr  moven     ;Start drive with max speed
1070 00 10 00 0000  lsr  geth     ;Get G FAC heading
1070 00 10 00 0000  ldaa  cheadn   ;Get MSB of current heading
1070 01 03 0000  cmpa  #theadn   ;If given & current heading are equal
1070 01 07 0000  beq  1072     ;Then test LDB MSB of current heading
1070 00 17 00 0000  bnl  1074     ;Else go get next heading & update
    
```

NAV.A PROGRAM LISTING (continued)

```

1075 05 12 02 10012 1000 ahead1 1000 0000 current heading
1076 44 1000 1000 1000 1000 1000 1000 1000 1000
1077 44 1000 1000 1000 1000 1000 1000 1000 1000
1078 44 1000 1000 1000 1000 1000 1000 1000 1000
1079 44 1000 1000 1000 1000 1000 1000 1000 1000
1080 11 00 1000 1000 1000 1000 1000 1000 1000 1000
1081 17 00 1000 1000 1000 1000 1000 1000 1000 1000
1082 10 00 1000 1000 1000 1000 1000 1000 1000 1000
1083 10 12 00 10000 1000 1000 1000 1000 1000 1000
1084 10 12 00 1000 1000 1000 1000 1000 1000 1000
1085 10 10 00 1000 1000 1000 1000 1000 1000 1000
1086 10 10 00 1000 1000 1000 1000 1000 1000 1000
1087 10 10 00 1000 1000 1000 1000 1000 1000 1000
1088 10 10 00 1000 1000 1000 1000 1000 1000 1000
1089 10 10 00 1000 1000 1000 1000 1000 1000 1000
1090 10 10 00 1000 1000 1000 1000 1000 1000 1000
1091 10 10 00 1000 1000 1000 1000 1000 1000 1000

```

NAV.A PROGRAM LISTING (continued)

actdat	1011	acta sta	1011	actdat	1011	actata	1000
actdat	1012	acta sea	1012	actat	1012	actar	1001
thead1	1013	theadw	1013	thead	1013	thead	1002
act	1014	act	1014	act	1014	act	1003
act	1015	act	1015	act	1015	act	1004
act	1016	act	1016	act	1016	act	1005
act	1017	act	1017	act	1017	act	1006
act	1018	act	1018	act	1018	act	1007
act	1019	act	1019	act	1019	act	1008
act	1020	act	1020	act	1020	act	1009
act	1021	act	1021	act	1021	act	1010
act	1022	act	1022	act	1022	act	1011
act	1023	act	1023	act	1023	act	1012
act	1024	act	1024	act	1024	act	1013
act	1025	act	1025	act	1025	act	1014
act	1026	act	1026	act	1026	act	1015
act	1027	act	1027	act	1027	act	1016
act	1028	act	1028	act	1028	act	1017
act	1029	act	1029	act	1029	act	1018
act	1030	act	1030	act	1030	act	1019
act	1031	act	1031	act	1031	act	1020
act	1032	act	1032	act	1032	act	1021
act	1033	act	1033	act	1033	act	1022
act	1034	act	1034	act	1034	act	1023
act	1035	act	1035	act	1035	act	1024

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"

NAV.A OPERATING INSTRUCTIONS (continued)

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 off 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 I-2
POSITION.BAS Program Listing I-4

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",READFILES$
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",PRINTFILES$
60 OPEN "I",#1,READFILES$
70 OPEN "O",#2,PRINTFILES$
REM*
REM*   A DATA STRING IS READ AND THEN EACH DATA SEGMENT IS
REM*   CONVERTED AND STORED ON DISK
REM*
80 INPUT#1,DATALINES$
90 IF EOF(1) THEN END
100 WORDS$ = MIDS$(DATALINES$,3,4)
110 GOSUB 300
120 PRINT#2,VALUE%," ";
130 FOR I% = 11 TO 26 STEP 5
140 WORDS$ = MIDS$(DATALINES$,I%,4)
150 GOSUB 300
160 GOSUB 450

```

CONVERT.BAS PROGRAM LISTING (continued)

```

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).

```

POSITION.BAS PROGRAM LISTING (continued)

```

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

Phase II Sample Test Data..... J-2

GYRAC Phase II Sample Test Data

TEST #2B (SENSITIVITY = 0.393)

TIME(SEC)	POSITION	
	X(FT)	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	4.62
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	K-2
Gyro Navigation Test Run Number 2	K-3
Gyro Navigation Test Run Number 3	K-4

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	87.36328125	- 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
3E8	1000	87.890625	- 4	- 0.3515625
3E8	1000	87.890625	- 4	- 0.3515625
3E7	999	87.80273438	- 5	- 0.439454125
3E7	999	87.80273438	- 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 L-2

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

Quantity	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

AD A164036

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS			
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited			
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
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.			
11. TITLE (Include Security Classification) on back		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
		12. PERSONAL AUTHOR(S) Bloom, Roland J. and Ramey, William J. Jr.			
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM 85 Jan to 85 Dec	14. DATE OF REPORT (Yr., Mo., Day) 1985/12/2	15. PAGE COUNT 251		
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary)			
FIELD	GROUP	Robot, Robotics, Robot Navigation			
17	07				
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
<p>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.</p>					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS <input type="checkbox"/>		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		

Approved for Public Release: LAW AFB 190-17
 Lynn E. Wolaver 16 JAN 86
 Dept. for Research and Professional Development
 Wright-Patterson AFB OH 45433

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. (746...)

END

FILMED

3 - 86

DTIC