IMPLEMENTATION OF A REMOTE COMPUTER CONTROLLED AUTOMATIC GUIDED VEHICLE

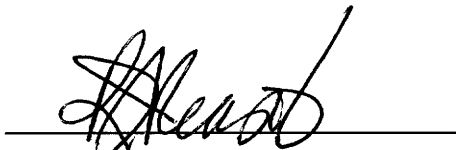
by
Roberto F. Lu

Project report submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in the partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING
in
Industrial and Systems Engineering

APPROVED:

Dr. Michael P. Deisenroth



Dr. Roderick J. Reasor

Dr. Ralph D. Badinelli

March, 1994
Blacksburg, Virginia

LD
5655
V851
1994
L8
C.2

C.2

IMPLEMENTATION OF A REMOTE COMPUTER CONTROLLED AUTOMATIC GUIDED VEHICLE

by
Roberto F. Lu

Committee Chairman: Dr. Michael P. Deisenroth
Industrial and Systems Engineering

ABSTRACT

The effectiveness of a material handling system is essential to a competitive manufacturing environment. Automatic Guided Vehicles (AGVs) are an important technology within today's modern manufacturing facility. Academic programs in manufacturing and industrial engineering must find ways to include this technology in their instructional and research programs to provide the students with sufficient knowledge to address material handling systems design. This project was a first step in the development of such a facility at Virginia Polytechnic Institute and State University.

This project accomplished two main objectives. The first one was to create a functional AGV system with the "Orbitor" Automatic Guided Vehicle that incorporates advanced hardware features. This system included a taped guide path with bar codes on the floor, and the communication linkage via wire and radio frequency between the Orbitor on-board computer and a remote computer. The second objective was to establish a software system for the overall control of the Orbitor AGV system. This objective involved creating software for the on-board computer and the remote computer. Software communication protocols were implemented and tested between the remote computer and the Automatic Dispatching Unit, and between the Intersection and Communications Unit and the

on-board computer. The user of this system was able to interactively define tasks for the AGV from the remote computer. The AGV provided feedback to the remote computer on the status of the vehicle and it sent appropriate messages when the tasks were complete.

ACKNOWLEDGMENTS

In the course of this project, many people have assisted me and offered their support. I am most grateful to my advisor, Dr. Michael P. Deisenroth, who suggested this project topic and who has been most generous with his valuable time and knowledge. His support and important guidance have been very much appreciated.

I would like to thank my parents for their support and encouragement. Special gratitude is expressed to my wife, Sophia, for her patience, understanding, and cheerful forbearance particularly during the final days of the preparation. The efforts they made were unimaginable and this can only acknowledge them for a fraction of my gratitude.

I am deeply thankful to the other two committee professors: Dr. Roderick J. Reasor and Dr. Ralph D. Badinelli for their valuable suggestions, advices, and reviewing of this project.

I would to thank the ISE Department of the Virginia Tech for the use of its facilities during the years of my graduate study. I also appreciate my friends Mr. Bill Mabry, Mr. Dan Arrington, Mr. Niels Ketelhohn, and Mr. Rosendo Molina for their helps in the earlier stage of this project.

Thanks goes to the gentlemen of the Apogee Robotics, Inc. for their unselfish technical assistance. I am also pleased to acknowledge my indebtedness to my bosses of the New River Castings Company for the opportunities they have given me and for their generous finical support throughout my graduate study at the Virginia Tech.

TABLE OF CONTENTS

ABSTRACT ii
ACKNOWLEDGMENTS iv
TABLE OF CONTENTS v
LIST OF TABLES viii
LIST OF FIGURES ix
TRADEMARK ACKNOWLEDGMENTS x

CHAPTER 1: INTRODUCTION
1.1 Project Motivation 1
1.2 Project Objectives 2
1.3 Project History 3

CHAPTER 2: THE APOGEE AGV BASICS
2.1 The Elements of the Apogee AGV System 5
2.2 Vehicle Basics 7
2.3 Guide Path and Bar Codes 8
2.4 Control Systems Components 12
2.5 Available Reference Materials of the AGV 16

CHAPTER 3: FUNDAMENTAL CONTROL SYSTEM CONCEPTS
3.1 The AGV System Control Overview 19
3.2 The Operations of the AGV 20
3.3 Vehicle Control Computer Concepts 21
3.3.1 Bar Code Input Information 22
3.3.2 HP-71B Command Codes 24

CHAPTER 4: THE BASIC AGV PROGRAMMING
4.1 The On-board HP-71B Computer of the AGV 30
4.2 The Editing Environment of the HP-71B 32
4.3 General HP-71B BASIC Programming 34
4.4 HP-71B Interaction With the Vehicle Controller 37

TABLE OF CONTENTS

CHAPTER 5: REMOTE COMPUTER BASIC AVG INTERACTION
5.1 The Communication System 43
5.2 Remote Computer Configuration 45
5.3 Software Communication Concepts 46
5.3.1 Message Formats 47
5.3.2 Communications Software Considerations - Remote Computer. 49
5.3.3 Communications Software Considerations - HP-71B Computer. 52
5.3.4 Communications Software Considerations - Both Systems 54

CHAPTER 6: EXAMPLE OF REMOTE COMPUTER DRIVEN AGV SYSTEM
6.1 The Demonstration Guide Path 56
6.2 Demonstration Software Protocol 57
6.3 The HP-71B Program 61
6.3.1 General Program Organization 61
6.3.2 General Program Flow 63
6.3.3 Line-by-Line Explanation 66
6.4 The Remote Computer Program 71
6.4.1 General Program Organization 71
6.4.2 General Program Flow 73
6.4.3 Line-by-Line Explanation 74
6.5 The ICUTEST Program 77

CHAPTER 7: KNOWN PROBLEMS AND FUTURE OPPORTUNITIES
7.1 The AGV Hardware Problems 78
7.2 The Opportunity for Future Studies 79

CHAPTER 8: CONCLUSION AND RECOMMENDATIONS 83

REFERENCES 85

TABLE OF CONTENTS

APPENDIX A: The Example Program of the AGV On-board Computer 86

APPENDIX B: The Example Program of the AGV Remote Computer 109

APPENDIX C: The ICUTEST Program of the HP-71B Compuper 126

APPENDIX D: The Template for the Floor Bar Code 127

LIST OF TABLES

Table	Title	Page
1	W String Character Codes	22
2	HP-71B Orbitor Command Codes	24
3	A Command Codes	25
4	Vehicle Controller Query/Status Response Codes	27
5	S Command Codes	28
6	HP-71B Command Code Details	39
7	Example HP-71B Command Codes	40
8	Layout of the RS-232 Device Bit Scheme	42
9	The Pin Functions of the IBM 25 Pin Serial Port	45
10	The Assignment of Floor Bar Code Mapping W Strings	58
11	Message Formats of the Demonstration Software	60

LIST OF FIGURES

Figure	Title	Page
1	Apogee AGV System Overview	5
2	The Orbitor	8
3	The Guide Path Installation With the Sonar On	10
4	The Floor Bar Codes	11
5	The Overview of the HP-IL Communications Loop	12
6	The Components of the ICU	14
7	The Components of the ADU	15
8	Configuration of the RS-232 Cable	16
9	Communication System Components	43
10	The Demonstration Guide Path	56
11	The Proposed Guide Path for the New Engineering Building	82
12	The Flow Chart of the Structure of the Example Program THEO	86
13	The Flow Chart of the THEO Main Program	87
14	The Flow Chart of the CHKBAT Subroutine	88
15	The Flow Chart of the CHKTOTE Subroutine	89
16	The Flow Chart of the DELIVER Subroutine	90
17	The Flow Chart of the INITLIFT Subroutine	91
18	The Flow Chart of the LOOK Subroutine	92
19	The Flow Chart of the CHECK Subroutine	93
20	The Flow Chart of the PICKUP Subroutine	94
21	The Flow Chart of the TRAVEL Subroutine	95
22	The Flow Chart of the CHKSUM Subroutine (in THEO)	96
23	The Flow Chart of the Structure of the Example Program REMOTE1.BAS	109
24	The Flow Chart of the REMOTE1.BAS Main Program	110
25	The Flow Chart of the CHKSUM Subroutine (in REMOTE1)	111
26	The Flow Chart of the CLEARLOC Subroutine	112

TRADEMARK ACKNOWLEDGMENTS

Apogee and Orbitor are registered trademarks of the Apogee Robotics, Inc.

AT&T is a registered trademark of American Telephone and Telegraph Company.

HP, HP-71B, and HP-IL are registered trademarks of the Hewlett-Packard Company.

IBM and IBM AT are registered trademarks of the International Business Machines Corporation.

MS-QBASIC and MS-DOS are registered trademarks of the Microsoft Corporation.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies.

CHAPTER ONE
INTRODUCTION

1.1 Project Motivation

The effectiveness of a material handling system is essential to a competitive manufacturing environment. Material handling operations can not contribute to the final value of a product for consumers. They will only increase the manufacturing cost. Automatic Guided Vehicles (AGVs) can be effective in reducing the material handling cost and increasing the overall productivity of manufacturing and non-manufacturing companies. An AGV is a vehicle that operates unattended and can perform various material handling tasks. Academic programs in manufacturing and industrial engineering must find ways to include this technology in their undergraduate and graduate programs.

The Department of Industrial and Systems Engineering (ISE) at Virginia Polytechnic Institute and State University has one such vehicle available in the Robotics and Automation Laboratory (RAL). The Apogee Robotics, Inc "Orbitor" was previously used at the AT&T plant in Fairlawn Virginia and was donated to the ISE Department for use in instructional and research programs. This vehicle has been used by students in projects in the Facilities Planning and Material Handling class. The vehicle was designed to handle light weight loads. It can determine the desired route based upon information encoded in the bar codes that are taped to the floor and from messages that are transmitted from a remote computer to the Orbitor. The class projects conducted during the past have been limited in scope and have not taken advantages of advanced features of the Orbitor.

1.2 Project Objectives

The primary goal of this project was to provide a foundation for planning and implementing a fully functional AGV system that would include remote control of the Orbitor from a personal computer. There were two specific objectives in this project. The first objective was to test the various features of the vehicle to see if they worked and to document known problems that could not be overcome. This involved a thorough study of hardware documentation and the implementation of a small guide path to serve as a testbed. The testbed was needed for both independent vehicle operations and for vehicle-remote computer integration.

The second objective was to establish a software system for the overall control of the Orbitor AGV System. This involved creating basic transportation task software for the Orbitor, creating communication routines for both the Orbitor and the remote computer, and creating a menu driven program for the remote computer to interact with the operator and the vehicle. This software system would be used to demonstrate system functionality and it could also serve as a reference for future software systems development efforts.

This project provides an illustration of how to integrate the AGV system components into a total system. As such, it is a meaningful first step in the creation and implementation of a more elaborate system for use within the ISE research and instructional programs.

The final step of this project was to document the knowledge gained from this work so that others might implement more sophisticated control system with the Orbitor. This report is the result of that process.

1.3 Project History

To implement a fully functional AGV system, a number of basic problems needed to be addressed at the outset of the project. The basic hardware and software features of the vehicle had to be studied and tested for functionality. A small multi loop layout was created to test the system both as an independently operated vehicle and as a vehicle that received commands from a remote computer. Programs were written for the vehicle and the remote computer to implement communications. Finally, a program was written for the remote computer that could interactively send commands to the vehicle and receive appropriate feedback. Only after these steps were taken could plans for a fully functional system be developed that could have a high probability of actually being realized.

While prior projects with the vehicle had been completed, no attempt had been made to fully test the functionality of the Orbitor and document the results. Previous projects were limited to independent vehicle operations and simply bypassed problems that were encountered. The first major step of this project was to test the various features at the vehicle on a small loop with no vehicle-remote interaction. A loop was created that incorporated both branching and multiple P/D stations. All of the barcode readers and sensors were tested for functionality. A complete understanding of the vehicle was obtained before remote computer operations were undertaken.

Two other major problem areas that were addressed were the creation of software for the Orbitor and the remote computer and the integration of the remote computer into the Orbitor system. Programs were developed for the Orbitor to perform basic transportation tasks in a multi loop, multi P/D station

environment. Communication software for both the vehicle and the remote computer was written and debugged. Additionally, an operator interactive program for the remote computer was developed to illustrate remote computer control concepts and to test system performance. Apogee documentation provided specifications of communication strings that pass between the remote computer and the vehicle. However, these features had not been implemented and the hardware software interfaces needed to be tested. Software was developed for the remote computer to create this dialogue in a meaningful fashion. Details of these programs follow in subsequent chapters.

CHAPTER TWO
THE APOGEE AGV BASICS

2.1 The Elements of the Apogee AGV System

An Automatic Guided Vehicle (AGV) is a vehicle that operates unattended and can perform a variety of material handling tasks. In this project, an Apogee Robotics Inc. "Orbitor" was used that had previously been installed at the AT&T plant in Fairlawn Virginia. The total AGV system consisted of three major components: the vehicle, an Automatic Dispatching Unit (ADU), and a remote computer. The vehicle can be further divided into three subsystems: the HP-71B computer, the Motorola 6809 microprocessor and the Intersection and Communications Unit (ICU). This is illustrated in Figure 1.

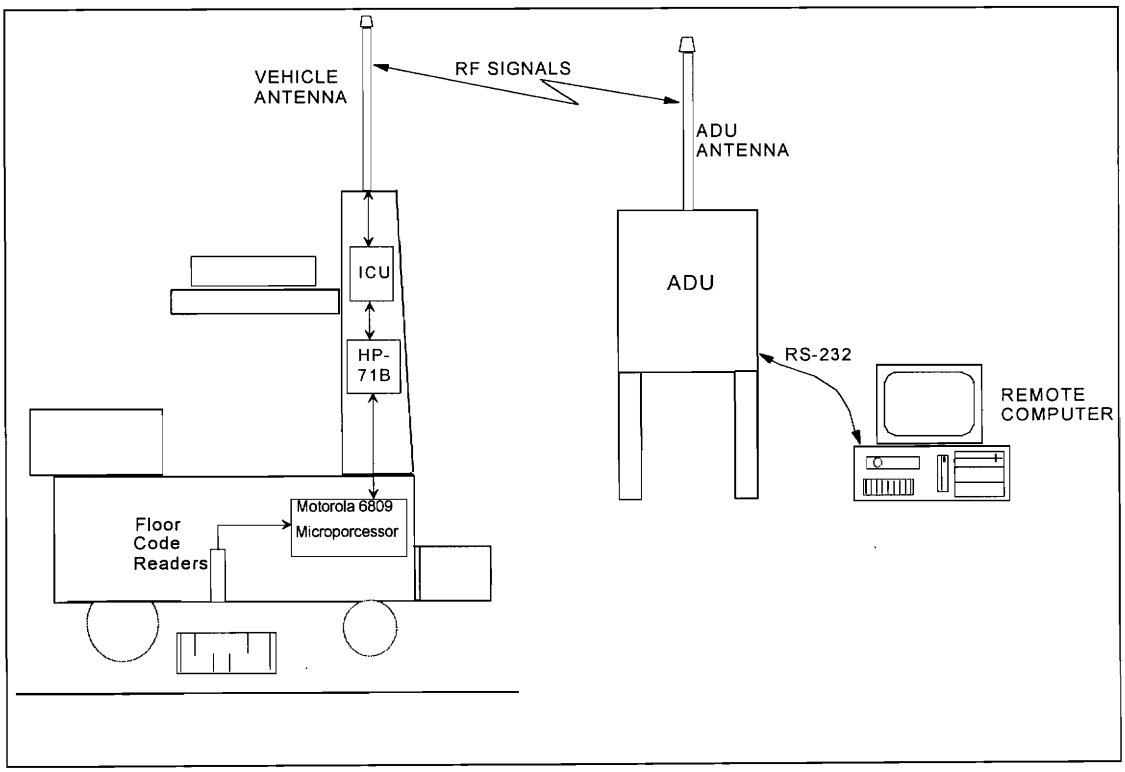


Figure 1 Apogee AGV System Overview

The Apogee Orbitor has the ability to be manually maneuvered by a Manual Control Pendant. The Manual Control Pendant is connected to the rear of the Orbitor when moving the Orbitor from one location to another without any guide path. It provides four switches that controls the left motor, the right motor, the lift assembly, and the arm of the lift. In the manual mode, the Orbitor can be driven in reverse.

The Orbitor can be programmed directly by the user for both independent and integrate operations. The Orbitor follows a guide path of retro-reflective tape by use of optical sensors. Additionally, barcodes, constructed from retro-reflective tape, are placed along the guide path near branches or Pickup/Discharge (P/D) stations for information input to the vehicle. User programming of the vehicle is done by programming the on-board HP-71B computer. All material handling tasks, routings, and trouble shootings of the Orbitor is controlled by commands given from the HP-71B to the Motorola 6809 microprocessor of the Orbitor. The Motorola 6809 Microprocessor is responsible for high speed lower level input and output functions that include device controls, timing, guide path and bar code sensing, and directional decision making. The HP-71B is also interfaced with a remote computer for integrated operations in a more complex AGV system setup.

The Orbitor has an Intersection and Communications Unit (ICU) that is used for vehicle to remote computer communication. It can also be used for vehicle to vehicle communication for intersection traffic controls. The Automatic Dispatching Unit (ADU) is a floor mounted communications unit that connects directly to the remote computer and communicates with the vehicle mounted ICU through radio link. The ADU and remote computer are connected by an RS232

connection. The ADU allows the remote computer to send instructions and read status information to and from the Orbitor.

2.2 Vehicle Basics

The vehicle (shown in Figure 2) can pick-up and discharge a load without any operator interaction. The maximum payload capacity of the vehicle lift assembly is 55 pounds. Tote size can very from 13 by 10 inches to 24 by 20 inches. The lift assembly of the Orbitor can be programmed to any height between 24 and 50 inches from the floor. Loading/Unloading can only be done on one side of the vehicle. The load can be placed in a tote box which is bar coded for its identity. The Orbitor is equipped with a second bar code reader to distinguish between tote boxes. A sonar sensor that is located at the extendible end of the lift assembly can be programmed to detect if an obstacle exists on the shelf to which the Orbitor has a load to deliver or to pick-up.

There are three different sets of safety devices available on the Orbitor. They are the safety bumpers, ultrasonic sonar detectors, and the E-Stop button. Touch sensitive front and side bumpers will stop the vehicle immediately upon contact with any obstructions or objects. Three programmable ultrasonic sonar detectors scan the forward path for obstacles and allow the vehicle to slow down automatically and stop. The Orbitor will stop immediately when the E-Stop button is pressed. An audio warning is available through user defined programs.

The Orbitor is powered by three twelve-volt automobile batteries in the battery compartment. The batteries are linked in parallel and can be charged by using a regular twelve-volt automobile battery charger. The voltage status of

many components in the Orbitor can be obtained by the on-board HP-71B computer.

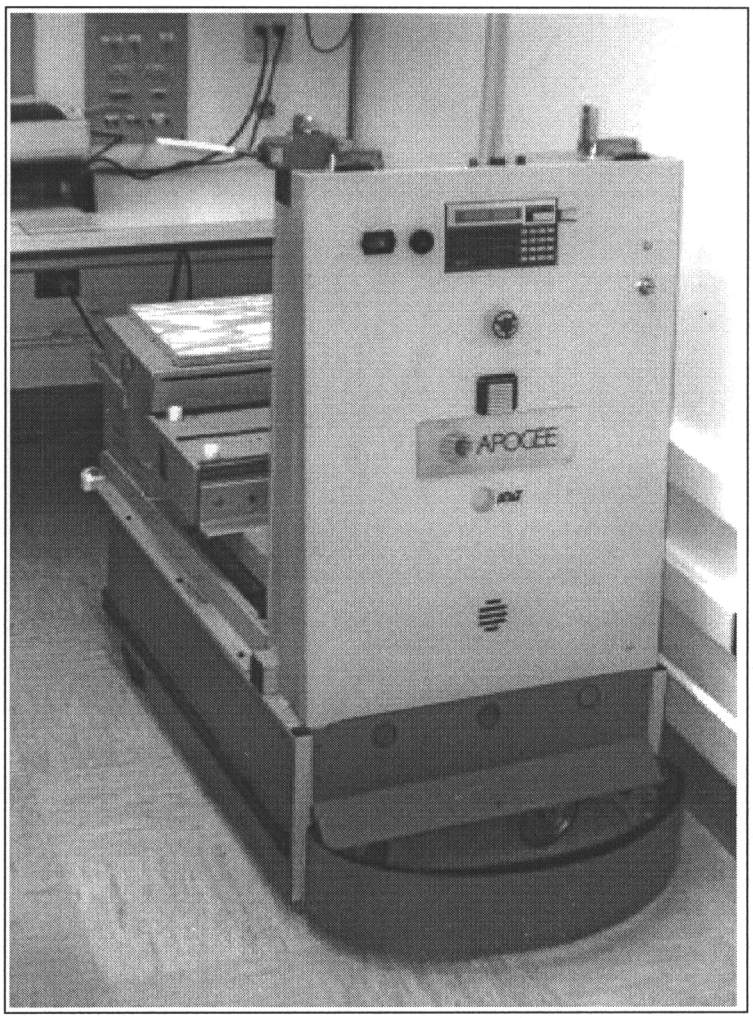


Figure 2 The Orbitor

2.3 Guide Path and Bar Codes

The floor bar codes can be used to mark certain predetermined points along the guide path where the Orbitor is to perform one of many possible

functions. These functions include: identify home locations, identify P/D station locations, activate the voice module, ring the warning beep, decrease or increase the speed of the Orbitor, select turning direction, activate the sonar, and deactivate the sonar. The bar codes must be located to the right of the guide path and by a distance of three inches. They must be on a straight section of the path at least twelve inches from any curve. Two consecutive bar codes must be separated by at least twelve inches.

The Orbitor has two optical sensors mounted vertically on a board in the middle of the vehicle, one scans the guide path and the other scans the floor bar codes. Both of the sensors were adjusted three quarters of an inch above the floor and properly tuned for the floor condition of the hall ways in Whittemore Hall. Further adjustment, if necessary, can be performed according to the Orbitor Manual which will be introduced later in this chapter.

The guide path of the Orbitor was formed by 5/8 inch wide retro-reflective tape placed on the floor. The bar codes were applied with the same tape and positioned next to the path to provide information on P/D station locations, turning decision, vehicle speed, and other instructions. The minimum distance between the edge of the guide path and walls or other objects is fifteen inches when the Orbitor is traveling in a straight line. When the vehicle is to make a turn, it is possible for the front sonar to detect the wall it is approaching and not complete the turn. With the sonar on, a 30 inch clearance must exist between the curve and the wall. If sonar is turned off, the clearance can be reduced to 22 inches. Once the vehicle is traveling in a straight line, only 15 inches of clearance is necessary. The minimum turning radius of the vehicle is 30 inches. Figure 3 illustrate a 90 degree turn with sonar on. The minimum U turn width with the sonar off is 97 inches. Guide paths can be crossed each other only at 90 degree

angles. The Orbitor does not have the ability to be programmed to move in reverse, so the guide path design must avoid dead-end segments. The overall separation between the diverging paths at a branch needs to be one inch or less in the first twelve inches of the branch. It is recommended that the predetermined guide route be marked precisely on the floor before the application of the guide path retro-reflective tape to the floor.

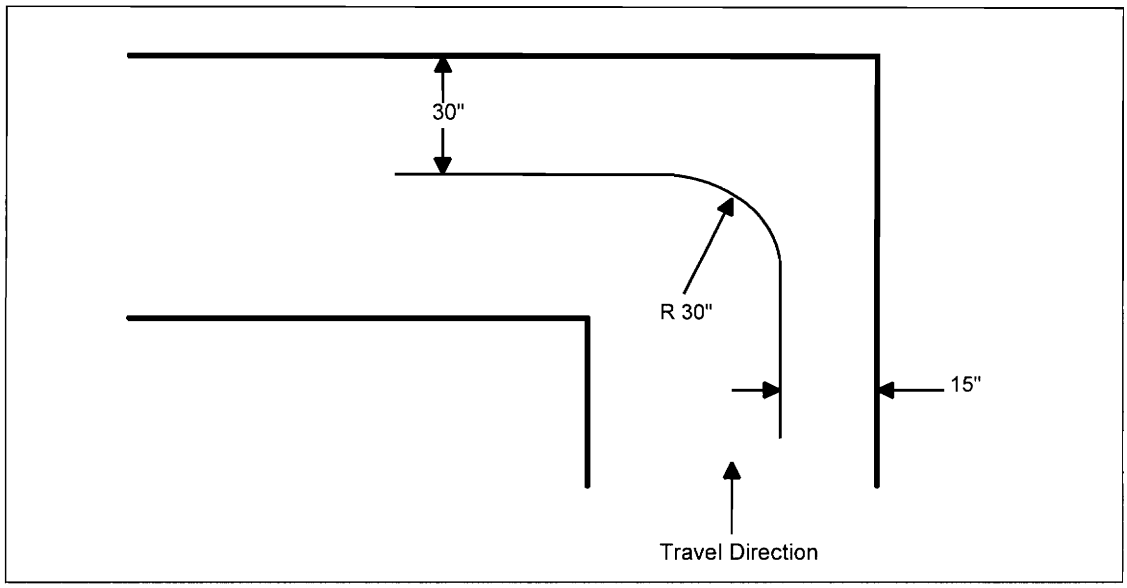


Figure 3 The Guide Path Installation With the Sonar On

The floor bar code reader on the Orbitor is configured to read an eight-bar binary code with six data bars, a start bar, and an end bar. The total combination of possible codes is sixty four. The bar codes on tote boxes and the bar codes that are applied along the guide path share the same geometric pattern but differ in sizes. Floor bar codes need to be positioned precisely with respect to the guide path. There is a template available for making precise bar codes (see Appendix D). A typical bar code example and its relative position to the guide path is shown in the Figure 4.

The bar codes on the tote box are formed with the same geometric pattern. The size of the bar codes for the tote box is two by three and a quarter inches. They need to be applied on the side of the tote box that is facing the front of the Orbitor. The codes are located two inches from the edge and three eighth of an inch above the bottom of the box. There is a more detailed description and ready to copy template of the tote box bar code in the Orbitor Operation and Maintenance Manual.

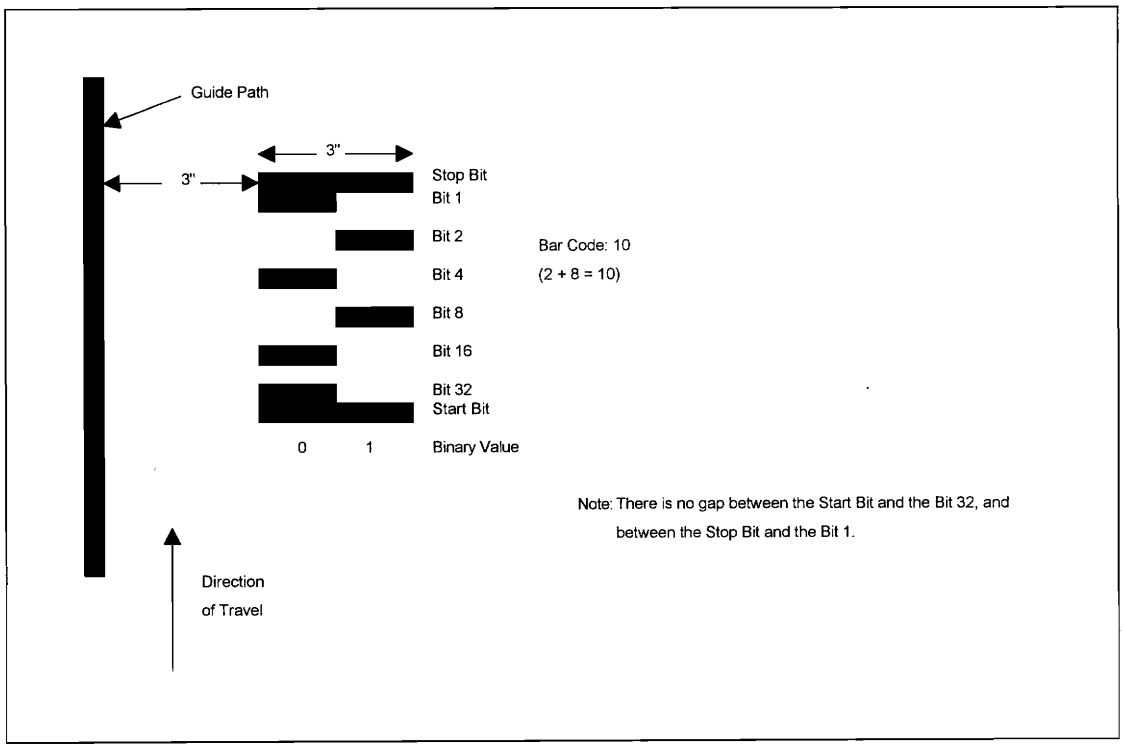


Figure 4 The Floor Bar Codes

2.4 Control Systems Components

The Orbitor has three primary subsystems that are interconnected by an HP communications loop (HP-IL) as shown in Figure 5. The HP-IL communications loop consists of the HP-71B computer (as the controller of the HP-IL), the Motorola 6809 microprocessor (as Device 1), and the Intersection and Communication Unit (ICU) (as Device 2). Device numbers need to be specified in programming the HP-71B computer. All message travel is in one direction on the HP-IL. The controller, HP-71B, is responsible for issuing instructions to other components in the HP-IL. There can be only one controller in an HP-IL at one time. All the devices, include the controller of the HP-IL, can be either a talker or a listener depending on the message routing. There can be only one talker and one listener at the same time in an HP-IL system.

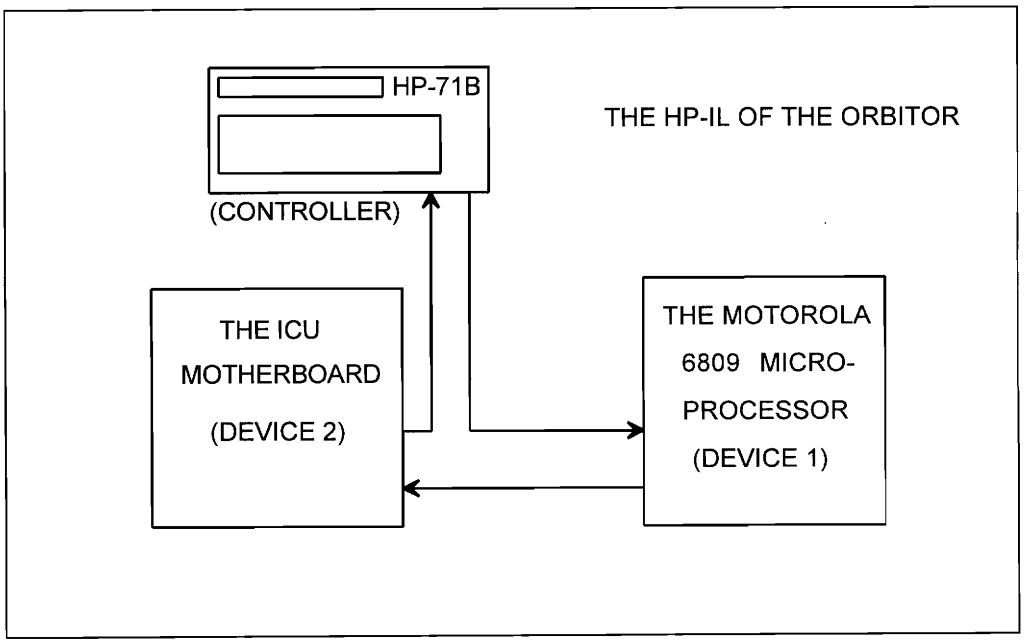


Figure 5 The Overview of the HP-IL Communications Loop

The HP-71B computer is the master controller of the Orbitor. User programs, written in HP BASIC, can command the vehicle to perform desired tasks. Additionally, the HP-71B computer can communicate with a remote computer through the on-board ICU. This feature allows the vehicle to adapt to changing conditions in the production system in which it is employed. The Motorola 6809 microprocessor is the actual vehicle controller. It carries out high speed, low level tasks in response to commands from the HP-71B computer. It also responds to inputs from the vehicle sensors and the floor bar code reader. Details of these concepts are presented in the next chapter.

The Intersection and Communications Unit (ICU) is an on-board interface that collects communication data and information either from the on-board HP-71B computer or from the remote computer. In addition to the ability of communicating with a remote computer, the on-board ICU can also be programmed through the HP-71B computer to perform vehicle to vehicle communications when multiple AGVs are employed in a system. The components within the ICU include an ACE board, a modem, and an ICU motherboard. They are shown in the Figure 6.

The HP-IL/RS232 interface links the HP-71B computer to the ICU. The ACE board is a high speed control card that uses a Motorola 68HC11 processor to handle incoming and outgoing messages between the HP-71B and the modem. The Modem is used for short range wireless communication that applies Frequency Modulation (FM).

The remote computer communicates with the Orbitor through a floor mounted Automatic Dispatching Unit (ADU). The primary function of this unit is to

convert the serial communications of the remote computer into radio frequency communications in order to communicate with the vehicle mounted ICU.

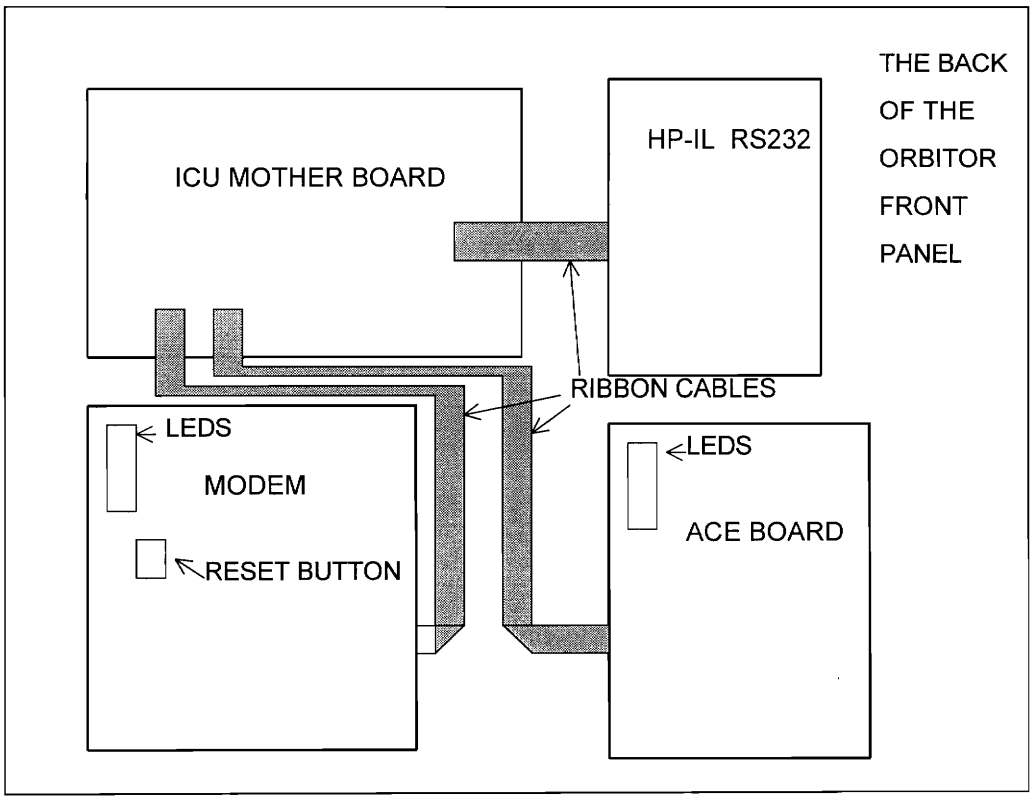


Figure 6 The Components of the ICU

The elements within the ADU are similar to the ones in the ICU. The ACE board and the Modem are vertically inserted in the ADU motherboard in a metal box. Figure 7 has shown a picture of the inside of the ADU. The power switch is located at the lower right corner. The RS-232 / RS-422 selector switch is at the top left corner. The communication cable between the serial port of the remote computer and the ADU is connected at the lower left corner of the ADU motherboard. The antenna is connected at the middle bottom of the ADU

motherboard. The height of the antenna of the ADU box should be approximately the same as the height of the antenna of the Orbitor.

The distance limitations for the radio frequency communication between an ADU and an ICU is approximately 9 feet. Hence, multiple ADUs must be employed if a remote computer is going to communicate with the vehicle in different place along the path. A single computer can be tied to multiple ADUs, if desired. Therefore, it is only necessary to have one computer to communicate with multiple ADUs along the guide path. Each ADU to computer connection must be made as a point-to-point connection. No provisions are made for multidrop communications.

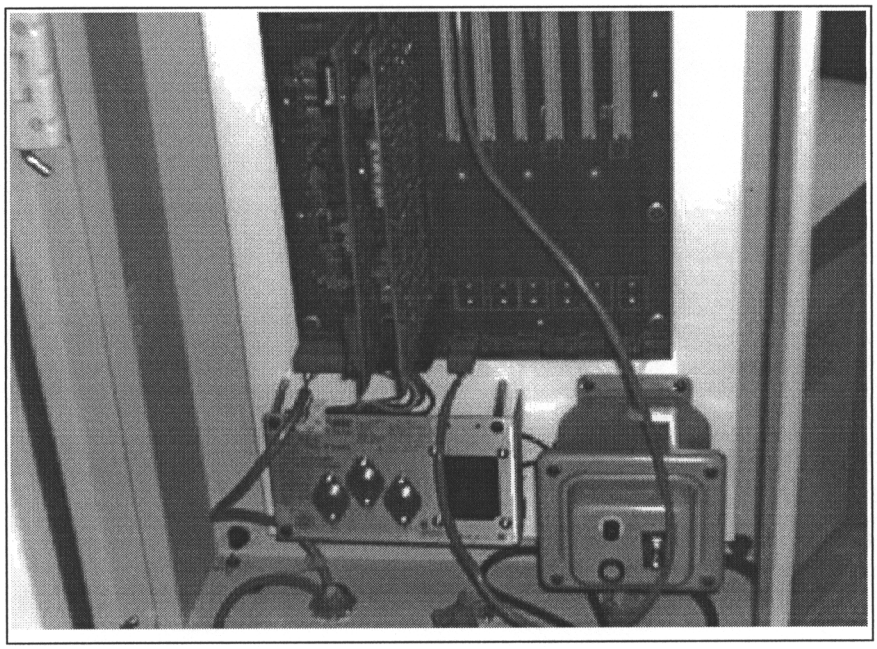


Figure 7 The Components of the ADU

The remote computer can be any computer capable of generating an RS-232 or RS-422 signal. An IBM 7552 Industrial Computer was used in this

project. The RS-232 cable was connected from the COM1 port of the computer to the ADU. The configuration of the RS-232 cable between the remote computer and the ADU is shown in the Figure 8.

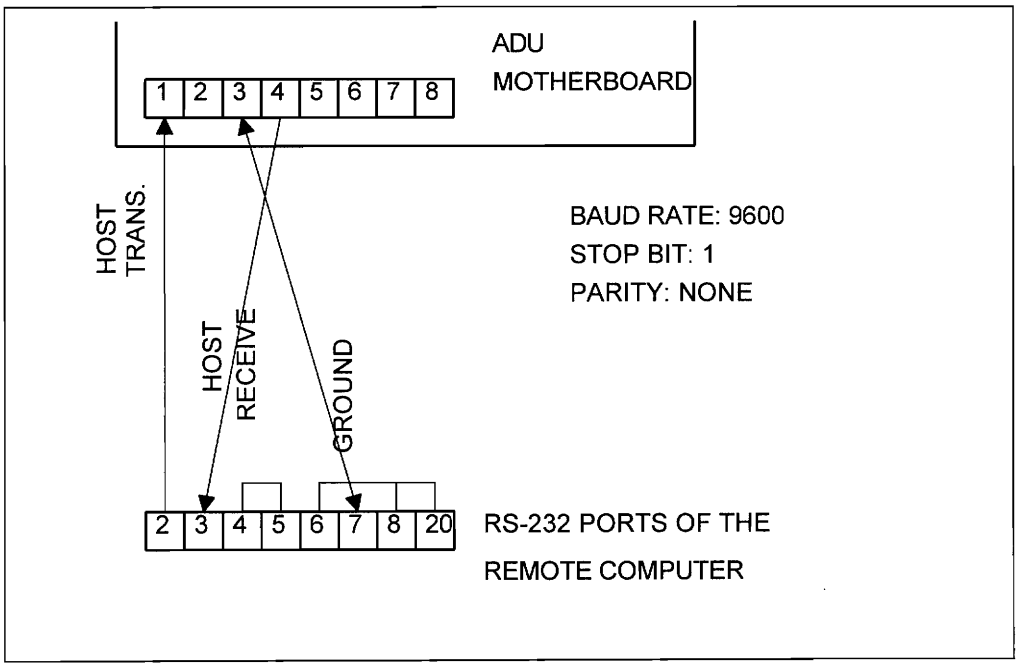


Figure 8 Configuration of the RS-232 Cable

2.5 Available Reference Materials of the AGV

All the concepts, descriptions, and demonstrations about the Apogee Orbitor in this work are derived from the combinations of the following documents. Copies of documents are available in the Robotics and Automation Laboratory. The available reference materials of the Orbitor and brief summarization of them are listed below.

Apogee "Orbitor" Automatic Guided Vehicle System Owner's Manual (Ref. Apogee #510-0009, 1987). This document is really a collection of manuals. It contains an overview of the Orbitor, the installation and operating procedures, the maintenance and troubleshooting, the ADU Operator's Instruction Manual, the ICU Operator's Instruction Manual, the Floor Board Adjustment Procedure, the Electronic Repair on Static Sensitive Devices in the AGV, and the Speech Processor Manual. The protocols for remote to on-board computer communications are in the ADU and the ICU manuals of this book. Some of the material contented in this manual overlap with the manual that will be described next.

Orbitor Automatic Guided Vehicle System Operation and Maintenance Manual (Preliminary Copy, Apogee Robotics, Inc., 1990). As the manual discussed above, this manual provided information about the installation, the features of the Orbitor, the maintenance, the trouble shooting, and the repair of the Orbitor. In addition to the overlapped information with the previous manual, this manual provided many useful example programs for the HP-71B computer.

HP 82401A HP-IL Interface Owner's Manual for the HP-71 (HP #82401-90001, January, 1985). This manual gives general information about the overviews, operations, and connections of the HP-IL. Almost all the keywords that are associated with the HP-IL for the HP-71B are listed and explained in this manual as well.

HP-71 Owner's Manual (HP #00071-90001 Rev. E, March 1987). This manual demonstrated the basic operations and the programming of the HP-71B computer. This manual does not illustrate any Orbitor related activities since it is published by Hewlett Packard. However, it is highly recommended that all the

Orbitor programmer read this manual before touching the HP-71B computer. This manual can help Orbitor users in understanding how to handle those important files that were saved in the HP-71B computer. The programming environment and the syntax of the BASIC language in the HP-71B computer are explained in detail in this manual.

HP-71 Reference Manual (HP #00071-90010, October 1987). This manual has a lot of overlapped information with the HP 82401A HP-IL Interface Owner's Manual for the HP-71. Some of the necessary keywords for operating the Orbitor through the HP-71B computer are provided only in this manual. As the previous manual discussed above, this one does not provide anything associated with the Orbitor. All of the HP-71 related keywords, error and system messages, warnings, and some Orbitor not related mathematical discussions are explained in this book.

HP-71 BASIC Made Easy (by Joseph Horn, 1985, Synthetix). This book explains how to program the HP-71B computer in a more friendly fashion. It was not written to replace the HP-71 owner's manual but to complement it. The HP-71B users will find this manual provides practical examples that are easier to comprehend. Nevertheless, this one does not say anything about the Orbitor either.

Three blue prints of the Orbitor are available for those who are interested in the electronic schematic of the Orbitor. The titles of the prints are: AGV CHASSIS W/LIFT, VEHICLE SCHEMATIC WITH SONNENSCHEIN BATT., and AGV-LIFT.

CHAPTER THREE
FUNDAMENTAL CONTROL SYSTEM CONCEPTS

3.1 The AGV System Control Overview

At the base hardware level, the Motorola 6809 microprocessor serves as the controller of the vehicle. It receives inputs from the vehicle sensors and provides drive signals to the vehicle actuators. It provides real-time, dynamic control of all vehicle functions. It is not "programmed" by the user in the sense of a programming language. Rather, it response to commands given to it by the HP-71B computer. For example, the HP-71B computer might direct the Motorola microprocessor to "enable the sonar" detectors. From that point on, the Motorola 6809 microprocessor would begin sensing any obstacles that might be found in the vehicle path and act appropriately. Additionally, the Motorola 6809 microprocessor responds to information encoded into the bar codes that are read by the floor bar code reader. For example, a bar code might be used to change the speed of the vehicle or to make it stop at the next Pickup/Delivery (P/D) station. Details of the Motorola 6809 microprocessor operations are given in this chapter.

The HP-71B computer serves as a higher level, user programmable controller. Information obtained from the Motorola 6809 microprocessor is interpreted by the user written logic and results in commands to the Motorola 6809 microprocessor to perform desired tasks. For example, the user written HP BASIC program in the HP-71B computer could first ask the Motorola 6809 microprocessor what bar code was found on the tote box. The BASIC program logic could then determine a desired path and direct the vehicle to a desired location. Additionally, the HP-71B program can monitor vehicle parameters,

(such as battery voltages, motor currents, etc.) and take appropriate actions. Finally the HP-71B computer can communicate with a remote computer for system control type behavior. Programming of the HP-71B computer is addressed in Chapter 4.

A remote computer can communicate with the HP-71B computer. As explained earlier, the remote computer is connected to a floor mounted ADU which communicates with the onboard ICU via radio frequency. (The ICU connects to the HP-IL communications loop.) While the structure of this communication link is specified, the internal contents of the messages is complete open to user specifications. Remote computer control can be used to change vehicle routing, provide alternative destinations, etc. Details of this communication structure and remote computer operations are given in Chapter 5.

3.2 The Operations of the AGV

There are three different operating modes of the Orbitor. First, it can be manually directed to go in forward or in reverse, to raise or lower the lift assembly, and to extend or retract the arm of the lift assembly by using the Manual Control Pendant. Second, the Orbitor can perform tasks along the guide paths without any interactions with the remote computer. This mode involves interaction between the Motorola 6809 microprocessor and the HP-71B computer. Third, the Orbitor can be integrated with a remote computer for receiving commands and providing status feedback.

The Manual Control Pendant provides easy access of some of the features of the Orbitor and has already been discussed. It is a good practice to turn the

HP-71B off when operating the Manual Control Pendant. The most important function of this device is that it is the only way of moving the Orbitor in reverse besides pushing the Orbitor manually.

The HP-71B on-board computer provides intelligent control of the Orbitor when it is not associated with a remote computer. It is programmed through user input via the keyboard of the HP-71B. The HP BASIC programming language is used to program the HP-71B. In the HP-71B computer there is a TEST program that is available for diagnostic testing of the basic functions of the Orbitor. Multiple programs can be stored in the HP-71B and a program can be executed by typing "RUN filename" and then press the [ENDLINE] key. The programming of the HP-71B will be discussed more detail in the Chapter 4.

The remote computer can interactively communicate with the on-board HP-71B computer and assign tasks accordingly. This requires appropriate software on both the HP-71B and the remote computer. This setup was tested and the programs are discussed in Chapter 6. The program illustrates the bi-directional interface of this communication. The interconnection of the wiring of the remote computer was discussed earlier.

3.3 Vehicle Control Computer Concepts

The Motorola 6809 microprocessor serves as the base level, vehicle controller. It is not user programmable. It receives basic sensor inputs and activates vehicle actuators appropriately. Additionally, it receives input information from floor mounted bar codes and commands from the HP-71B computer which modify its operations.

3.3.1 Bar Code Input Information

When the bar code reader reads a floor mounted bar code, a coded value between 0 and 63 is obtained. This value is used by the Motorola 6809 microprocessor to access a specific character in an internal array to determine what function should be performed. The internal array is referred to as a W string and has 64 character positions. The character positions are numbered 0 through 63.

Table 1 W String Character Codes

Characters	Functions
C	Slow down to speed 4 and stop at next station
D	Slow down, move 12 inches, then stop
E	Sonar enable
F	Sonar disable
G	Continue moving with no changes
L	Turn left at all diverging intersections
N	Ring the chime
R	Turn right at all diverging intersections
S	Stop immediately
4-9	Go at speed 4, or speed 5, ..., or speed 9
0	Go at speed 10

When the vehicle reads a bar code, the specified W string character indicated by the bar code number is then interpreted and its function performed. Table 1 identifies valid W string functions. The C, D, and S character codes cause the vehicle to stop. Vehicle speed is controlled by the characters 0, and 4 through 9. The G character is used as a filler when no specific function is required at that point for the current W string. Characters L and R specify the

turning or branching mode. (The vehicle is always in either a left or right branch mode.) Characters E and F enable and disable sonar and a character code of N causes the chime to ring.

The W sting in the Motorola 6809 microprocessor can be dynamically changed by the user written program in the HP-71B computer. (This is explained shortly.) Hence, as the W string is modified, the character in a given position can change and thus the function performed at a specific bar code location can also change. The bar codes only give the position index. The current W string must be accessed to interpret the actual character code.

To help solidify the idea of vehicle control with W strings, consider a case when the current W string is

"GDNNLNNNGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG".

Note that there are exactly 64 characters in the string. The W string always has 64 characters, no more and no less. The bar code value (0-63) will be used to "index" the specific position in the string. Note also that the string has a lot of "G" characters. The "continue" code is used when no specific action is required. Given the string noted above, when the Orbitor reads a floor bar code with a value of 1, it will slow down and move about 12 inches then stop. If the Orbitor reads a floor bar code with a value of 4, it will make a left turn at all of the diverging intersections. It will continuous going when encountered floor bar code values of 0, 8, and any values greater than 8. The Orbitor will beep after it reads a floor bar code of 2, 3, 5, 6, or 7 according to the example W string above. The programmer of the Orbitor needs to match what the Orbitor will do after passing a

specific floor bar code with the position number of the command characters in the string. The HP-71B can change the control of the W string whenever needed.

3.3.2 HP-71B Command Codes

The HP-71B computer can send commands to the vehicle controller. Some of these commands cause specific functions to be performed. Others request information from the Motorola 6809 microprocessor. Details regarding programming the HP-71B computer are presented in the next chapter. Table 2 outlines the 15 basic types of command.

Table 2 HP-71B Orbitor Command Codes

Codes	Functions
A	Voltage Measurements
B	Button Lights
C	Charger Control
E	Lift Elevate
G	Speed Control
H	Tote Check
L	Button Enable
N	Chime
Q	Query/Status
R	Lift In/Out
S	Sonar Control
T	Talk
W	Floor Code Mapping
Z	Zero Lift
0	(Digit Zero) Speak Number

The A commands request voltage data from the Motorola 6809 microprocessor. There are sixteen different voltage commands (two of which are not really used). When the HP-71B computer issues a voltage command, the Motorola 6809 microprocessor responds with the appropriate value. The A commands are given in Table 3.

Table 3 A Command Codes

Codes	Related Voltage Information
A0	Ground offset voltage
A1	Battery voltage (fused)
A2	E-Stop voltage
A3	System voltage on power board
A4	Lift motor current
A5	Right motor current
A6	Left motor current
A7	Not used
A8	Left bumper voltage
A9	Right bumper voltage
A10	Center bumper voltage
A11	Not used
A12	Floor scanner PCB +5V/2
A13	Lift PCB +5V/2
A14	Sonar PCB +5V/2
A15	Digital control PCB +5V/2

The B commands control the red and green buttons on the top front surface of the Orbitor. A B0 turns the light off while a B1 turns the lights on. The lights must be on and enabled to be used for operator input. The Q command can be used for query the vehicle for operator input and will be explained shortly.

The C commands control the vehicle battery charging system. A C1 turns the charger relay on and a C0 turns the charger relay off. Since the C commands refer to charging the Orbitor by activate or de-activate the floor charging strips of the vehicle, the Orbitor must be directed to a floor charging station for these function to work properly. A C0 command must be issued upon moving the vehicle off the charging station at the end of a charging operation in order to avoid any electrical safety hazards.

The E command causes the Motorola 6809 microprocessor to move the lift to a specified distance above the floor. The desired elevation is specified as an argument to the command and is given in pulses. There are 78 pulses per inch. To move the lift to an elevation of 30 inches, the argument would be specified as 30 x 78 or 2340 pulses. The Z command must be executed to initialize the lift before any E commands can be given. This command is discussed below.

The G commands control the speed of the vehicle. A G0 will stop the vehicle and disable the red and green operator lights. A G1 command causes the vehicle to resume travel at the previously specified speed. Commands G2 through G10 are specific speed commands. A speed of G10 represents a speed of about 120 feet per minute.

The H command code causes the Orbitor to check the lift for the presence of a tote and return a value of H0 or H1. An H0 indicates that there is no tote present on the vehicle. A code of H1 indicates there is a tote present.

The L command codes enable/disable the red and green operator interface buttons addressed earlier when the B command codes were discussed. An L0

disables both buttons while an L1 enables the buttons. The Q command (query) is used to see if an operator has pressed either of the buttons.

An N command code causes the vehicle to activate a chime or a short beep of less than one second duration.

Table 4 Vehicle Controller Query/Status Response Codes

Codes	Meanings
I00000	Orbitor has stopped normally
I00004	Tachometer pulse not present (AGV won't move)
I00005	Guide path has been lost
I00006	Bumper been pressed. Use G1 to resume
I00007	Sonar obstruction. Use G1 to resume
I00008	Orbitor is moving forward normally
Fnnnnn	Floor code number nnnnn has been read
Tnnnnn	Tote box code number nnnnn has been read
L00001	Green button was pressed
L00002	Red button was pressed

The Q command code request information from the Motorola 6809 microprocessor. The response will be one of the strings listed in Table 4. The vehicle controller will send the response associated with the last change of state of the vehicle. For example, if the vehicle has come to a normal stop and the HP-71B sends a Q command to the Motorola 6809 microprocessor, it will respond with an I00000 code. If the vehicle has stopped because of an obstruction in its path, an I00007 code would be returned. Note that the Q command can be used to detect an operator input on a green or red button immediately after it is pressed.

The R command codes cause the lift arm to be extended or retracted. (Remember the lift arm only extends to one side of the vehicle.) The RE command extends the lift arm while the RR command causes the arm to retract. After completing the lift movement, the Motorola 6809 microprocessor returns an R1 code indicating successful completion of the operation or an R0 code indicating an operational failure.

The S command codes are associated with the Orbitor sonar as shown in Table 5. The SE and SD codes enable and disable sonar respectively. The SL, SS, SR, and ST commands are used to measure distances to obstructions. The Motorola 6809 microprocessor returns a value in milliseconds that can be converted to an approximate distance in inches by multiplying the value by 0.66.

Table 5 S Command Codes

Codes	Meanings
SE	Enable the sonars
SD	Disable the sonars
SL	Measure distance with the left sonar
SS	Measure distance with the center sonar
SR	Measure distance with the right sonar
ST	Measure distance with the lift arm sonar

The T command code activates the voice synthesizer of the Orbitor. The voice synthesizer of the Orbitor can speak according to a specified allophone variable string. A "Z" character is needed at the end of this string to stop the voice synthesizer. Detailed description of how to assembly an allophone string is provided in the Apogee "Orbitor" Automatic Guided Vehicle System Owner's Manual.

The concept and use of W strings by the vehicle controller was introduced in Section 3.3.1. The W command code is used to pass the W string from the HP-71B computer to the vehicle controller. The new string is passed as an argument to the W command. This will be further discussed in the next chapter.

The Z command code will cause the lift to be lowered to its lowest position. An argument is also specified with the command which specifies the number of pulses associated with elevation. Since the elevation of the lift above the floor is approximately 21.75 inches, this argument should be 21.75 x 78 or 1596.50. The vehicle controller returns a Z1 or a Z0 code representing successful completion or abnormal termination of the operation respectively.

The O (digit zero) command code causes the vehicle to "speak" digits. The string of digits to be spoken is passed as an argument to the command. The vehicle does not attempt to give the value of the string, rather individual digits are "spoken". Hence, "12" would be spoken as "1", "2". This feature permits easy checking of the values being read by bar code readers.

CHAPTER FOUR
THE BASIC AGV PROGRAMMING

4.1 The On-board HP-71B Computer of the AGV

The on-board HP-71B computer is used to store and execute user written programs that provide high level control of the Orbitor. The HP-71B computer is programmed in the BASIC programming language. This computer is located on the front panel of the Orbitor above the emergency stop button. Programs for the HP-71B computer must be key in directly on the keypad of the computer which is mounted on the front panel of the vehicle. The keypad of the HP-71B computer contains two groups of keys. The alphabetic keys are located on the left side of the computer and the numeric keys are located on the right. There are special programmable functions printed in either blue or gold color on or above the keys. These functions can be activated by pressing the blue "g" or the gold "f" key first, and then pressing the desired key next. All the alphabetic keys are laid out the same way as a standard typewriter.

The HP-71B can be turned on by pressing the ON key at the lower left corner. The currently activated password option feature of the HP-71B requires users to enter the password upon turning the HP-71B on. If a wrong password is entered, the HP-71B will turn itself OFF at once.

The display of the HP-71B computer holds a maximum of twenty-two characters at one time. The total length of a line in HP-71B programs can be as long as ninety-six characters. The [ENDLINE] key, located near the middle of the keyboard layout, performs the similar function as the Enter key of a regular

computer. A line of commands will not be executed without pressing the [ENDLINE] key at the end of the line.

The HP-71B computer draws power from the Orbitor batteries when it is attached to the Orbitor and the battery cable of the Orbitor is properly connected to the batteries. There are also four AAA size batteries that are located in the back of the HP-71B computer. The existence of the internal batteries of the HP-71B is necessary to provide the power that is used to hold the written programs in the computer. If the HP-71B is detached from the Orbitor, regular battery replacements of the HP-71B is necessary. No keys should be touched on the HP-71B keypad when replacing the batteries.

The HP-71B has sixty four kilobytes of memory. There are four available memory ports located at the back of the HP-71B below the alpha keypad. Each port can be filled with a memory module that can hold an additional sixty-four kilobytes of memory. There is a module in port zero at the present time. The memory management of the HP-71B is explained in detail in the HP-71B Owner's Manual. The user can type in MEM [ENDLINE] to check available memories in the HP-71B. The keyboard of the HP-71B can be re-defined by the user and the definition of the keys can be saved in a special memory location. It was not necessary to modify the memory assignment of the Orbitor since it was properly configured previously at the AT&T plant before the vehicle was donated to the ISE Department.

The HP-71B communicates with the vehicle controller and the remote computer through the HP-IL communication link. The HP-71B is the controller of the HP-IL communication loop and it will always recognize the vehicle controller as the Device 1 and the ICU as the Device 2 as Figure 5 illustrated.

4.2 The Editing Environment of the HP-71B

The editor of the HP-71B computer contains many keywords that can be used in creating, editing, and executing programs and in viewing directories, as well as many other necessary editing functions. Writing and running programs in the HP-71B are discussed in more detail in the HP-71 Owner's Manual. However, since the manual is a general purpose manual related to the HP-71B computer, there are no references that deal directly with the Orbitor. Examples that are found in subsequent chapters of this document will prove helpful in trying to program the HP-71B computer to interact in the AGV system.

Commands that are capitalized represent keywords of the language and must be entered as they appear. Words in Italics must be user supplied inputs. Words that are in square brackets represent specific keys on the keypad of the HP-71B computer. For example, [ENDLINE] means to press the [ENDLINE] key at the end of a statement. A statement in a program will be saved by pressing the [ENDLINE] key at any position in the statement. Line numbers are required for every line of the program and designate the sequence of the program. Duplicated line numbers are not allowed. The syntax of every statement in programming the HP-71B is automatically checked upon pressing the [ENDLINE] key.

The keywords for programming the Orbitor in the BASIC language will be listed and explained below:

- EDIT *filename* [ENDLINE] -- to edit an existing file or to create a new file.
- CAT ALL [ENDLINE] -- to review the directories of the HP-71B computer.

- LIST *line#* [ENDLINE] -- to list line *line#* of the current file for checking but not for editing.
- FETCH *line#* [ENDLINE] -- to list line *line#* of the current file for editing.
- PURGE *filename* [ENDLINE] -- to delete a file. A deleted file in the HP-71B can not be undeleted. The only way to recover a deleted file is to type it back in.
- DELETE *filename* [ENDLINE] -- to erase all the contents of the *filename* but the file will be a blank file and will still be listed in the category without any memory occupied.
- RUN *filename* [ENDLINE] -- to execute a file.
- TRACE FLOW [ENDLINE] -- to turn on the trace function.
- TRACE OFF [ENDLINE] -- to cancel the TRACE condition.
- RENUMBER *x, y, a, b* [ENDLINE] -- to renumber lines *a* through *b* so that line *a* becomes line *x* and succeeding lines are numbered in increments of *y*.
- DELAY *x* [ENDLINE] -- to display each line for *x* seconds when executing a program.
- CONTRAST *x* [ENDLINE] -- to change the contrast of the display. Default value is nine. The range of *x* is zero to fifteen.
- COPY *filename1* TO *filename2* [ENDLINE] -- to copy *filename1* to *filename2*. The contents of *filename1* will not be changed.
- RENAME *filename1* TO *filename2* [ENDLINE] -- to rename *filename1* as *filename2*.
- MERGE *filename, start line#1, final line#2* [ENDLINE] -- to copy from line#1 to line #2 of the *filename* into the current file. The line numbers will be copied over unchanged. The original file, *filename*, will not be changed.

- [f] [-char] -- to delete a character.
- [f] [-line] -- to delete the rest of a line. If the cursor is positioned between the line number and the first character of the line then execute this command with [ENDLINE], the line will be erased from the current program.
- [f] [i/r] -- to change the type mode to the insert mode, or vise versa, while editing a program.
- [f][cont], or press the [on] key and then press [f][cont] -- to continue running the previously interrupted program.
- [on] [on] -- to reset the HP-IL at the HP-71B.

If no file name was specified and there was no file currently active in the HP-71B, the default "workfile" can be reached by pressing EDIT [ENDLINE]. The "workfile" is a file that allow users to practice programming the HP-71B. This "workfile" will not be purged but can be deleted. If a variable command is not keyed in correct, the HP-71B will voluntarily insert the DISP command in front of the variable command after the [ENDLINE] was pressed.

4.3 General HP-71B BASIC Programming

All variables in the HP-71B BASIC can be defined as REAL (12-digit), or SHORT (5-digit), or INTEGER (from -99999 to +99999). The default variable value is an INTEGER zero. Arrays are defined with a letter and an optional digit; no multiple letters nor multiple digits are allowed in defining a name for an array, e.g. P1(4). The number of elements in the array is given in parentheses. Matrixes are defined in a similar fashion, such as M(2, 3), or REAL M(2, 3). The "REAL" positioned in front of the name of a matrix is necessary to create a real

precision matrix. The default lower bound of an array is zero. This is also true on the dimensions of a matrix. However, the lower bound of arrays can be set by using OPTION BASE command. For example, if it is desired to start the index values at one, the following statement can be used:

```
50 OPTION BASE 1
```

String variables are formed with the name of the string followed by an optional dimension inside parentheses, and then the string size inside square brackets. Some examples of such string variable are as follows: N$="Hello", DIM U$[64], and DIM D$(6)[64]. The DIM D$(6)[64] sets up memory for 6 strings containing up to 64 characters each. A variable reference, R$(J)[2,12], references the characters that are in the Jth string of R$ starting at the 2nd character and stopping at the 12th character. For example, the follow line could be used to pass a substring to the CHKSUM subroutine:

```
610 CALL CHKSUM (R$(J)[2,12], K$).
```

Multiple strings can be connected by using the "&" character, e.g. "abc"&"123" will yield "abc123". A string, A$, that contains numeric characters in it can be converted to number by using the VAL(A$). A number, x, can be converted to a string by using the STR$(x).

The loop in the HP-71B BASIC can be constructed as:

```
10 FOR X = 1 TO 10 @ BEEP @ DISP x @ NEXT X
```

The command: DISP "XXX" will displays XXX on the HP-71B screen. The CHR$(x) changes the number x in the ASCII table to a single character string, e.g. CHR$(10) is the LF, one character, Line Feed.

A label in the HP-71B BASIC can contain up to eight letters or digits. It must begin with a letter and be enclosed in single quotation marks. Where a label identifies a statement or statements, a colon must be placed after the last quotation mark. Where a label references a statement, no colon is needed. For example:

```
500 'START':
510 .....
520 .....
530 IF NOT S THEN 'START'
```

The remarks in a program are notes by using the "!" symbol in any position of a line of statement. The rest of the statement after the "!" symbol will not be read by the HP-71B computer.

There are few special command lines that should appear at the beginning of most programs:

```
10 RESET HPIL
20 DESTROY ALL
30 ENDLINE
...
100 END
```

The RESET HPIL command states that the HP-71B computer is the controller of the HP-IL and resets the status of the interface loop. The DESTROY ALL erases all the variables and arrays from a previous run and reclaims previously consumed memory in the HP-71B computer. The ENDLINE statement will automatically restore a carriage return and a line feed (end of line terminators) at the end of a statement that contains "variable commands". This will be discussed below.

4.4 HP-71B Interaction With the Vehicle Controller

The HP-71B computer communicates with the Orbitor controller via messages that contain command codes as described in section 3.3.2. These command codes cause the Orbitor to perform specific tasks or request status information. A message can be sent to the vehicle controller via the HP-IL by using the OUTPUT command, and a return message can be received from the vehicle controller by using the ENTER command. For example:

```
100 OUTPUT : 1;"Q"
110 ENTER : 1; A$
120 DISP "STATUS IS ";A$
```

The number "1" in the OUTPUT and ENTER commands specifies the device number addressed on the HP-IL. The letter "Q" in the OUTPUT command is a coded command for the vehicle controller to send its status to the HP-71B computer. A list of all HP-71B command codes was given in Table 2. The A$ string variable is an arbitrary string used for temporarily saving the message. There has to be a space between the keyword "OUTPUT" and the following

colon. The device number, 1 or 2, has to be placed between the colon and a semicolon. The DISP command will display whatever is enclosed between double quotation marks and the content, or the value, of the following string (A$), or a variable.

Some command code are fixed in length. Examples of this type of command code are B0 and B1, L0 and L1, N, etc. Other commands have variable length. Examples of this type of command code include A and G command codes. These variable length command codes necessitate the use of the ENDLINE BASIC statement at the beginning of the program as stated earlier. Finally, some command codes are followed by an argument to pass additional information to the vehicle controller. Examples of this type of code are the E, T, and W command codes. Additionally, the 0 (zero) command code falls in this category.

Some command codes require no response from the vehicle controller. This includes codes such as N for a beep. Other codes have an optional 0/1 response to signal abnormal/successful completion of the requested task. For example, the Z command from the HP-71B computer will evoke a Z0 or a Z1 response from the vehicle controller upon completion of the initialization of the lift. Finally, some command codes solicit more elaborate responses. The Q command code is the best example of this. Additional examples include A and S codes. Responses that go unread are lost after the next response is generated.

Table 6 presents the HP-71B command codes with this new information. Some example code segments are given in Table 7.

Table 6 HP-71B Command Code Details

Codes	Detail Functions
A	Voltage measurement, variable length, no argument, string response needed for proper operation.
B	Button light control, fixed length (B0 or B1), no argument, no response given by vehicle controller.
C	Charger control, fixed length (C0 or C1), no argument, no response given by vehicle controller.
E	Lift elevate, variable length, argument needed, E0 or E1 response given by vehicle controller.
G	Speed control, variable length (G0, G1, ..., G10), no argument, no response needed from vehicle controller.
H	Tote check, fixed length, no argument, H0 or H1 response given by vehicle controller.
L	Buttons enable, fixed length (L0 or L1), no argument, string response needed for proper operation.
N	Ring the chime, fixed length, no argument, no response given by vehicle controller.
Q	Query/Status, fixed length, no argument, string response needed for proper operation.
R	Lift extend or retract, fixed length (RE or RR), no argument, R0 or R1 response given by vehicle controller.
S	Sonar control, fixed length (SE, SD, SL, SS, SR, or ST), no argument, variable response given by vehicle controller.
T	Talk control, variable length, argument needed, voice response from the vehicle controller/voice synthesizer.
W	Floor code mapping, fixed length, argument needed, no response from vehicle controller.
Z	Zero lift, variable length, argument needed, Z0 or Z1 response given by vehicle controller.
0	(Digit Zero) Speak number, variable length, argument needed, voice response from the vehicle controller/voice synthesizer.

Table 7 Example HP-71B Command Codes

Code Segment	Comments
2250 OUTPUT : 1;"A0" 2260 ENTER : 1;A$	These statements request the ground offset voltage of the Orbitor for battery voltage checking. (A subroutine, CHKBAT, is executed after every task that the Orbitor has performed in the example program in the Chapter 6. This subroutine checks two voltages of the Orbitor and will be discussed in more detail later.)
100 OUTPUT :1;"E";INT (36*78)	This statement causes the lift assembly to be raised to a value of 36 inches above the floor.
100 OUTPUT :1;"G5"	This statement causes the vehicle to travel at speed G5 which makes the Orbitor travels about 60 feet per minute.
100 OUTPUT :1;"Z";INT(21.75*78) 110 ENTER : 1;A$ 120 DISP A$	These statements cause the lift arm to lower to its zero position which is 21.75 inches above the floor. The screen of the HP-71B will display either Z1 (successful) or Z0 (failed) after the example has been executed.
100 OUTPUT :1;"RE" 110 ENTER : 1;A$ 120 DISP A$	These statements request the lift arm to extend. After the arm has extended successfully, A$ will contain an R1 for display. If operation failed, A$ will be R0.

One final example needs to be covered to insure understanding of W string operations. This example is given below. Lines 105, 106, and 107 define three characters strings that are possible W strings. Each string is 64 characters long and contains appropriate control character codes. Later in the program, line 1130 actually passes one of the W strings to the vehicle controller. The string that is passed would depend on the value of the index variable J. Since the stings have different control codes in some of the positions, the response of the vehicle to a

given floor bar code value would change depending upon the current W string in the vehicle controller.

```
100 DIM H$(3)[64]
105 H$(1)="GDNNLNNNGGGGGGGGGGGGGGGGGGGGG
      GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG"
106 H$(2)="GNDNLNNNGGGGGGGGGGGGGGGGGGGGG
      GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG"
107 H$(3)="GNNDLNNNGGGGGGGGGGGGGGGGGGGGG
      GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG"
.....
1130 OUTPUT : 1;"W"&H$(J)
```

There is a bit in one of the HP-IL addresses that represents the status of the ICU buffer. The value of this bit can be either one (the ICU buffer is not empty), or zero (the ICU buffer is empty). This bit can be referenced by using the following statement.

```
520 S=BIT(MOD(SPOLL('RS232') DIV 256, 256),2)
```

In the above statement, the "BIT" function returns the value of a single bit of an integer. The "SPOLL" function returns a value that represents one or more status bytes from an HP-IL device.

The MOD function returns the remainder of an integer division operation. Table 8 shows the layout of the RS-232 device bits.

Table 8 Layout of the RS-232 Device Bit Scheme

3rd byte								2nd byte								1st byte							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1
multiplier x 256^2								x 256								x 1							

CHAPTER FIVE
REMOTE COMPUTER BASIC AGV INTERACTION

5.1 The Communication System

Functionally the Apogee AGV/Remote computer communication system consists of four separate subsystems as shown in Figure 9. Any computer with either an RS-232 or RS-422 serial port can be used as the remote computer in an Orbitor AGV system. The Automatic Dispatching Unit (ADU) is floor mounted communications unit sold by Apogee Robotics, Inc. The primary function of the ADU is to serve as an electrical transmission link between the serial input/output port of the remote computer and the radio frequency transmission to/from the Orbitor. The Intersection and Communication Unit (ICU) is the vehicle mounted link between the radio frequency transmission of the ADU and the HP-71B computer.

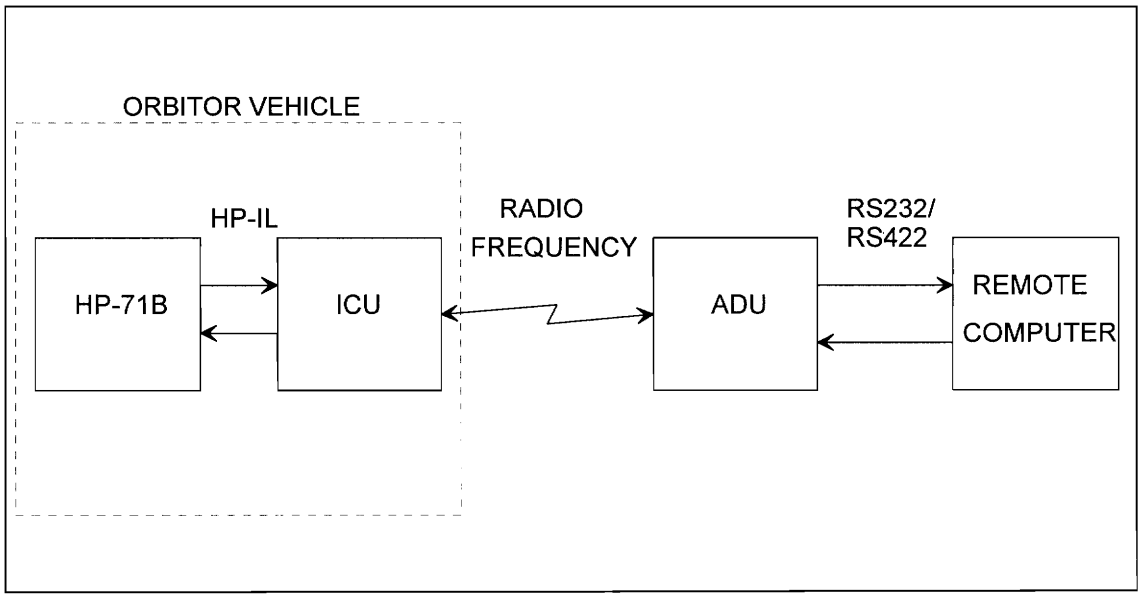


Figure 9 Communication System Components

The vehicle side of the communication is controlled by the HP-71B computer as the controller of the HP-IL interface. The ICU is device number 2 on the HP-IL. The remote computer controls the other side of the communication system. The remote computer needs a serial port that can be connected with the ADU by an RS-232 or RS-422 cable. The communication between the remote computer and the ADU can be achieved by any appropriate programming language that is implemented on the remote computer.

The radio frequency communication between the ADU and the Orbitor is limited in distance. It is strongly recommended that the vehicle be within nine feet of the ADU while communication takes place. For the best results, from experiments in this project, this distance should be within six feet. The ADU will not be able to communicate with the Orbitor when the distance between them is greater than fifteen feet. Computer monitor interference can cause the communication to fail between the ADU and the Orbitor. By position the computer monitors in a direction that is not facing either the ADU or the Orbitor, this problem can be eliminated. Foreign objects, such as metal tables and chairs, can also block radio frequency communications.

There are four LEDs on both the ACE board and the modem of both the ADU and the ICU. The #4 LED of the modem is on when there power is supplied. The #1 LED of the modem is off when idle and it will be on at the same time as the #2 LED of the ACE board when the modem is in transmission. The #2 LED of the modem is off when idle and bright when detecting transmissions. The #3 LED of the modem is on when idle and flickers when receiving a signal. The #1 and #3 LEDs of the ACE board are inactive always. The #2 LED of the ACE will go on while transmitting a message through the modem antenna. The #4 LED of the ACE board will go on when transmitting a message to the computer, either the

HP-71B or the remote computer. The most important LED to pay attention is the #4 LED of the ACE board, since this one signals the receiving of a message.

5.2 Remote Computer Configuration

While any computer with either an RS-232 or RS-422 serial port can be used as the remote computer in an Orbitor AGV system, this project employed an IBM 7552 Industrial Computer. The computer was configured with a 386 Central Processing Unit, eight Mega Bytes of Random Access Memory, a 120 Mega Bytes capacity hard disk drive, and a 1.44 Mega Bytes three and one half inches floppy disk drive.

Table 9 The Pin Functions of the IBM 25 Pin Serial Port

Pin	Function	Mnemonic
2	Transmit data	TXD
3	Receive data	RXD
4	Request to send	RTS
5	Clear to send	CTS
6	Data set ready	DSR
7	Signal ground	GND
8	Carrier detect	CD
20	Data terminal ready	DTR
22	Ring indicator	RI

A serial cable was used to connect the COM1 port of the IBM computer and the ADU. The cable was configured as shown in the Figure 8. Table 9 summarizes the pin functions of the 25 pin serial port of the IBM 7552 computer.

In this installation, the number two pin transmits data from the computer to the ADU. The number three pin receives data from the ADU. The number seven pin is the signal ground. At the computer, pins 4 and 5 and pins 6, 8, and 20 are connected to satisfy any electrical handshake that might be needed by the computer hardware operations. The ADU switch must be set to RS-232.

Software development on the remote computer was done under the MS-DOS Version 6.0 operating system. All remote computer programming was done in MS-QBASIC.

5.3 Software Communication Concepts

Conceptually, the communication process is simply a dialogue between the HP-71B computer and the remote computer. However, the reality of the system is that there are three separate dialogs involves: HP-71B computer/ICU dialogue, ICU/ADU dialogue, and ADU/remote computer dialogue.

To send a "message" from the remote computer to the HP-71B computer, the remote computer first sends a message to the ADU via the serial communications cable. The ADU returns an acknowledgment to the remote computer (via the serial communications cable) and it send the message to the ICU via radio frequency communications. The ICU then sends the message to the HP-71B computer via the HP-IL communications loop. The HP-71B acknowledge receipt of the message to the ICU.

An acknowledge receipt of any messages is always necessary in order to prevent out of sequence communications. If an acknowledgment is not issued

by the remote computer or the HP-71B computer upon receipt of a message, the ADU or the ICU will automatically repeat the original message about every eighteen to twenty seconds. The message sequence number can be applied to govern this message re-transmitte matter both in the remote and the HP-71B computer.

The communications dialogue between the HP-71B computer and the ICU and the communications dialogue between the remote computer and the ADU are known and are the subjects of this chapter and the subsequent example. However, the true nature of the communications between the ICU and the ADU is not known. Information on this process is not found in the HP documents and it was not monitored during this project.

5.3.1 Message Formats

There are three different message formats in the HP-71B/remote computer communication system of the Orbitor.

$###xxxxxxxxBB<CR><LF>
@###xxxxxxxxxxxxBB<CR><LF>
!###<CR><LF>

where
$ = start character of a message of length 16
@ = start character of a message of length 20
! = start character of an acknowledge message
= numeric message sequence number

x = application dependent variable data

BB = block check characters for error checking

<CR> = ASCII Carriage Return

<LF> = ASCII Line Feed

$ and @ messages are sent between the computers as data carrying units. The $ messages are used to transmit 8 bytes of data (the xxxxxxxx's) while @ messages carry 12 bytes of data. All data is transmitted as ASCII characters. The ! messages are used to acknowledge receipt of a message. These messages are passed between the HP-71B computer and the ICU and between the remote computer and the ADU. They represent "local" acknowledgment of a good message.

The block check characters (BB) are used to check the message for possible transmission errors and formed by checksum subroutines in both the remote and the HP-71B computers. The checksum of a $ or @ message is determined by adding the value of each character of the message plus the sequence number digits and then modulating the sum by 256. The result is then expressed as two hexadecimal digits and the ASCII character codes for those digits form the two block check characters (BB). If, for some reason, it is desired to transmit a message without using the checksum block checking subroutines, ASCII CHR$(25) must be placed in both of the block check characters.

The numeric message sequence numbers are used to insure successful receipt of a message within the HP-71B/ICU and ADU/remote computer loops. The $ or @ messages carry a sequence number as three ASCII digits - ###. The receiving unit checks for transmission errors (by calculating and checking the

block check characters) and then acknowledges the message by transmitting a ! message with the same sequence number as received.

The content of the actual $ and @ messages is completely left to the user. The only restriction is that the message must have either 8 or 12 data bytes.

5.3.2 Communications Software Considerations - Remote Computer

There are several essential BASIC coding techniques that are necessary to program the remote computer successfully. First, the serial ports of the remote computer has to be opened as a file address for communication with the ADU. This was accomplished on the IBM 7552 Industrial Computer as follows:

```
320 OPEN "COM1:9600, N, 8, 1" FOR RANDOM AS #1
```

COM1 indicated that the RS-232 cable is attached to the computer on the serial port number 1. For proper communications between the ADU and the computer, the port must have a configuration of 9600 baud, no parity, 8 data bits, and 1 stop bit. This is accomplished in the "9600, N, 8, 1" part of the statement. These parameters are predefined in the Apogee "Orbitor" AGV System Owner's Manual. There are three options in the QBASIC in determine the mode of the serial port that is about to be opened. The port can be FOR INPUT, FOR OUTPUT, or FOR RANDOM. Since the remote computer needs two way communication with the ADU, the COM1 port must be opened for both input and output, which is referred to the RANDOM mode. The "#1" in the example above is a logical address for a file or a location for a device. Hence, subsequent input/output statements access the ADU by referencing this logical address.

The communication between the remote computer and the ADU will not function if the COM1 port has not opened. Additionally, a properly configured serial cable must be connected between the computer and the ADU before the communication port is opened. The ADU does not need to be turned on before opening the COM1 port. The port can be opened in the "Immediate" window in the QBASIC by pressing the function key F6. This will ensure that the serial port is opened successfully.

The remote computer can send a message to the ADU by executing a statement as follows:

```
1180 PRINT #1, TRAN$
```

The "#1" needs to be the same number as the logical address that was specified earlier in the OPEN command of the same program. The "TRAN$" is a message string with the following comment. It must be either a properly formatted $ or @ message. The contents of data bytes (xxxxxxxx's) are completely arbitrary but the sequence numbers should be assigned to messages in an ascending order. This is done to insure that acknowledgments are associate with the messages in the right order.

The ADU will acknowledge a message sent by the remote computer with a ! followed by the three-digit message sequence number that was part of the remote computer's message and then carriage return and line feed characters. This acknowledgment is sent by the ADU when it has received the message from the host successfully. The ADU also transmits the message via radio frequency signals to the ICU on the Orbitor.

In order for the remote computer to verify that the message has been sent, the "LOC" command can be applied. According to Mackenroth's QBASIC Primer Plus (1991), the "Variable = LOC(n)" states "Find the present location in buffer n and return that value to the Variable." In serial communication, the LOC statement is used to return the number of characters that have been read/send in the input/output buffer. It is important to verify that the #1 device buffer has the message that was sent by the remote computer to the ADU. This can be accomplished as the example below.

```
1270 A$ = INPUT$(LOC(1), #1)
1280 PRINT "THE MESSAGE FROM THE HOST TO THE AGV IS:"; A$
```

The LOC statement will not reflect the complete message that was sent from the remote computer to the ADU right away. Executing a dummy 1500-count loop in front of the LOC statement will usually help in collecting the full length of the outgoing message through the LOC of the #1 device.

The format of incoming and outgoing messages between the remote computer and the ADU are the same as they are between the HP-71B and the ICU. All three types of messages were explained previously. Since the COM1 buffer serves both as an input and an output buffer, the program in the remote computer needs to have the ability to verify that the message sent from the HP-71B computer located in the #1 buffer of the remote computer is not the same one that had been sent out by the remote computer itself earlier. There are couple ways to resolve this likely to happen dilemma. First, the LOC can be cleared by reading the current input. So, the next message in the buffer will be the one that was sent from the HP-71B computer. Second, exam the incoming

message before take it, since both the ADU and the ICU will sending messages out repeatedly. The program in the remote computer can keep checking the incoming message until the right one has arrived. This project employed both of the above methods. More detailed explanation is provided in the next chapter.

5.3.3 Communications Software Considerations - HP-71B Computer

The logic in coding the HP-71B computer for the communication purpose is generally the same as it in the remote computer. The approaches, however, are quiet different. There is a bit in one of the HP-IL addresses that represents the status of the ICU buffer as mentioned in Chapter 4. The program in the HP-71B must examine this bit before a message can be received from the remote computer. Communications between the HP-71B computer and the ICU is done via the HP-IL and no additional wiring is necessary. The following statements illustrate necessary commands to receive a message from the remote computer/ADU to the ICU/HP-71B computer.

```
500 'START'
520 S = BIT(MOD(SPOLL('RS232') DIV 256,256),2) @ DISP "WAITING
    HOST" @ WAIT 5
530 IF NOT S THEN 'START'
  .....(set counter)...
570 ENTER : 2; R$(J)
  .....(check message length)....
600 DISP "HOST MSG= "&R$(J) @BEEP @WAIT 3
  .....(call checksum subroutine)....
640 A2$=X3$&R$(J)[2,4]&X2$
```

```
650  OUTPUT : 2; A2$ @DISP A2$[1,4]
```

In the above example, the HP-71B and the ICU will be running an endless loop until a message is received from the remote computer (the S bit will be a 1). The ENTER command is needed to collect the message into a string. This string is then examined by its length, message sequence number, and block checking characters. The acknowledge string from the HP-71B to the ICU is assembled in line 640 where X3$ is a ! and X2$ is <CR><LF>. The acknowledgment of a message successfully received to the HP-71B is expressed in the line 650 above. The status of the ICU buffer will be reset (to a value of 0) after the HP-71B has issued the acknowledgment string. The message sequence number must be included in the string for this process to function. The number 2 in both line 570 and line 650 is referring the device number 2 which is the ICU of the HP-IL communications loop.

The program in the HP-71B computer must use the same device number, 2, in sending messages to the remote computer via the ICU/ADU. The format of any outgoing message in the HP-71B computer is the same as the message format in the program of the remote computer. The $ or the @ character needs to be the first character of a message followed by three ASCII digits as the message sequence number, and either 8 or 12 arbitrary user defined message contents, finally two block checking characters at the end of the message. For example, the S2$ has been constructed according the above criteria, the following statements will forward the S2$ to the remote computer via the ICU/ADU.

```
780  OUTPUT : 2; S2$
790  ENTER :2; B$
```

820 DISP "ACK"B[1,4] @WAIT 2

The S2$ is forwarded to the ICU (and then to the ADU and the remote computer) in line 780. There is no communication port to open or wire to connect beforehand between the ICU and the HP-71B computer. The status buffer of the ICU in the HP-IL will not be cleared until the execution of the line 790. Since the ICU will automatically send an acknowledge message to the HP-71B computer upon receipt of the S2$. This acknowledge message issued by the ICU uses the same format as the acknowledge message from the ADU to the remote computer. It has a ! followed by three-digit message sequence number that was part of the message from the HP-71B and then carriage return and line feed characters. It is necessary for the HP-71B to collect this acknowledgment from the ICU into a dummy string, B$, in order to reset the status of the ICU for the next assignment. The content of the ICU acknowledge message can be displayed, as shown in line 820, for user's information but not mandatory.

5.3.4 Communications Software Considerations - Both Systems

There are two special useful commands available in this system. FC<CR><LF> can be sent either from the HP-71B to the ADU or by the remoter computer itself to the ADU to cancel any commands being sent by the ADU to the remote computer. FTxxxx<CR><LF> can be sent by the HP-71B to the ICU. The ICU will echo the complete message back to the HP-71B including the optional number of letters (x). This FT command is useful in debugging if the ICU is communicating with the HP-71B or not. There is an ICUTEST program written in the HP-71B computer to perform the communication test between the HP-71B

and the ICU of the HP-IL by employing the FT command. The contents of this program is listed in the Appendix C.

CHAPTER SIX
EXAMPLE OF REMOTE COMPUTER DRIVEN AGV SYSTEM

To illustrate remote computer control of the Orbitor, a demonstration guide path was created and software was written and debugged for the HP-71B computer and for the IBM 7552 Industrial Computer. This chapter documents that demonstration.

6.1 The Demonstration Guide Path

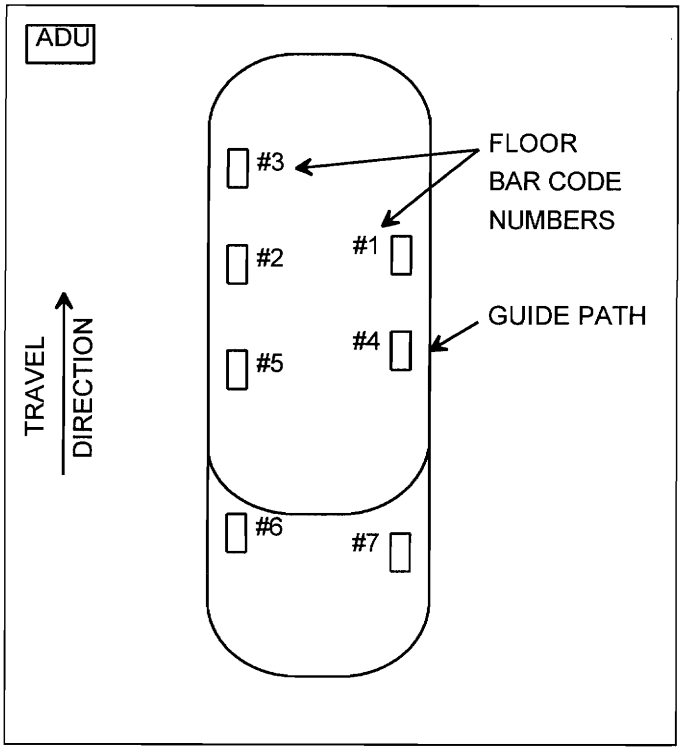


Figure 10 The Demonstration Guide Path

The guide path that was applied in front of the Modern Manufacturing Process Laboratory on the first floor of the Whittemore Hall for demonstrating this project is shown in the Figure 10. There are two loops in this layout. A bar code value of four was used in the program to direct the Orbitor to go either on the big loop or on the small loop. A bar code value of three indicated the home location of the Orbitor. Bar code values of one and two designated P/D stations. The other bar code values do not have any specific function in this layout. They were placed in the layout to demonstrate that the Orbitor can distinguish different floor bar codes precisely. The bar code values are actually used by the HP-71B computer as indices to arrays of task functions. The HP-71B program defines what task or activity is associated with a specific code through the passing of a W string to the vehicle controller.

6.2 Demonstration Software Protocol

The AGV can have one active floor bar code mapping W string at one time. In order to direct the Orbitor to the specified station for every task assignment, every station will need its own W string to guide the AGV. The floor bar code mapping W strings for the AGV to follow are defined in three places in the THEO program which is the demonstration program in the HP-71B computer of the Orbitor. The first one is for the ADU/HOME location. The second one is for the stations that are along the big loop and the last one is for the small loop. The difference of the W strings for the big and the small loop is at the number four code of their strings. The AGV will follow the left path to travel along the big loop, and the right path to go on the small loop. The first eight elements of the W strings of the stations and the ADU are listed in Table 10.

Table 10 The Assignment of Floor Bar Code Mapping W Strings

Location	BCode	Station	"W" str	0	1	2	3	4	5	6	7	8	...
P/D 1	1	101	B. loop	G	D	N	N	L	N	N	N	G	G
			S. loop	G	D	N	N	R	N	N	N	G	G
P/D 2	2	102	B. loop	G	N	D	N	L	N	N	N	G	G
			S. loop	G	N	D	N	R	N	N	N	G	G
ADU	3	Home	B. loop	G	N	N	D	L	N	N	N	G	G
			S. loop	G	N	N	D	L	N	N	N	G	G

The length of communicating messages between the HP-71B computer and the remote computer can be either 16 ($ type) or 20 (@ type) characters as mentioned earlier. The length of messages and their sequence numbers were used in the demonstration programs to distinguish the message types.

The demonstration program, REMOTE1.BAS, that was used in the remote computer, only sent the $ type messages with the message sequence number from 100 to 199. Each message begins with a "$" sign, followed by three numeric digits, eight characters of message, two block check characters, and it ends with a carriage return and a line feed. The numeric digits were converted from the three-digit message sequence number. The eight characters of message contains the commands that were inputted by the user. The user message includes a Y (Yes) or N (No) character that tells the HP-71B whether this message contains commands or not, the AGV destination station number (home-300, station 101, or station 102), the required task for the AGV (Pickup, Delivery, or stay Home), the big (B) or the small (S) loop for routing purpose, and two dummy characters (AA) for filler. The two block checking characters were formed and attached to the @100 series message that was formed by the user

after checksum calculation has performed. Lastly, ASCII carriage return and line feed characters are added to the end of this message.

No message acknowledgment will be issued unless a checksum has performed for the message. This stays true for both the remote computer/ADU and the HP-71B/ICU. However, there is a special character, CHR$(25) of the ASCII table, that can be used to substitute the block checking characters. This CHR$(25) character was used in the @300 and @400 series messages that will be discussed below.

There were two different message lengths ($ and @ types) and three different message sequence numbers (200's, 300's, and 400's) that were used in the demonstration program, THEO, of the HP-71B computer. The messages that use the 2xx sequence numbers started with a $ followed by the sequence number and a brief message plus block checking characters and the carriage return and the line feed characters. The demonstration program THEO used $200 type messages to let the remote computer know that the AGV was ready for the next assignment.

The messages that had either 300s or 400s as their message sequence number were feedback messages from the HP-71B computer to the remote computer. Both of the 300s and the 400s messages started with the @ character. The @300 type message stated that there was a tote on the lift and the AGV could not perform the pickup task which was specified in the $100 type message from the remote computer. When the THEO program in the HP-71B computer sent an @400 type message to the remote computer, it stated that there was no tote on the lift to deliver as requested by the $100 type message from the remote computer. If the AGV could perform the task that was specified

from the $100 message from the remote computer, the program THEO sent an @200 type message to the remote computer. The message formats are summarized in Table 11.

Table 11 Message Formats of the Demonstration Software

Characters	$100 MSG.	$200 MSG.	@200 MSG.	@300 MSG.	@400 MSG.
1	$	$	@	@	@
2	1	2	2	3	4
3	0	0	0	0	0
4	0	0	0	0	0
5	Y/N	A	*	T	L
6	3/1/1	G	A	O	I
7	0/0/0	V	G	T	F
8	0/1/2	R	V	E	T
9	P/D/H	E	-	-	-
10	B/S	A	A	O	I
11	A	D	C	N	S
12	A	Y	K	-	-
13	Block Ch.	Block Ch.	-	L	E
14	Block Ch.	Block Ch.	O	I	M
15	<CR>	<CR>	K	F	P
16	<LF>	<LF>	*	T	T
17			Block Ch.	Block Ch.	Block Ch.
18			Block Ch.	Block Ch.	Block Ch.
19			<CR>	<CR>	<CR>
20			<LF>	<LF>	<LF>

6.3 The HP-71B Program

The program THEO was written to control the HP-71B for the demonstration of this project. In this example program, there is a main program and nine subroutines. The flow charts and the source code of the main program and all subroutines are presented in Appendix A.

6.3.1 General Program Organization

The main program first initializes various variable and arrays and then causes the Orbitor to go to the home location and wait for a message. The incoming message is checked and the status of the lift is considered. If there is a problem, a message is sent back to the remote computer for further input. If no problems exist, the main program establishes which W sting is needed and directs the vehicle controller appropriately. After the desired task is completed, the main program calls the travel subroutine to return the vehicle to the home position. A message is then sent to the remote computer and the vehicle waits for a new request from the remote computer.

The CHKBAT subroutine calculates the difference between the voltage at the E-Stop button and the ground offset voltage of the vehicle and return the result to the main program. If this voltage reading is less than 11.5 volts, the program will prompt users with a "Low Voltage" message.

The CHKTOTE subroutine verifies the presence of a tote box on the lift assembly of the Orbitor. This subroutine is used in the DELIVER and PICKUP subroutines, and in the main program.

The DELIVER subroutine is called when a delivery task has been assigned to the Orbitor and the vehicle has arrived at the P/D station. This subroutine will check the existence of a tote box on the station and on the lift assembly first. After verified that it is feasible to perform a delivery function, the lift arm will be raised, extended, and lowered so that a tote box can be dropped onto the P/D station. After that, the CHKTOTE subroutine will be called to verify that there is no tote box on the lift assembly. This is followed by retracting the lift arm back.

The INITLIFT subroutine initializes the lift assembly to the zero position where is 21.75 inches above the floor and to retract the arm of the lift assembly for the reason of safety. This subroutine is executed near the beginning of the main program before the TRAVEL subroutine starts to take the vehicle to its home position for any assignment.

The LOOK subroutine activates the sonar on the lift assembly to check the presence of a tote box on the P/D station upon a delivery or a pickup task. This subroutine is called from both the DELIVER and the PICKUP subroutines.

The PICKUP subroutine performs similar activities as the DELIVER subroutine. It will verify the presence of a tote box on the P/D station first. Then, the arm of the lift assembly will be zeroed, extended, raised, and retracted. The CHKTOTE subroutine will then be called to check the toto box on the lift. Finally the lift will be lowered to its zero position.

The TRAVEL subroutine takes the Orbitor to assigned destination. The first time when the TRAVEL subroutine was called in the main program, it will direct the AGV to its home position for communication with the remote computer. After each task, the TRAVEL subroutine will be called again to take the vehicle to

the home position for the next assignment. During the execution of the TRAVEL subroutine, the vehicle is moving and collecting the value of floor bar codes that the vehicle has passed and other feedback: such as bumper status, sonar obstruction, tachometer pulse status, etc. This subroutine has a routine function to "speak" the value of a floor bar code by calling the voice synthesizer in the Orbitor. This was implemented to insure that the codes were read properly.

The subroutine CHKSUM checks $ or @ messages (both outgoing and incoming) and generates the block check characters. The block check characters are determined by adding the value of each character of the message plus the sequence number digits and then modulating the sum by 256. The result is then expressed as two hexadecimal digits and the ASCII character codes for those digits form the two block check characters.

The subroutine CHECK is called to check the second character of any response issued by the vehicle controller as the result of an action caused by a command code. As summarized in Table 6, command codes such as RR or RE will initiate a response of either an R0 or an R1, this CHECK subroutine is utilized to check the 0 (zero), operation failed, or the 1, operation normal.

6.3.2 General Program Flow

The headers, counters, flags, arrays, strings, and some definitions were declared at the beginning of the program THEO. The vehicle is then brought back to the home position by establishing the correct W string and calling the TRAVEL subroutine. The lift assembly is initiated after every fifty tasks and at the beginning of the first task. The AGV will be directed to its home location where

the ADU is located before it is ready to receive any message from the remote computer. The vehicle then waits for a message from the remote computer. The task to be done is encoded into the message as described earlier. As soon as the AGV arrives at the home location, the program THEO will be running an endless loop until it has sensed a message in the ICU that was sent from the remote computer. Before the HP-71B can evaluate the data content of this message, the length and the message sequence number of the message will be verified. The check sum subroutine will be called to compare the block checking characters. The program will loop back to the ICU status checking on incoming messages if any of the verifications failed.

The program will then break the message string into separate characters in order to determine the station number, the required task, and the desired loop size. Meanwhile, the HP-71B is checking the existence of a tote on the lift assembly. After all the conditions are sorted out, the HP-71B will send a message to the remote computer stating that the AGV can or cannot perform the asking task by using one of the @200, @300, or @400 type messages.

If the AGV can perform the task asked by the remote computer, an @200 series message will be sent back to the remote computer. If the AGV has a load on the lift and the message from the remote computer requires a pickup task, an @300 series message will be sent back to the remote computer. Therefore, the remote computer can prompt the user either to remove the tote from the lift or to send a delivery message. If the AGV has no load on the lift and the message from the remote computer asks for a delivery task, an @400 series message will be sent back to the remote computer. Hence the user of the remote computer will be prompted to either put a load on the lift or to change to a pickup task.

The message content is used to determine correct W string to send to the vehicle controller as explained in section 6.2. The TRAVEL subroutine is called after the AGV has determined the desired W string. In the TRAVEL subroutine, the HP-71B keeps entering the status from the vehicle controller. The status can be a floor code number, bumper has been depressed, or normal operation messages. The AGV will perform either a pickup or a delivery function after it has arrived the specified station. This is done by calling either subroutine PICKUP or subroutine DELIVER.

The lift assembly operates relatively slow when compared to the execution speed of the HP-71B computer. Therefore, the CHECK subroutine is called many times while the AGV is doing a pickup or delivery task in order to pause the HP-71B. (reference to the Appendix A, line 3180 to line 3225.) Before the AGV does either a pickup or a delivery, the sonar of the lift assembly is activated, by calling the LOOK subroutine, to check the presence of a tote box on the station. The subroutine LOOK is to return a true or false statement in determine whether the lift is able to pickup or deliver a load. At the end of either a pickup or a delivery job, the lift assembly will be lowered to the zero height and the arm will be retracted for the safety reason, since it will be traveling to the ADU afterwards.

The program will then assembly another message to be sent to the remote computer to indicate that the task has completed and the Orbitor is ready for the next assignment. The check sum subroutine will be utilized in forming the block checking characters of this outgoing message. The voltage of the batteries will be measured by the CHKBAT subroutine while waiting for the remote computer to receive the message. If the voltage is lower than 11.5 volts, the program will prompt the user with a beep and a "Battery Low" statement on the HP-71B screen.

6.3.3 Line-by-Line Explanation

The line by line explanations of the main program and the subroutines are presented in bullet styles below.

Main Program:

- Lines 10 to 30, a simple header for the program was written.
- Lines 40 to 70, typical starting statements for the HP-71B computer.
- Lines 80 to 430, variable and array declarations, where N5=200 is the message counter for normal operations, E2=300 is the message counter for cannot pickup tote error, and E3=400 is the message counter for cannot deliver tote error.
- Line 435, initialize the lift assembly.
- Line 440 to 450, send the AGV to the ADU/HOME for instructions that will be transmitted from the remote computer.
- Line 500, 'START' label.
- Line 520, the ICU buffer status.
- Line 530, if the ICU buffer is empty, the program will continue looking for message that will be sending to the ICU.
- Line 540 to 560, counters for message verifications.
- Line 570, takes message from the ICU to the R$ of the HP-71B
- Line 580 to 600, check the received message and display it.
- Line 610 to 630, checksum verification.
- Line 640 to 655, the HP-71B acknowledges the ICU for a message received OK.
- Line 660 to 670, check two of the message characters.
- Line 690, to find out is there any tote on the lift already or not.

- Line 700 to 730, decide what is the relation between the received message and the condition of the lift assembly of the Orbitor.
- Line 740 to 840, when there is a tote on the lift, the AGV cannot do a pickup, so sending a 300 series error message back to the remote computer then go back to 'START'.
- Line 850 to 950, similar to the previous one but this one has no tote to deliver.
- Line 955 to 966, situation normal, the AGV can do the asked task and an AGV-ACK-OK message will be sent to the remote computer.
- Line 1000 to 1030, find out the station number and the loop size.
- Line 1040 to 1070, the floor code declaration for the big loop.
- Line 1080, directs the Orbitor to the specified station.
- Line 1100 to 1110, to decide either do a pickup or a delivery.
- Line 1120 to 1190, to do a task around the small loop, similar to the previous one.
- Line 1200 to 1250, to do a pickup.
- Line 1260 to 1310, to do a delivery.
- Line 2000 to 2004, the AGV go back to the ADU/HOME location for the next task.
- Line 2010 to 2050, create and send an AGVREADY message to the remote computer.
- Line 2070 to 2090, check the voltage of the Orbitor.
- Line 2110 to 2140, reset counters and go back to 'START' for another task.

CHKBAT subroutine:

- Line 2200 to 2240, CHKBAT subroutine header.
- Line 2250 to 2280, find out the ground voltage of the AGV.
- Line 2290 to 2330, find out the voltage at the ESTOP of the AGV.

- Line 2340, compares voltage.
- Line 2350 to 2380, report voltage status and end.

CHKTOTE subroutine:

- Line 2400 to 2440, CHKTOTE subroutine header.
- Line 2450 to 2480, find out if there is any tote on the lift and end sub.

DELIVER subroutine:

- Line 2500 to 2540, DELIVER subroutine header.
- Line 2560 to 2570, find out if there is any tote on the lift and report status.
- Line 2590 to 2600, call LOOK subroutine for checking is there any tote has already delivered to the P/D station and report status.
- Line 2610 to 2635, raise the lift, extend the arm, lower the lift, and check the status of the lift.
- Line 2650 to 2710, check the status of the lift to verify if there is any tote left on the lift.
- Line 2720, retracts the arm of the lift assembly.
- Line 2725 to 2730, check the status of the arm and end sub.

INITLIFT subroutine:

- Line 2800 to 2830, INITLIFT subroutine header.
- Line 2840 to 2850, retract the arm and zero the lift assembly.
- Line 2860 to 2870, reset the counter and end sub.

LOOK subroutine:

- Line 2900 to 2940, LOOK subroutine header.
- Line 2950, give this task three trials.
- Line 2960, activate the lift sonar.

- Line 2970 to 2990, collect the distance between the obstruction and the lift sonar.
- Line 3000 to 3040, determine if there is an object in front of the sonar or not and end the sub.

PICKUP subroutine:

- Line 3100 to 3130, PICKUP subroutine header.
- Line 3140, trur/false flag.
- Line 3150, label.
- Line 3160 to 3170, call LOOK to see if there is any blockage in front of the lift assembly.
- Line 3180 to 3240, retract the arm, zero the lift, extend the arm, raise the lift, then retract the arm, lower the lift. Also check the status accordingly.
- Line 3250 to 3260, check the status of the tote on the lift.
- Line 3262 to 3270, zero the lift and end the sub.

TRAVEL subroutine:

- Line 3300 to 3340, TRAVEL subroutine header.
- Line 3350, set the flag and the counter.
- Line 3360, use the "W" string and the specified floor code string to direct the AGV on the guide path.
- Line 3370, use the "G1" to start the traveling of the AGV in case there was any errors. The G1 can resume the traveling function of the vehicle.
- Line 3380, the vehicle normally travels with speed "G8". This "G8" can be any desired speed.
- Line 3400 to 3420, if there is an error happened five times, the AGV will be stopped.

- Line 3430 to 3450, request the query status from the Orbitor microprocessor from the HP-71B computer and save the status in the A$.
- Line 3460, if the most recent status reflects an "F" prefix, direct the program to the 'FLRCODE' label.
- Line 3470, if the status is not an "I" type status, then go get it again.
- Line 3480 to 3550, comprehend the status and react or prompt the user accordingly.
- Line 3580 to 3650, display and read out the number of the floor code that was passed by the Orbitor. End the sub.

CHKSUM subroutine:

- Line 3700 to 3730, CHKSUM subroutine header.
- Line 3740 to 3780, add the ASCII value of each character of the check sum string, an incoming or an outgoing message, together.
- Line 3790 to 3820, modulate and replace the value with its own complement value that was calculated previously.
- Line 3830 to 3850, the modified sum byte value is then split into two bits, the high order 4 bits -- B1, and the lower order 4 bits -- B2.
- Line 3860 to 3870, if the equivalent DEC value of the binary B1 (or B2) is greater than 9 then convert B1 (or B2) to a letter starts from A (ASCII "A" = 65 = 55 + B1, if B1 = 9 +1 = 10). If the B1 (or B2) is less than or equal to 9, then convert B1 (or B2) to a number (ASCII "1" = 49 = 48 + 1, if B1 = 1)
- Line 3880 to 3910, form the check sum string by converting the B1 and the B2 to numeric characters and return the string. End the sub.

CHECK subroutine:

- Line 4000 to 4040, CHECK subroutine header.

- Line 4050 to 4090, find out the value of the second digit of the status message in the current buffer of the Orbitor controller. If it is a 1 then the most recent action was performed successfully or the value will be a 0. Display an error message if the value is zero. End the sub.

6.4 The Remote Computer Program

The REMOTE1.BAS program was written for the remote computer to communicate with the THEO program of the AGV on-board HP-71B computer. The flow charts and the source code of the main program and all subroutines are presented in the Appendix B.

6.4.1 General Program Organization

The main program declared the boundaries for all the variables, arrays, and subroutines at the beginning. There are several precaution prompts printed on the screen to guide the user for proper sequence of HP-71B/remote computer operations. The serial port, COM1, was then opened for communication between the remote computer and the ADU. The outgoing message that contains user inputted task instructions from the remote computer to the HP-71B will then be formed by an interactive fashion between the remote computer and the user. After the checksum characters added to the message string, this message is then transmitted to the HP-71B/ICU via the ADU.

The successful transmission of the message form the remote computer to the ADU is verified by the ADU acknowledgment. The LOC(1) is a buffer that

stores the contents of the RS-232 port. The transmission condition of the message is detected by counting how many characters are in the LOC(1). In the main program, this counting is used whenever there is a communication event between the ADU and the remote computer. Since the HP-71B will send a message to the remote computer for the status of the AGV after it has received the message from the remote computer, it is needed for the program in the remote computer to verify the feedback message of the HP-71B computer. By assigning the feedback message with a different length (20) from the outgoing message (16) that came from the remote computer, the program in the remote computer is robust in comprehending the acknowledgment from the HP-71B computer.

The CLEARLOC subroutine is called in the program to clear the LOC(1) buffer. This subroutine keeps reading whatever is in the buffer until there is nothing left in it. This is an effective way of emptying the buffer of the serial port which may have the old message from the ADU. It takes about 1000 to 1500 counts between the time a message is sent to the ADU and the time the whole message is collected in the LOC buffer. Therefore, before every input from the LOC(1), there is a dummy loop.

The subroutine CHKSUM checks both outgoing and incoming messages and generates the block check characters. The block check characters are determined by adding the value of each character of the message plus the sequence number digits and then modulating the sum by 256. The result is then expressed as two hexadecimal digits and the ASCII character codes for those digits form the two block check characters. The CHKSUM subroutine is called for the "$200*AGVREADY*" message that is sent from the HP-71B to the remote computer at the end of a task. This practice provides a communication

verification between two parties for every task and gives the user a choice to continue or to quit if the checksum operation failed.

6.4.2 General Program Flow

The headings and declarations are listed at the beginning of this program. The user is prompted with several necessary information concerning the THEO program, the ADU and the serial cable right after the REMOTE1 was executed. The user is then asked to input task related information after the serial port is opened. The first message that is going to the ADU contains the message sequence number ("100"), the station number (300 for home, or P/D station 101 or 102), the desired loop (B or S), the type of task (P, D, or H), and the block checking characters will be formed. In the formation of this message, the user is asked by series of questions followed by the check sum calculation and the final message string assemblies.

In this program, the remote computer sends out the first message, the 16-character "$" type. The HP-71B will interpret the message and send a "@" type, 20-character, message back to the remote computer. The remote computer will then prompt the user whether the AGV can perform the required task or the reason why the AGV can not perform the asked task. The user will have opportunities to continue this two way communication between the remote and the HP-71B computer within the REMOTE1.BAS program until there is a feasible task has been assigned for the AGV to perform. While the Orbitor is doing its job, the program in the remote computer keeps examining the contents in the LOC(1) buffer. If it is not the "$200*AGVREADY*", end-of-task message from the AGV,

the program in the remote computer will continue scanning and clearing the contents in the LOC(1) buffer by using the CLEARLOC subroutine and a loop.

Upon receipt of the end-of-task message, the CHKSUM subroutine will be called and the block check characters will be verified. The user has an opportunity to start another task for the Orbitor or to end at this point.

6.4.3 Line-by-Line Explanation

The line by line explanation of the example program, REMOTE1.BAS, in the AGV remote computer is listed below in bullet styles.

Main Program:

- Line 10 to 90, program headings.
- Line 100 to 200, declaration of the arrays, variables, and subroutines.
- Line 210 to 300, prompt the user some basic information.
- Line 310 to 330, open the com1 port of the remote computer.
- Line 340 to 390, set default communication string elements.
- Line 430 to 460, prompt the user to run the 'THEO' program of the HP-71B computer.
- Line 480 to 610, ask the user to select a station.
- Line 630 to 810, ask the user to select the type of the task that can be pickup, delivery, or stay home.
- Line 830 to 970, ask the user to select the big or the small loop.
- Line 990 to 1070, verify the selected functions with the user.
- Line 1090, the label for many branches to loop back.
- Line 1100, assemblies the message string for check sum calculation.

- Line 1110 to 1130, perform the check sum of the message string and print it out.
- Line 1140, finalizes the outgoing message with the Carriage Return and the Line Feed attached at the end of the check sum characters.
- Line 1180, sends the message out to the ADU.
- Line 1190 to 1220, a dummy loop to provide time for the LOC to receive msg.
- Line 1230 to 1250, determine has the LOC received enough information yet.
- Line 1260 to 1290, collect the outgoing message form the LOC so that the LOC can be emptied.
- Line 1300 to 1310, message counter.
- Line 1320 to 1430, while the AGV is doing the tote checking, the remote keeps waiting for the feedback message from the AGV.
- Line 1440 to 1460, inputting the feedback message to a string.
- Line 1470 to 1500, a dummy string to buy some time.
- Line 1510 to 1570, check the message sequence number of the received message.
- Line 1580 to 1600, separate the message sequence number for the next step.
- Line 1610 to 2190, depends on the message sequence number, prompt the user some questions or direct the program to a waiting stage while the AGV is doing the required task.
- Line 2210 to 2360, using the same logic in receiving the AGV-READY message from the AGV at the end of a task.
- Line 2380 to 2540, take the message into a string and prepare for the check sum.
- Line 2550 to 2560, do a check sum on the received message.

- Line 2570, calls the CLEARLOC subroutine to reset the status of the LOC(1) buffer.
- Line 2580 to 2920, depend on the result of the check sum calculations, prompt the user to continue or to quit. If it is a normal operation, the program will loop back and asks the user to input station number, and so on.

CHKSUM subroutine:

- Line 5000, CHKSUM subroutine header.
- Line 5010 to 5030, declare dimension and variables, find the length of the message string.
- Line 5040 to 5060, add the ASCII value of each character of the check sum string, an incoming or an outgoing message, together.
- Line 5070 to 5100, modulate and replace the value with its own complement value that was calculated previously.
- Line 5110 to 5130, the modified sum byte value is then split into two bits, the high order 4 bits -- B1, and the lower order 4 bits -- B2.
- Line 5140 to 5150, if the equivalent DEC value of the binary B1 (or B2) is greater than 9 then convert B1 (or B2) to a letter starts from A (ASCII "A" = 65 = 55 + B1, if B1 = 9 +1 = 10). If the B1 (or B2) is less than or equal to 9, then convert B1 (or B2) to a number (ASCII "1" = 49 = 48 + 1, if B1 = 1)
- Line 5160 to 5190, form the check sum string by converting the B1 and the B2 to numeric characters and return the string. End the sub.

CLEARLOC subroutine:

- Line 6000 to 6110, The CLEARLOC subroutine. This subroutine uses a conditioned loop to empty the content of the LOC(1) buffer.

6.5 The ICUTEST Program

The ICUTEST program in is to check the communication between the HP-71B computer and the ICU within the HP-IL communications loop in the Orbitor. A successful execution of the ICUTEST program indicates that the HP-IL is working properly. This has found very useful in debugging communications between the remote computer and the HP-71B computer.

In this program, an "FT" command is used to send a dummy message to the ICU. The ICU will echo the message back to the HP-71B if the communication link of the HP-IL between them is trouble-free. The line by line explanations are listed in the Appendix C with the program.

The power of the Orbitor and the HP-71B computer need to be turned on before executing this program. The ICUTEST program can be ran by typing "RUN ICUTEST" then press the [ENDLINE] key at the keypad of the HP-71B computer. The screen of the HP-71B computer will display the outgoing message from the HP-71B to the ICU, a waiting message for a short period of time, and the ICU response message. If this program can not be executed successfully, press the reset button, which is illustrated in Figure 6, of the modem board and run it again.

CHAPTER SEVEN
KNOWN PROBLEMS AND FUTURE OPPORTUNITIES

7.1 The AGV Hardware Problems

There exist several hardware problems within the Orbitor AGV system that are identified here. Each of these is discussed in detail below. The sensor that can sense the vertical retro-reflective tape of the P/D station is not working. The lift assembly can either be raised to the upper extreme or lowered to the zero point instead to the specific height that is asked by a program. The light beams associated with the tote box bar code reader are not focused on the tote box, so no codes can be read from tote boxes. The battery charging strips are not linked to the batteries, hence the concept of the automatic charging station is not feasible.

The Orbitor is designed to stop at the next P/D station after encountered the floor bar code C. The sensor that senses the signal associated with this function is located inside the side panel of the vehicle which is facing the P/D station. When this sensor is not functioning properly, the Orbitor will be endlessly seeking for the P/D station after passed the floor bar code C. In this work, instead of using the floor code C, floor bar code D (which causes the Orbitor to stop after twelve inches after passed the code) was used. In the W strings of Table 10, there is a D in each string.

The height of the lift assembly of the Orbitor is designed to be programmable. There is a Hall Effect sensor mounted on the motor of the lift assembly. When this sensor can not detect the pulses caused by the rotation of the drive motor, the lift assembly will not go to the HP-71B computer specified

height. Consequently, there was no specified height for the lift assembly to reach during either a pickup or a delivery task in the example program THEO.

The tote box bar codes are read by two photo sensors that emit light beams toward the bar code while the tote box is being retracted by the arm of the lift assembly. The size of the bar code and the image of the light beam need to be the same. However, they are not close to the same size at all and the size of the light beam appears to be unadjustable. Consequently, this feature was not used in this work.

The Orbitor has three charging strips located near the floor surface at the middle bottom of the vehicle. They are designed to match a set of charging contacts at the vehicle charging station. The batteries were designed to be located beneath the surface of the vehicle bed. They are currently setting on top of the bed and can be charged by using a regular automobile battery charger. These hardware restrictions made charging the Orbitor a lengthy manual process.

7.2 The Opportunity for Future Studies

There are many opportunities for future studies of the Orbitor AGV system. Multiple ADUs along the guide path can be driven from a single remote computer since the remote computer communicates with only one ADU at a one time. The RS-422 cable can be utilized for further communication distance between the ADUs and the remote computer as well. More powerful programming tool, such as many of the popular C++ compilers, in the remote computer can be employed for multi-tasking, object oriented programming.

Feedback mechanisms can be incorporated into the P/D station to provide the remote computer information of the presence of tote boxes. The user of the remote computer can make a more accurate assignment for the Orbitor before sending out a message string to the HP-71B computer if the tote box status of the P/D stations are known.

The data transferring between the HP-71B computer within the HP-IL interface can be integrated to have a computer as one of the devices of the loop and cables can be applied to download or upload information to or from the HP-71B computer. The contents of the programs in the HP-71B can not be printed, they can be collected by writing the codes down line by line from the display of the HP-71B. The coding process of the HP-71B computer is not a pleasant task, since the HP-71B is mounted vertically at the front panel of the Orbitor. If a computer that can be linked to the HP-71B via the HP-IL or other feasible means for more practical file management within the Orbitor AGV system, the process of developing and debugging programs for the Orbitor can then be accelerated.

The Orbitor remote computer can be linked to an overall system controller that monitors and controls all the activities in a simulated computer integrated manufacturing environment. This Orbitor can be programmed with respect to other devices such as conveyors, robots, automatic storage and retrieval system, and machining cells within a system. The system controller can provide priorities and information for the Orbitor to follow and the functions of the Orbitor can be re-assigned base on the activities that are happening within the whole system.

A guide path is proposed for the future new Engineering Building that would employ two ADUs, three possible P/D stations, and a charging area for the

Orbitor. This is illustrated in Figure 11. The AGV can stop at the end of the conveyor of the Flexible Machining and Assembly System, at the conveyor near the robot/automatic machining work cell, and/or at the P/D station near the center of the room. There is a branch designed for the AGV to go to the charging area upon any low battery conditions. The corner radii along the guide path are 30 inches. The clearance of all the boundaries is sufficient for the Orbitor to travel in full speed without any blockage.

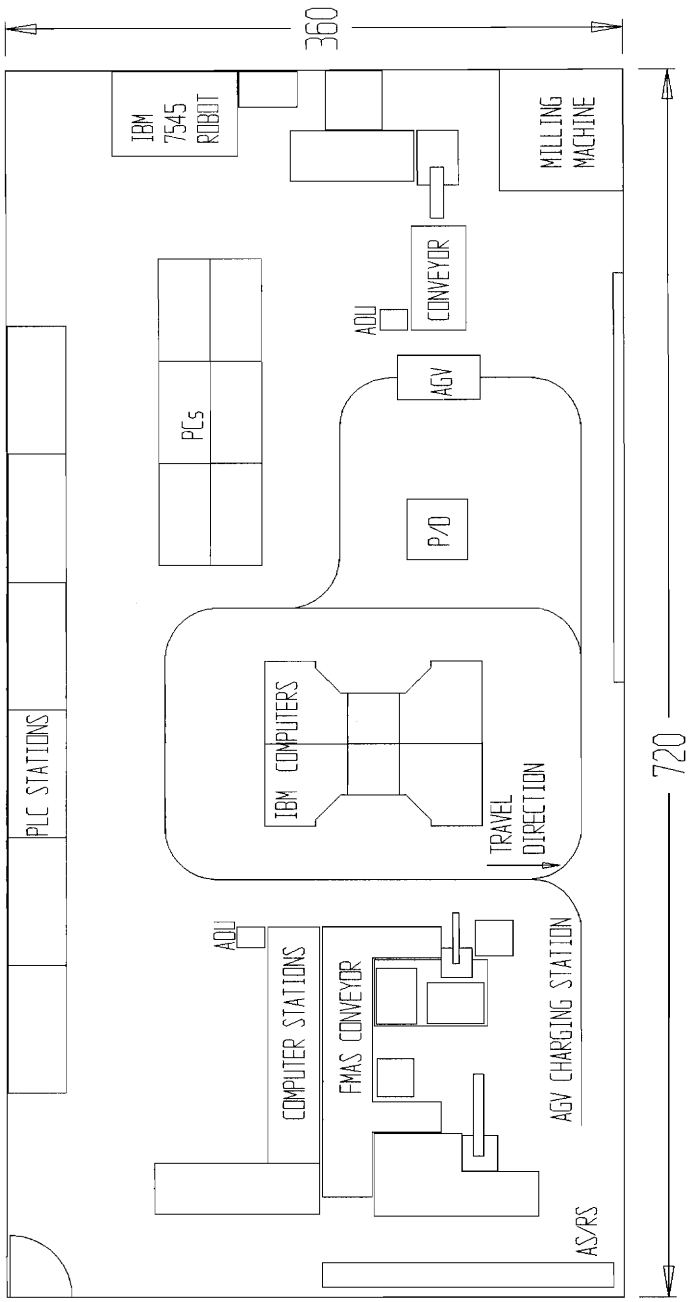


Figure 11 The Proposed Guide Path for the New Engineering Building

CHAPTER EIGHT
CONCLUSION AND RECOMMENDATIONS

There were two main objectives to be accomplished in this project. The first one was accomplished by creating a functional AGV system with the Orbitor that incorporates advanced hardware features. This system included a taped double-loop guide path with four meaningful bar codes on the floor, the HP-71B communication linkage between the Orbitor and the ICU, the radio frequency signal transferring between the ICU and the ADU, and the serial communications between the remote computer and the ADU.

The second objective was satisfied by established a software system for the overall control of the Orbitor AGV system. This objective involved creating software for the HP-71B and the remote computer. Additionally, software communication protocols were implemented and tested between the remote computer and the ADU, and between the ICU and the HP-71B computer.

At this stage, users of the Orbitor AGV system can operate the AGV by issuing instructions from the remote computer to the Orbitor. Acknowledgment from the HP-71B computer can provide the user either the reason why the Orbitor can not perform the required task due to the tote box status on the lift assembly or the Orbitor is ready to go. At the end of each task, the Orbitor will prompt the user of the remote computer for instructions of the next task. The interactive operation between the user and the Orbitor were programmed to repeat itself.

There are several recommendations that might be helpful to the future of this Orbitor AGV system. P/D station feedback mechanisms could provide information to the remote computer to indicate the status of tote boxes. This

information can also be used by the overall system controller in controlling the operations of the whole simulated computer integrated manufacturing environment.

The physical configuration of the P/D station can be created to have a plate that indicates the existence of a tote box on the station in front of the station with the same height as the tote box sonar of the lift assembly when it is at the zero position. So that the tote box sonar can be activated in front of a P/D station without raise the lift assembly in checking whether there is a tote box on the station or not.

An effort to make the hardware system of the Orbitor flawless would be another recommendation. Since there are still exist several hardware problems associated with the Orbitor as mentioned in the previous chapter, the user of the Orbitor would have more challenges in programming and operating the AGV system after those defective parts of the vehicle have been replaced and/or repaired.

The final recommendation would be integrating a more powerful computer that is governed by the overall system controller into the Orbitor HP-IL system. This computer would not only benefits the dynamic controls the Orbitor AGV system but also provides the flexibility in programming the Orbitor.

REFERENCES

Apogee "Orbitor" Automatic Guided Vehicle System Owner's Manual. (1987). Ft. Collins, CO: Apogee Robotics, Inc.

Horn, J., (1985). HP-71 BASIC Made Easy. Berkeley, CA: Synthetix.

HP-71 Owner's Manual. (1987). Corvallis, OR: Hewlett-Packard Portable Computer Division.

HP-71 Reference Manual. (1987). Corvallis, OR: Hewlett-Packard Portable Computer Division.

HP 82401A HP-IL Interface Owner's Manual for the HP-71. (1985). Corvallis, OR: Hewlett-Packard Portable Computer Division.

Mackenroth, D. R., (1991). The Waite Group's QBASIC Primer Plus. Carmel, IN: The Waite Group, Inc.

Orbitor Automatic Guided Vehicle System Operation and Maintenance Manual. (1990). Ft. Collins, CO: Apogee Robotics, Inc.

APPENDIX A

THE EXAMPLE PROGRAM OF THE AGV ON-BOARD COMPUTER

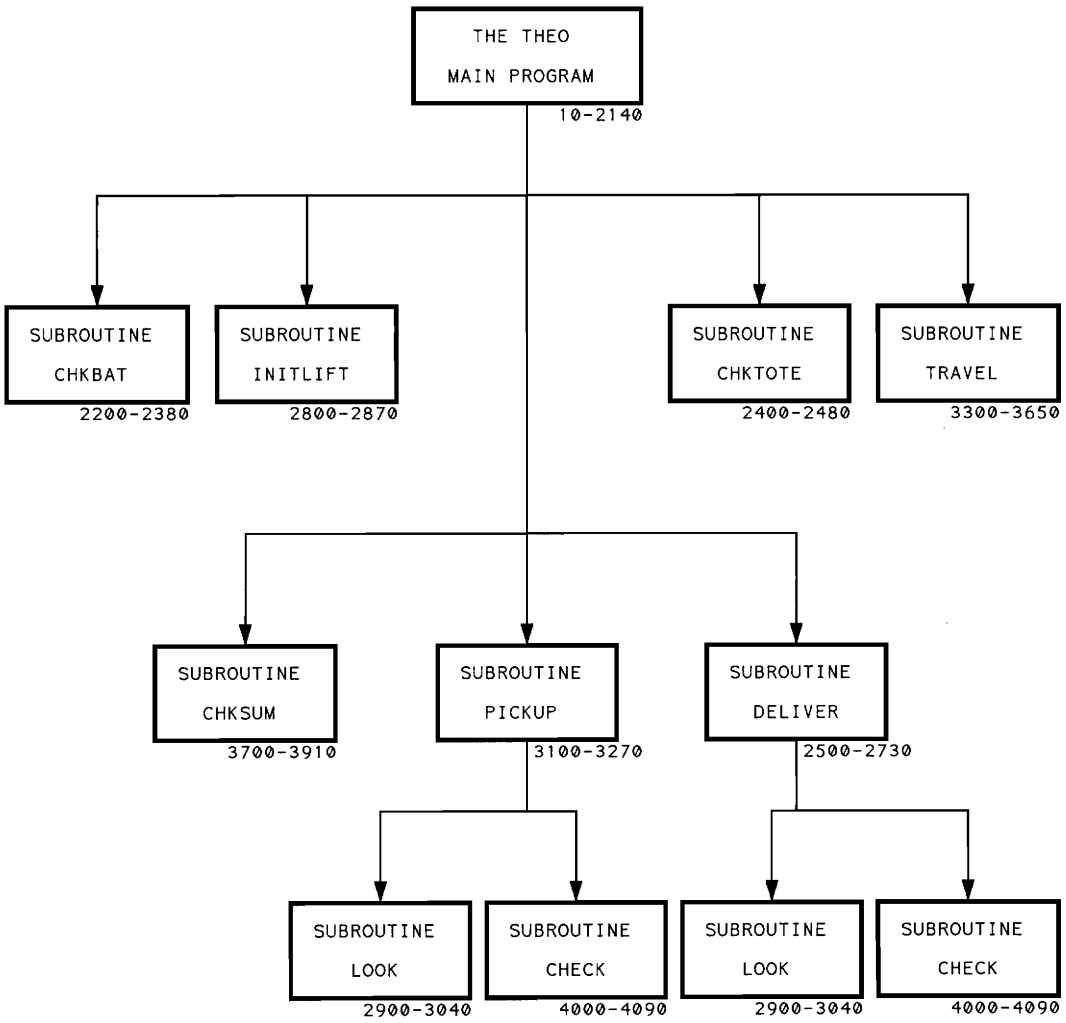


Figure 12 The Flow Chart of the Structure of the Example Program THEO

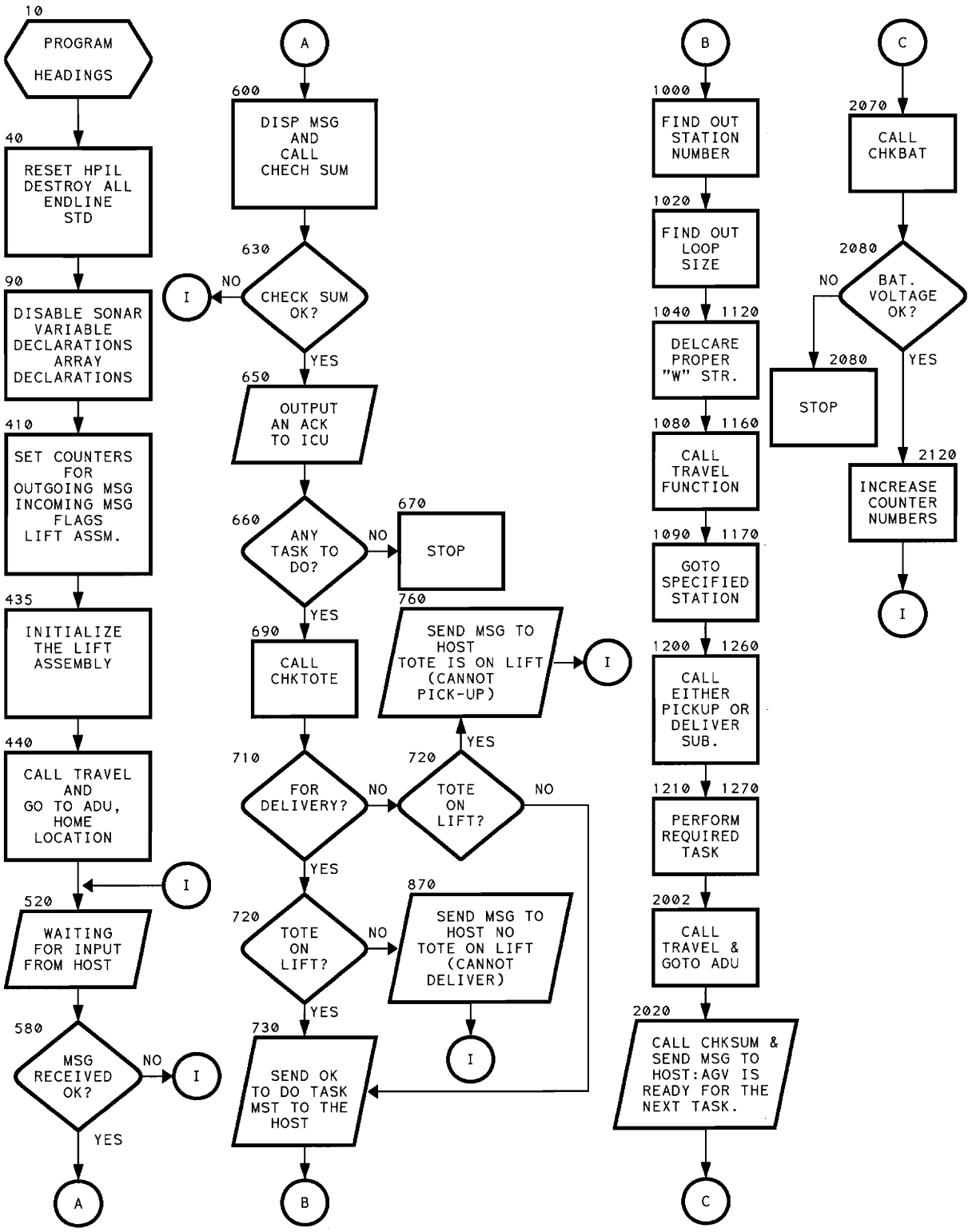


Figure 13 The Flow Chart of the THEO Main Program

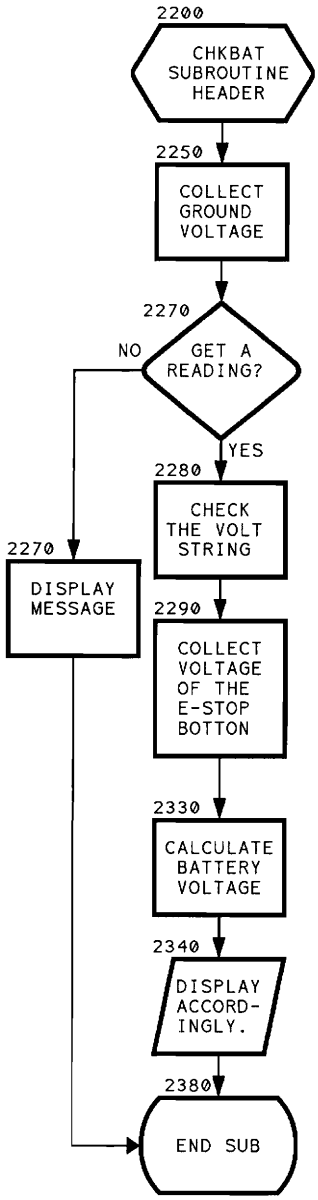


Figure 14 The Flow Chart of the CHKBAT Subroutine

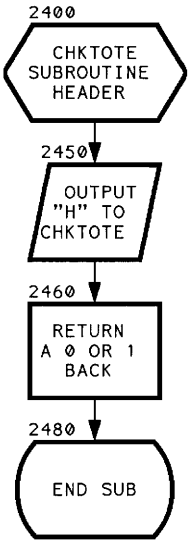


Figure 15 The Flow Chart of the CHKTOTE Subroutine

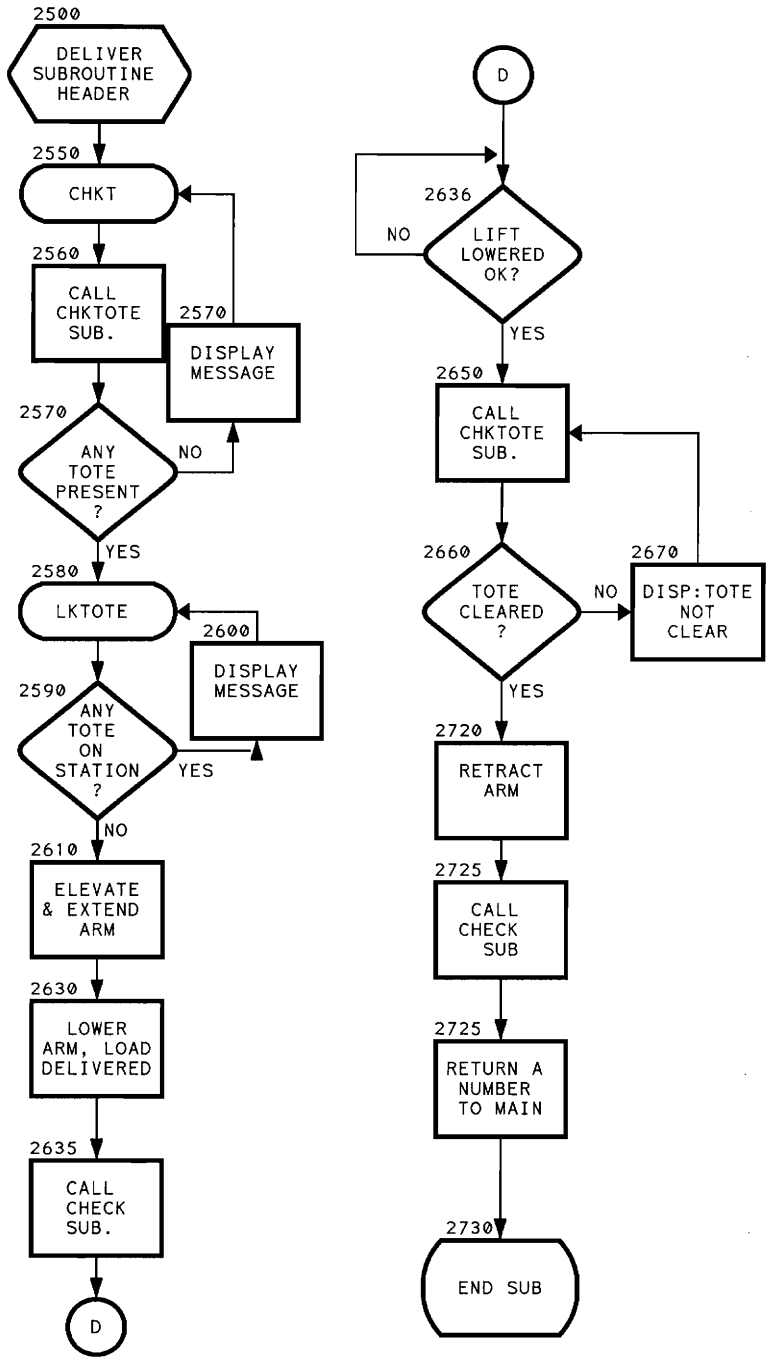


Figure 16 The Flow Chart of the DELIVER Subroutine

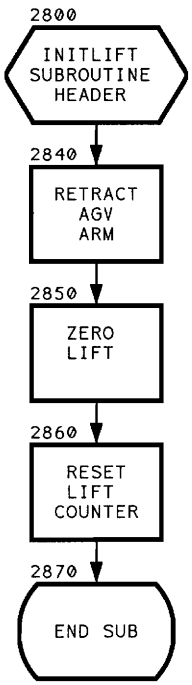


Figure 17 The Flow Chart of the INITLIFT Subroutine

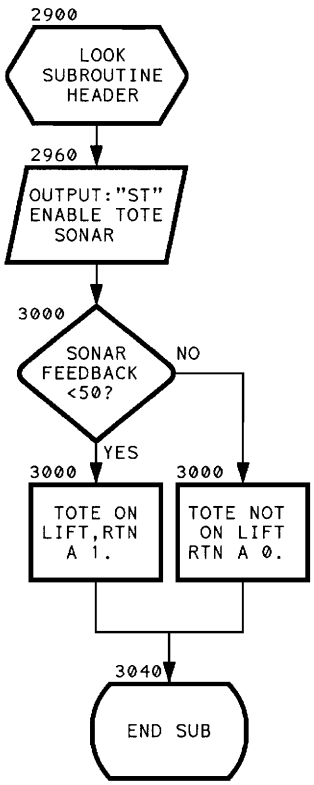


Figure 18 The Flow Chart of the LOOK Subroutine

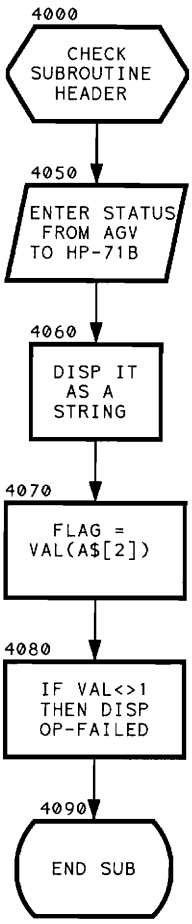


Figure 19 The Flow Chart of the CHECK Subroutine

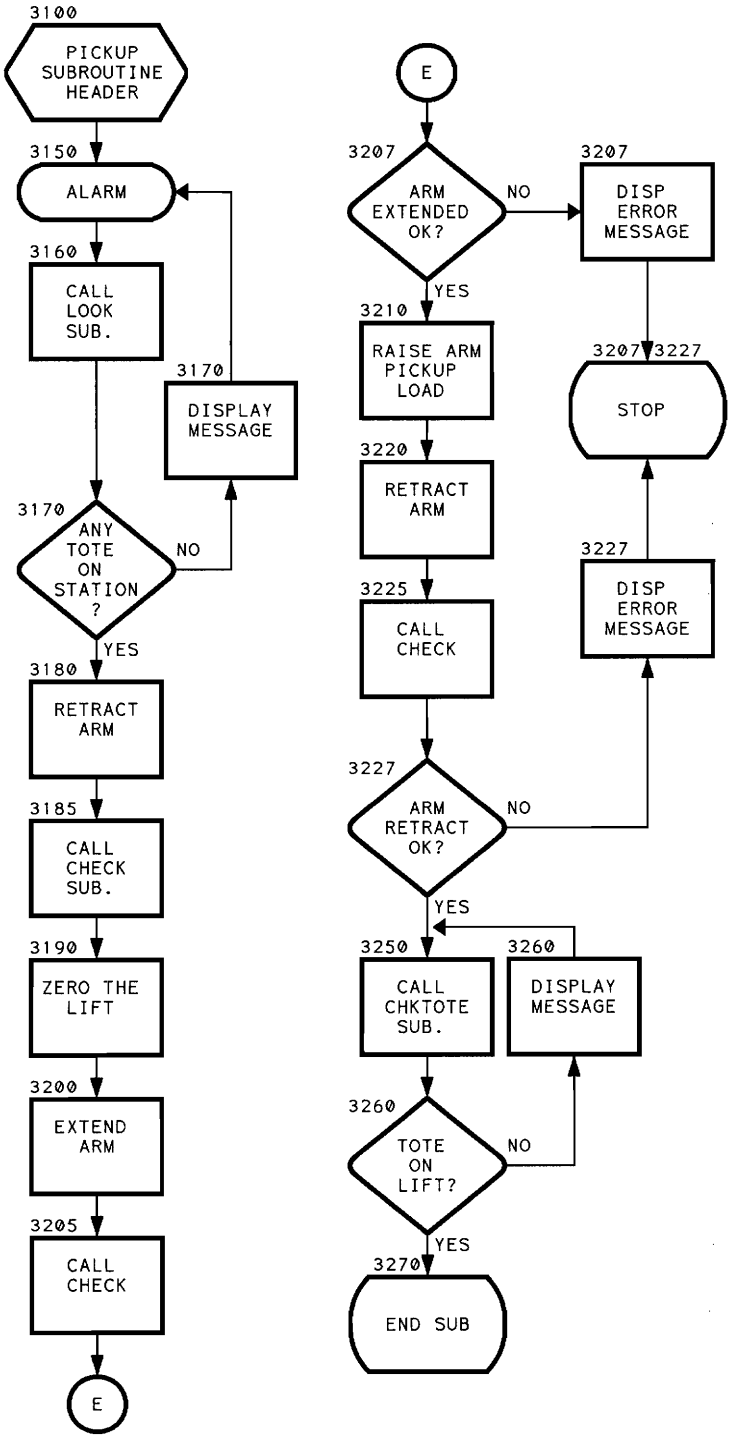


Figure 20 The Flow Chart of the PICKUP Subroutine

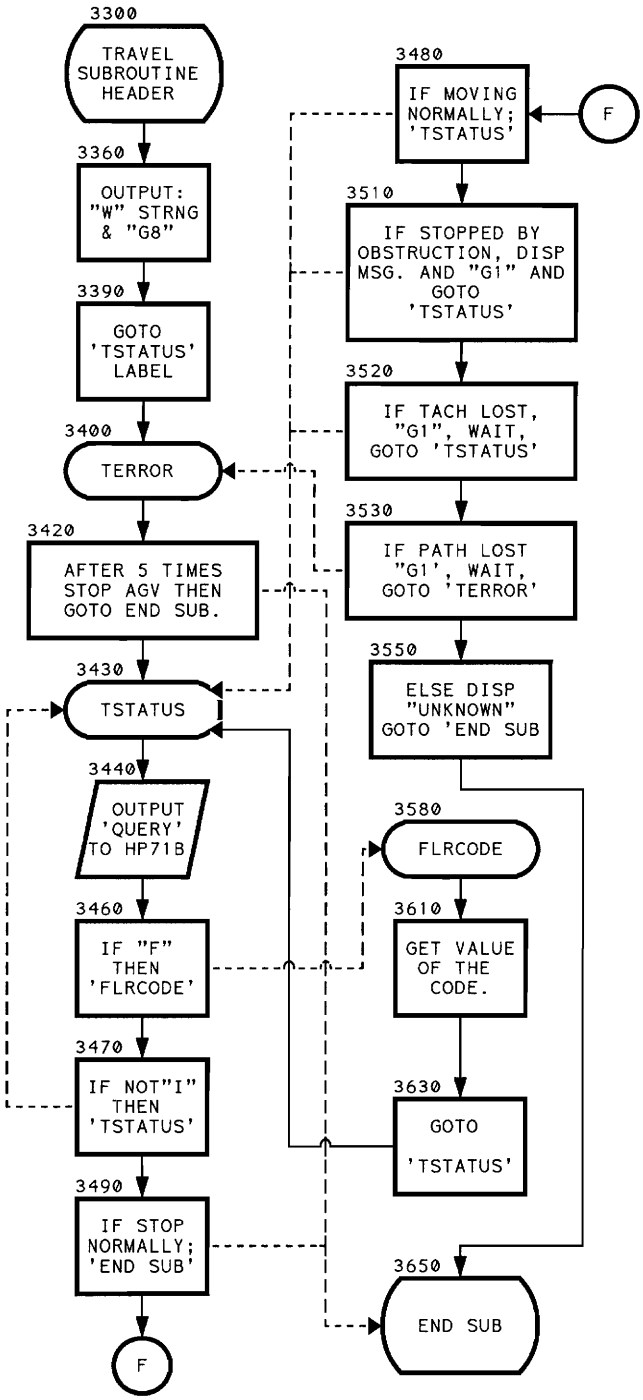


Figure 21 The Flow Chart of the TRAVEL Subroutine

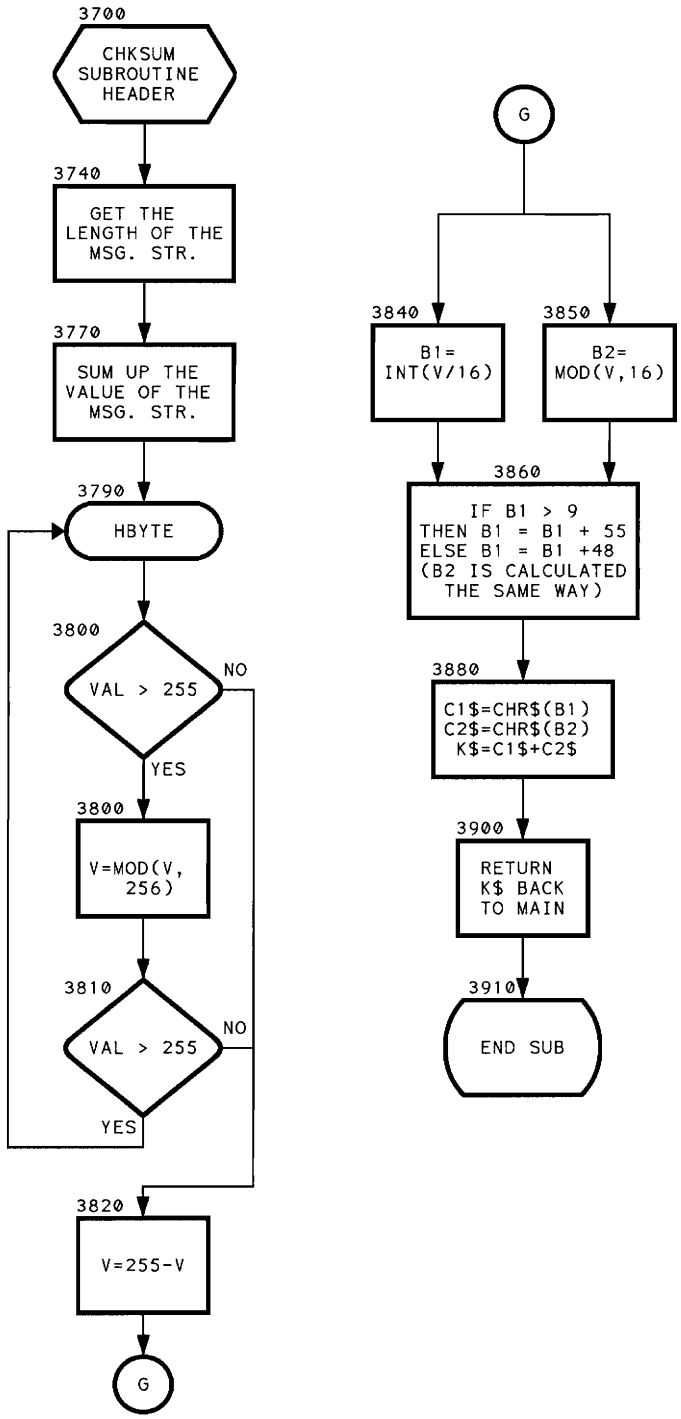


Figure 22 The Flow Chart of the CHKSUM Subroutine

The example program of the AGV on-board computer -- THEO

```
10    ! VIRGINIA TECH  DEPT OF ISE
20    ! PROGRAM NAME: "THEO" BY ROBERTO F LU 12/31/93
30    ! MATCHING HOST PROGRAM: "REMOTE1.BAS"
40    RESET HPIL @ DESTROY ALL  ! HP-71B is controller, clear memory
50    OPTION BASE 1             ! Set the base line for all the arrays
60    ENDLINE                   ! Automatic <CR><LF>
70    STD                       ! Standard display format for numbers
80    DISP "--- VT ISE AGV ---" @WAIT 1
90    OUTPUT:1; "SD"            ! Disable Sonar
100   Z0 = 21.75              ! Height in inches when the lift is zeroed.
110   Z1 = 78                   ! Pulse per inch for the lift motor
120   V1 = 11.5                 ! Minimum battery voltage
130   X1$ = CHR$(36)            ! The $
140   X2$ = CHR$(13)&CHR$(10)   ! 13=CR, 10=LF
150   X3$ = CHR$(33)            ! 33 = "!" String for ack msg.
160   E9$ = CHR$(10)            ! <LF>
170   X4$ = CHR$(64)            ! The @ Char
180   A3$ = "FC"&X2$          ! Cancel commands from the ADU to host
185   X5$ = CHR$(25)&CHR$(25)   !Psudo CHECKSUM for @... MSGs
190   ! S1=STATION # STRINGS
200   ! S$=COM SEND MSG
210   ! R$=COM RCV MSG
220   ! S2$=ERROR MSG STR FOR PICK-UP
230   ! S3$=ERROR MSG STR FOR DELIVER
240   ! D$=PATH TO STATION
250   ! T$=TRAVEL STRING
```

```
260   ! H$=PATH TO ADU/HOME
270   DIM S1$[14],S$[16], R$(10)[16], S2$[20], S3$[20], S5$[20]
280   DIM T$[64], D$(2)[64], H$[64]
400   H$="GNNDLNNNGGGGGGGGGGGGGGGGGGGGG
      GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG"! "W" sting to
      the ADU/HOME
410   X = 0                                ! Lift counter
420   J = 1 @F = 1 @ G = 1 @K = 0          ! other counters
430   E2=300 @E3=400 @N5 = 200  ! 200: ok, 300:can't pick,400:can't deliver
435   OUTPUT :1;"Z";INT(Z0*Z1);E9$  ! Zero the lift assembly
440   CALL TRAVEL(H$, E9$, F, A)     ! Send the AGV to HOME for comm.
450   IF NOT F THEN DISP "TRAVEL ERROR" @STOP
500   'START':                             ! The lable for endless task loop
510   IF X = 50 THEN CALL INITLIFT(Z0, Z1, E9$, X)  ! Init. lift every 50 runs
520   S = BIT(MOD(SPOLL('RS232') DIV 256,256), 2) @DISP "WAITING
      HOST" @WAIT .5          ! Collect the ICU buffer status into S
530   IF NOT S THEN 'START'  ! Keep collecting until S=1, msg from ADU rcv'd
540    IF K = 10 THEN K = 0      ! Msg counter verify
550   J = J + 1 !@K = K + 1
560   IF J= 11 THEN J =1
570   ENTER :2;R$(J)            ! Collect the incoming MSG to R$
580   IF LEN(R$(J))<>14 THEN DISP "BAD LEN" @WAIT 1 @GOTO 'START'
590   ! IF R$(J)[2,4] = R$(K)[2,4] THEN 'START'       ! Check the length and the
      message counter of the incoming msg
600   DISP "HOST MSG= "&R$(J) @BEEP @WAIT 3         ! Disp MSG
610   CALL CHKSUM(R$(J)[2,12],K$)           ! Checksum the MSG
620   DISP "CHECKSUM="&K$ @WAIT 2
```

```
630   IF K$<>R$(J)[13,14] THEN DISP "BAD CHKSUM" @WAIT 2 @GOTO
      'START'              ! Compare the Checksum with the one did by host
640   A2$=X3$&R$(J)[2,4]&X2$          !The ACK msg
650   OUTPUT :2;A2$ @DISP A2$[1,4]          ! Output ACK to ICU
655   OUTPUT :2;A3$                   ! Tell the ADU stop sending msg to host
660   IF R$(J)[5,5] = "N" THEN DISP "NO WORK" @STOP
670   IF R$(J)[9,9] = "H" THEN DISP "STAY HOME, NO WORK" @STOP
680   P = 0 @D = 0
690   CALL CHKTOTE(F)                 ! To see is there any tote on lift
700   IF R$(J)[9,9] = "P" THEN P=1    ! If this is a pickup task, then P=1
710   IF R$(J)[9,9] = "D" THEN D=1    ! If this is a delivery task, then D=1
720   IF F=1 AND P=1 THEN 'ERR3'   ! If tote on lift and asking for pickup,err3.
730   IF F=0 AND D=1 THEN 'ERR4' ELSE 'NOR1'  !If no tote to deliver, err4.
740   'ERR3':
750   E2 = E2 + 1
760   S2$=X4$&STR$(E2)&"TOTE-ON-LIFT"&X5$&X2$   ! Err msg to host
780   OUTPUT :2;S2$ @DISP S2$[1,18] @WAIT 2 @BEEP  ! send err3 to host
790   ENTER :2;B$                ! Take the ICU ack back to clear ICU buffer
820   DISP "ACK"&B$[1,4] @WAIT 2    ! Display the ACK from ICU
840   GOTO 'START'               ! Wait for another msg from the host
850   'ERR4':                    ! Same logic as the ERR#
860   E3 = E3 + 1
870   S3$=X4$&STR$(E3)&"LIFT-IS-EMPT"&X5$&X2$
890   OUTPUT :2;S3$ @DISP S3$[1,18] @WAIT 2 @BEEP
900   ENTER :2;B$
930   DISP "ACK"&B$[1,4] @WAIT 2
950   GOTO 'START'
955   'NOR1':              ! The AGV can perform the asked task by the host
```

```
957   N5=N5+1
959   S5$=X4$&STR$(N5)&"*AGV-ACK-OK*"&X5$&X2$  ! Send host a 200
      msg, which is a ok to perform task message
963   OUTPUT :2;S5$ @DISP S5$[1,18] @WAIT 2 @BEEP
965   ENTER :2;B$
966   DISP "ACK"$B$[1,4] @WAIT 2
1000  IF R$(J)[6,8] = "101" THEN T1=1          ! Find out the station number
1010  IF R$(J)[6,8] = "102" THEN T1=2
1020  IF R$(J)[10,10] = "B" THEN 'LOOPB'       ! Find out the loop size
1030  IF R$(J)[10,10] = "S" THEN 'LOOBS'
1040  'LOOPB':
1050  ! PATH TO P/D STATIONS, ARRAY SUBSCRIPT = BCODE#
1060  D$(1)="GDNNLNNNGGGGGGGGGGGGGGGGGG
      GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG"
1070  D$(2)="GNDNLNNNGGGGGGGGGGGGGGGGGG
      GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG"
1080  CALL TRAVEL(D$(T1), E9$, G, I)           ! Goto the specified station
1090  IF NOT G THEN DISP "TRAVEL ERROR" @STOP
1100  IF P = 1 THEN 'PICK1'                    ! Do either a pickup or delivery
1110  IF D = 1 THEN 'DROP1'
1120  'LOOPS':
1130  ! PATH TO P/D STATIONS, ARRAY SUBSCRIPT = BCODE#
1140  D$(1)="GDNNRNNNGGGGGGGGGGGGGGGGGG
      GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG"
1150  D$(2)="GNDNRNNNGGGGGGGGGGGGGGGGGG
      GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG"
1160  CALL TRAVEL(D$(T1), E9$, G, I)
1170  IF NOT G THEN DISP "TRAVEL ERROR" @STOP
```

```
1180  IF P = 1 THEN 'PICK1'
1190  IF D = 1 THEN 'DROP1'
1200  'PICK1':
1210  CALL PICKUP(Z0,Z1,E9$)          ! Pick up a load
1220  IF NOT G THEN DISP "CAN NOT PICK UP TOTE" @STOP
1250  GOTO 'ECHO'
1260  'DROP1':
1270  CALL DELIVER(Z0,Z1,E9$)         ! Deliver a load
1280  IF NOT G THEN DISP "CAN NOT DELIVER TOTE" @STOP
1310  GOTO 'ECHO'
2000  'ECHO':
2002  CALL TRAVEL(H$, E9$, F, A)      ! Go back to the ADU/HOME after task
2004  IF NOT F THEN DISP "TRAVEL ERROR" @STOP
2010  S1$=STR$(N5)&"AGVREADY"         ! Assembly a ready-to-go msg for host
2020  CALL CHKSUM(S1$,K$)             ! Checksum the outgoing msg
2030  S$=X1$&S1$&K$&X2$
2040  DISP "STATMSG="&S$[1,14] @WAIT 2
2050  OUTPUT :2;A3$                   ! Send the msg out
2060  OUTPUT :2;S$
2065  ENTER :2;B$ @DISP "ACK"&B$[1,4] @WAIT 2
2070  CALL CHKBAT(V1, V2, E9$, E$, F, G)  ! Check the voltage of the bat.
2080  IF NOT G THEN OUTPUT :1; "N" @DISP E$ @STOP
2090  DISP "BATTERY VOLTAGE:";V2
2110  IF N5 = 299 THEN N5 = 200
2120  X = X + 1
2130  GOTO 'START'                    ! Start the task again from the label: START
2140  END          ! The end of the main program, subroutines are listed below
2200  !*************************************************************
```

```
2210  ! SUBROUTINE CHECK BATTERY
2220  ! V = V(BAT) -V(GND),  V=A2-A0
2230  !*********************************************************************
2240  SUB CHKBAT(V1,V2,E9$,E$,F,G)
2250  OUTPUT :1;"A0";E9$              ! Find the ground offset voltage
2260  ENTER :1;A$
2270  IF A$[1,1] <> "A" THEN E$="CHECK 0 FAILED" @GOTO 'ENDBAT'
2280  V0 = VAL(A$[2])/100             ! Convert it to xx.xx format
2290  OUTPUT :1;"A2";E9$              ! Find the voltage on theE-Stop
2300  ENTER :1;A$
2310  IF A$[1,1] <>"A" THEN E$="CHECK BAT FAILED" @ G=0 @GOTO
      'ENDBAT'
2320  G = 1
2330  V2=VAL(A$[2])/100 - V0          ! Calculate the bat. volt.
2340  IF V2 > V1 THEN DISP "BAT VOLTAGE= ";V2 @F=1@GOTO 'ENDBAT'
2350  F=0
2360  E$ = "BATTERY LOW"
2370  'ENDBAT':
2380  END SUB
2400  !*********************************************************************
2410  ! SUBROUTINE CHKTOTE
2420  ! CHECK IF TOTE IS ON AGV
2430  !*********************************************************************
2440  SUB CHKTOTE (F)
2450  OUTPUT :1;"H"      ! Find out if the tote is on the lift, H=1 or not, H=0
2460  ENTER :1;A$
2470  F = VAL(A$[2])
2480  END SUB
```

```
2500 !************************************************************
2510 ! SUBROUTINE DELIVER
2520 !  PLACE TOTE ON SHELF
2530 !************************************************************
2540 SUB DELIVER (Z0,Z1,E9$)
2550 'CHKT':
2560 CALL CHKTOTE (F)
2570 IF NOT F THEN OUTPUT :1;"N";E9$ @DISP "NO TOTE TO
     DELIVER"@GOTO 'ENDL'
2580 'LKTOTE':
2590 CALL LOOK(E$,F)  ! See if there is a tote on the station already
2600 IF F THEN OUTPUT:1;"N";E9$ @DISP "INPUT BLOCKED" @GOTO
     'LKTOTE'
2610 OUTPUT :1;"E";INT(50*Z1);E9$          ! Raise the lift
2620 OUTPUT :1;"RE" @WAIT 3 @CALL CHECK(G)       ! Extend the arm
2630 OUTPUT :1;"Z"; INT(Z0*Z1);E9$  @WAIT 5     ! Lower the lift
2635 CALL CHECK(G)            ! Check if the lift has lowered
2636 ENTER :1;A$ @IF A$<>"Z1" THEN 2636 ELSE 'CLRTT'
2640 'CLRTT':
2650 CALL CHKTOTE(F)          ! Check if the tote is on the lift or not
2660 IF NOT F THEN 'TCLR'     ! TOTE CLEARED
2670 DISP "TOTE NOT CLEAR"
2680 OUTPUT :1; "N"           ! Beep!
2690 WAIT 3
2700 GOTO 'CLRTT'
2710 'TCLR':
2720 OUTPUT :1;"RR"           ! Retract the arm
2725 CALL CHECK(G)
```

```
2730  END SUB
2800  !*********************************************************
2810  !SUBROUTINE INITLIFT
2820  !*********************************************************
2830  SUB INITLIFT (Z0, Z1, E9$, X)
2840  OUTPUT :1;"RR"              ! Retract the arm
2850  OUTPUT :1;"Z"; INT(Z0*Z1);E9$  ! Zero the lift
2860  X = 0
2870  END SUB
2900  !*********************************************************
2910  ! SUBROUTINE LOOK
2920  ! ENABLE TOTE SONAR AND CHECK TOTE ON STATION
2930  !*********************************************************
2940  SUB LOOK(E$,F)
2950  FOR K = 1 TO 3
2960  OUTPUT:1;"ST"               ! Turn the lift sonar on
2970  ENTER:1; A$ @ DISP A$  ! Find out the object distance from the sonar
2980  IF A$[1,1] <>"S" THEN E$="LOOK ERROR" @DISP E$ @ GOTO
      'ENDLOOK'
2990  G = VAL(A$[2])              ! Convert the string number to a value
3000  IF G < 50 THEN F = 1 ELSE F = 0   ! Decide is there a blockage (load)
3010  IF F = 1 THEN GOTO 'ENDLOOK'
3020  NEXT K
3030  'ENDLOOK':
3040  END SUB
3100  !*************************************************************
3110  ! SUBROUTINE PICKUP   (ref to the comments in the Deliver sub.)
3120  !*********************************************************
```

```
3130  SUB PICKUP (Z0,Z1,E9$)
3140  G = 1
3150  'ALARM':
3160  CALL LOOK (E$,F)
3170  IF NOT F THEN OUTPUT:1;"N";E9$ @DISP "NO TOTE TO PICKUP"
      @GOTO 'ALARM'
3180  OUTPUT :1;"RR"
3185  CALL CHECK(G)
3190  OUTPUT :1;"Z";INT(Z0*Z1);E9$
3200  OUTPUT :1;"RE"
3205  CALL CHECK(G)
3207  IF NOT G THEN DISP "ARM ERROR" @STOP
3210  OUTPUT :1;"E";INT(50*Z1);E9$
3220  OUTPUT :1;"RR"
3225  CALL CHECK(G)
3227  IF NOT G THEN DISP "ARM ERROR" @STOP
3228  ENTER :1;A$ @IF A$ <> "R1" THEN 3228 ELSE 'CHKT'
3240  'CHKT':
3250  CALL CHKTOTE (F)
3260  IF NOT F THEN OUTPUT:1;"N";E9$@DISP "NO TOTE PICKED UP"
      @GOTO 'CHKT'
3262  OUTPUT :1;"Z";INT(Z0*Z1);E9$
3265  CALL CHECK(G)
3270  END SUB
3300  !*****************************************************
3310  ! SUBROUTINE TRAVEL
3320  ! INPUT THE "W" STRING AND GO TO THE DESTINATION
3330  !*****************************************************
```

```
3340  SUB TRAVEL (T$, E9$, F, I)
3350  C = 0 @ F = 1
3360  OUTPUT:1;"W"&T$          ! Travel according to the given T$
3370  OUTPUT:1; "G1";E9$       ! Start with speed 1
3380  OUTPUT:1; "G8";E9$       ! Travel with speed 8
3390  GOTO 'TSTATUS'
3400  'TERROR':
3410  C = C + 1
3420  IF C = 5 THEN OUTPUT :1; "G0";E9$ @F = 0@GOTO 'ENDTR'
3430  'TSTATUS':
3440  OUTPUT:1;"Q"              ! Find out the query status
3450  ENTER: 1; A$
3460  IF A$[1,1] = "F" THEN 'FLRCODE'  ! If it a Floor code, goto FLRCODE
3470  IF A$[1,1] <> "I" THEN 'TSTATUS'
3480  IF A$[2] = "00008" THEN 'TSTATUS'     ! Orbitor is moving normally
3490  IF A$[2] = "00000" THEN 'ENDTR'       ! Orbitor has stopped normally
3500  IF A$[2] = "00007" THEN DISP "MOVE PLEASE"  ! Sonar obstruction
3510  IF A$[2] = "00007" THEN OUTPUT : 1; "G1";E9$ @ WAIT 5 @GOTO
      'TSTATUS'                       ! Use G1 to resume operation
3520  IF A$[2] = "00004" THEN DISP "TACH LOST" @OUTPUT:1;"G1";E9$
      @WAIT 2 @GOTO 'TSTATUS'         ! Tachometer pulse absent
3530  IF A$[2]="00005" THEN DISP "PATH LOST" @OUTPUT:1;"G1";E9$
      @WAIT 2 @GOTO 'TERROR'          ! Guide path has been lost
3540  IF A$[2]="00006" THEN DISP "BUMPERS" @WAIT 5 @
      OUTPUT:1;"G1";E9$ @GOTO 'TSTATUS'     !Bumper &/or sice panel
      were actuated
3550  DISP "UNKNOWN I";A$           ! Otherwise
3560  F = 0
```

```
3570  GOTO 'ENDTR'
3580  'FLRCODE':
3590  DISP A$                    ! Display the floor code on HP-71B screen
3600  C = 0
3610  I = VAL(A$[2])             ! Get the value of the code
3620  OUTPUT:1;"0"&A$[2]&"Z"  ! Speak out the number of the floor code
3630  GOTO 'TSTATUS'
3640  'ENDTR'
3650  END SUB
3700  !*****************************************************************
3710  !SUBROUTINE CHKSUM
3720  !*************************************************************
3730  SUB CHKSUM (S$, K$)
3740  L = LEN (S$)               ! Get the length of the msg.
3750  V = 0
3760  FOR I = 1 TO L             ! Repeat the same number as the L
3770  V = V + NUM (S$[I])        ! Get the ASCII value of the msg summed up
3780  NEXT I
3790  'HBYTE':
3800  IF V>255 THEN V = MOD(V,256)          ! Modulate the value so its < 256
3810  IF V>255 THEN 'HBYTE'
3820  V = 255-V
3830  B1=V/16
3840  B1 = INT(B1)          ! The front half of the msg converted to B1, 4 bits
3850  B2 = MOD(V,16)     ! The other half ot the msg converted to B2, 4 bits
3860  IF B1>9 THEN B1 = B1 +55 ELSE B1 = B1 + 48        ! convert them to
3870  IF B2>9 THEN B2 = B2 + 55 ELSE B2 = B2 + 48       ! ASCII letters or #s
3880  C1$ = CHR$(B1)             ! Make B1 & B2 as characters
```

```
3890  C2$ = CHR$(B2)
3900  K$ = C1$&C2$
3910  END SUB
4000  ! ********************************************************
4010  !SUBROUTINE CHECK
4020  !DISPLAY AGV COMMAND RESPONSES 1=OK 0=BAD
4030  !*************************************************************
4040  SUB CHECK (G)
4050  ENTER:1;A$
4060  DISP A$
4070  G=VAL(A$[2])
4080  IF NOT G THEN DISP "OP-FAILED"
4090  END SUB
```

APPENDIX B

THE EXAMPLE PROGRAM OF THE AGV REMOTE COMPUTER

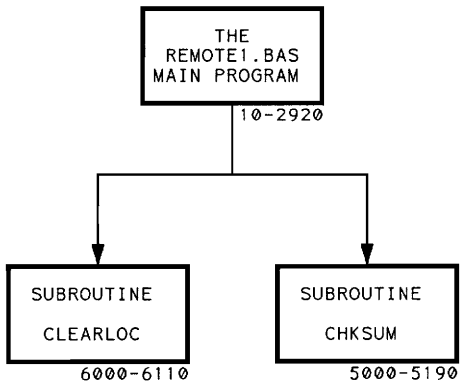


Figure 23 The Flow Chart of the Structure of the Example Program REMOTE1.BAS

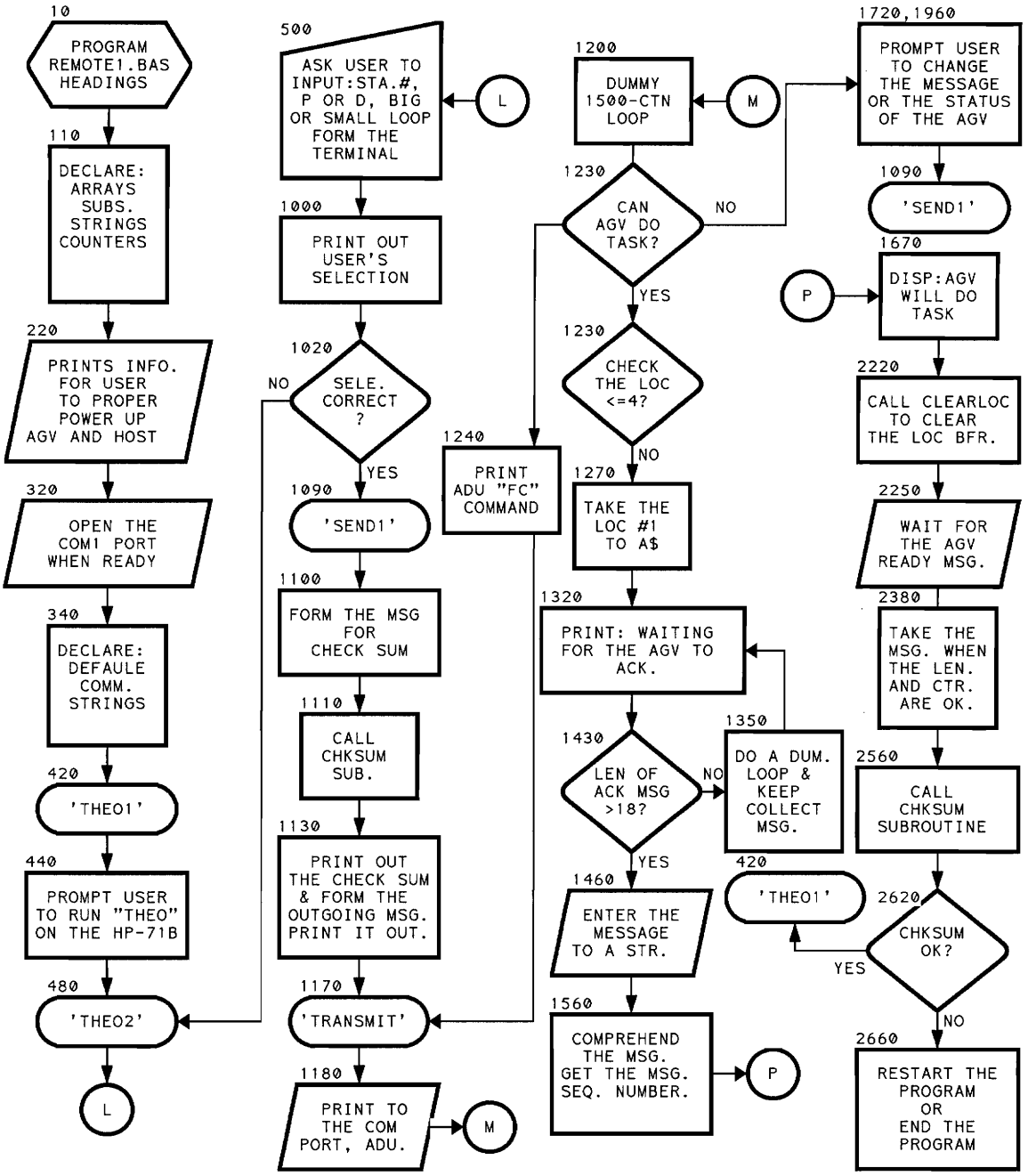


Figure 24 The Flow Chart of the REMOTE1.BAS Main Program

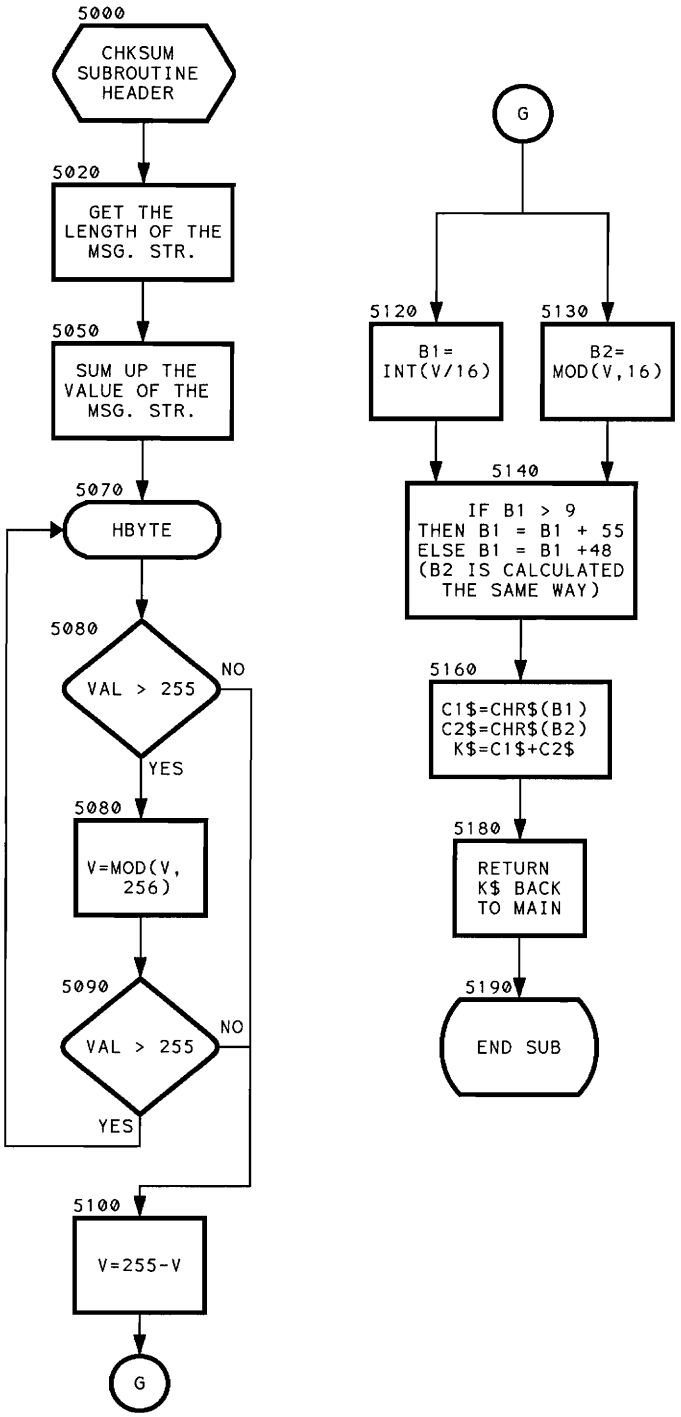


Figure 25 The Flow Chart of the CHKSUM Subroutine

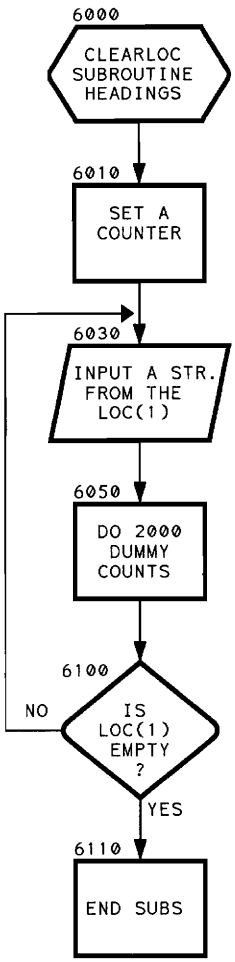


Figure 26 The Flow Chart of the CLEARLOC Subroutine

The example program of the remote computer -- REMOTE1.BAS

```
10    ' ************************************************************************
20    ' * VIRGINIA TECH.  DEPT. OF ISE
30    ' * THIS PROGRAM IS TO COMMUNICATE WITH THE ORBITOR
40    ' * BY USING THE REMOTE COMPUTER AND THE ADU.
50    ' * BY: ROBERTO F. LU    12/30/1993
60    ' * THIS PROGRAM SHALL BE RAN WITH THE "THEO" PROGRAM
70    ' * IN THE HP-71B COMPUTER OF THE AGV AT THE SAME TIME.
80    ' ************************************************************************
90
100   OPTION BASE 1                          'The baseline for arrays
110   DIM Q$(16), M(20), E(20)
120   DECLARE SUB CHKSUM (S$, K$)
130   DECLARE SUB CLEARLOC ()
140   X2$ = CHR$(13) + CHR$(10)              'X2$ IS THE CR AND THE LF
150   X3$ = CHR$(33)                         '33=!
160   CNT = 100                              'THE MESSAGE COUNTER
170   H = 1
180   L = 1
190   K = 2
200   J = 2
210   CLS
220   PRINT "PLEASE MAKE SURE THAT THE REMOTE COMPUTER AND
      THE ADU ARE "
230   PRINT "CONNECTED PROPERLY WITH THE SERIAL CABLE FROM
      THE ADU TO  "
240   PRINT "THE COM1 PORT OF THE REMOTE COMPUTER."
```

```
250  OPEN0:
260  PRINT
270  PRINT "DO NOT RUN THE AGV UNTIL THE REMOTE CMPUTER IS
     READY. "
280  PRINT "PLEASE MOVE THE AGV TO ITS HOME POSITION WHERE
     NEAR THE ADU."
290  INPUT "PRESS ENTER WHEN READY"; ETR$
300  IF ETR$ = "" THEN GOTO OPEN1 ELSE GOTO OPEN0
310  OPEN1:
320  OPEN "COM1:9600,N,8,1" FOR RANDOM AS #1      'Open a serial port
330
340  'DEFAULT COMMUNICATION STRING SETTINGS
350  STA$ = "300"                    'The ADU station, home station
360  E$ = "N"                        'No instruction given to the AGV
370  LOOP$ = "B"                     'Bigger loop
380  MOVE$ = "H"                     'P, D, or move to Home
390  DUMY$ = "AA"                    'Dummy string to fill blanks
400  PRINT
410
420  THEO1:
430  BEEP
440  PRINT "PLEASE RUN THE 'THEO' PROGRAM IN THE HP-71B
     COMPUTER NOW"
450  INPUT "PRESS ENTER WHEN READY"; ETR$
460  IF ETR$ = "" THEN GOTO THEO2 ELSE GOTO THEO1
470
480  THEO2:
490  PRINT
```

```
500   INPUT "PLEASE SELECT A STATION (1 OR 2)"; S$
510   PRINT "YOUR SELECTION IS: "; S$
520    IF S$ = "1" THEN
530     STA$ = "101"
540    ELSEIF S$ = "2" THEN
550     STA$ = "102"
560    ELSE
570     BEEP
580     PRINT
590     PRINT "WRONG SELECTION, ANSWER 1 OR 2"
600     GOTO THEO2
610    END IF
620   PORD1:
630   PRINT
640   INPUT "FOR PICK-UP, DELIVERY, OR STAY HOME (P/D/H)"; MOVE$
650   IF MOVE$ = "P" OR MOVE$ = "p" THEN
660    MOVEID$ = "PICK-UP"
670    PRINT "YOUR SELECTION IS: "; MOVE$
680   ELSEIF MOVE$ = "D" OR MOVE$ = "d" THEN
690    MOVEID$ = "DELIVERY"
700    PRINT "YOUR SELECTION IS: "; MOVE$
710   ELSEIF MOVE$ = "H" OR MOVE$ = "h" THEN
720    CLS
730    BEEP
740    PRINT "THANK YOU FOR USING VT AGV! BYE! BYE!"
750    END
760   ELSE
770    BEEP
```

```
780     PRINT
790     PRINT "WRONG SELECTION, ANSWER P OR D"
800     GOTO PORD1
810    END IF
820 '
830    LOOP1:
840    PRINT
850    INPUT "USING THE BIG OR THE SMALL LOOP (B OR S)"; LOOP$
860    IF LOOP$ = "B" OR LOOP$ = "b" THEN
870     LOOPID$ = "BIG"
880     PRINT "YOUR SELECTION IS: "; LOOP$
890    ELSEIF LOOP$ = "S" OR LOOP$ = "s" THEN
900     LOOPID$ = "SMALL"
910     PRINT "YOUR SELECTION IS: "; LOOP$
920    ELSE
930     BEEP
940     PRINT
950     PRINT "WRONG SELECTION, ANSWER B OR S"
960     GOTO LOOP1
970    END IF
980
990    PRINT
1000   PRINT "YOU HAVE SELECTED STATION #"; S$; " FOR "; MOVEID$;
1010   PRINT " USING THE "; LOOPID$; " LOOP."
1020   INPUT "IS EVERYTHING CORRECT (Y/N)"; AN$
1030   IF AN$ = "Y" OR ANS$ = "y" THEN
1040    E$ = "Y"
1050    GOTO SEND1
```

```
1060  ELSE GOTO THEO2
1070  END IF
1080 '
1090  SEND1:
1100  TR$ = MID$(STR$(CNT), 2, 3) + E$ + STA$ + MOVE$ + LOOP$ +
      DUMY$        'Make a string that can be used for check sum
1110  CALL CHKSUM(TR$, K$)          'Perform the check sum calculation
1120  PRINT
1130  PRINT "THE CHECK SUM OF THE HOST MESSAGE STRING IS: "; K$
1140  TRAN$ = "$" + TR$ + K$ + X2$   'Make the message string for comm.
1150  PRINT "MESSAGE FROM THE HOST TO THE ORBITOR IS: "; TRAN$
1160
1170  TRANSMIT:
1180  PRINT #1, TRAN$          'Transmit the message to the ADU
1190  N = 0                    'Dummy loop for the LOC(1) to have all chars
1200  FOR I = 1 TO 1500
1210  N = N + 1
1220  NEXT I
1230  IF LOC(1) <= 4 THEN      'Check the status of the LOC(1)
1240   PRINT #1, "FC" + X2$    'Tell the ADU do not feed back ack yet
1250   GOTO TRANSMIT
1260  ELSE
1270   A$ = INPUT$(LOC(1), #1)        'Get the outgoing message from LOC(1)
1280   PRINT "THE MESSAGE FROM THE HOST TO THE AGV IS:"; A$
1290  END IF
1300  CNT = CNT + 1
1310  IF CNT = 199 THEN CNT = 100
1320  PRINT "WAITING FOR THE AGV TO ACK"
```

```
1330  READ1:
1340  N = 0
1350  DO                          'Clear the LOC(1) and do a dummy count
1360     DMY$ = INPUT$(LOC(1), #1)
1370     M = 0
1380     FOR I = 1 TO 1000
1390     M = M + 1
1400     NEXT I
1410     PRINT N; DMY$;
1420    N = N + 1
1430   LOOP UNTIL LEN(DMY$) > 18          'Until the right length msg has in.
1440  READ2:
1450  IF LOC(1) < 18 THEN GOTO READ2
1460  ERR$ = INPUT$(LOC(1), #1)              'Take the msg from the HP-71B
1470  N = 0          'Do a dummy loop to provide time in taking the msg in str.
1480  FOR I = 1 TO 2000
1490  N = N + 1
1500  NEXT I
1510  'The format of the ERR$ = "@200KKKKKKKKKKKKbb"<CR><LF>
1520  IF L = 11 THEN L = 1       'Verify the msg sequence number
1530  K = K + 1
1540  L = L + 1
1550  IF K = 12 THEN K = 2
1560  E(K) = VAL(MID$(ERR$, 2, 3))
1570  IF E(K) = E(L) THEN GOTO SEND1
1580  PRINT ERR$
1590  BEEP
1600  I = VAL(MID$(ERR$, 2, 1))          'Determine what kind msg was recv'd
```

```
1610  SELECT CASE I
      CASE 1
1630      GOTO READ1
1640  CASE 2              ' AGV can do asked task, msg  sequence # 2xx.
1650      BEEP
1660      PRINT
1670      PRINT "AGV ACK OK.  AGV WILL PERFORM TASK."
1680      GOTO AGVMSG
1690  CASE 3              'AGV can not do asked task, load on, cannot pickup.
1700      BEEP            'Message sequence # 3xx
1710      PRINT
1720      PRINT "THERE IS A LOAD ON THE LIFT OF THE AGV, ";
1730      PRINT "CANNOT PICK-UP ANOTHER ONE!"
1740      PRINT "DO YOU WANT TO REMOVE THE LOAD OR CHANGE
          PICK-UP TO DELIVERY?"
1750  REDO1:
1760      INPUT "REMOVE (R) OR DELIVERY (D)"; BN$
1770      IF BN$ = "R" OR BN$ = "r" THEN
1780        PRINT
1790        PRINT "PLEASE REMOVE THE LOAD FROM THE LIFT OF THE
           AGV."
1800  CALBL1:
1810        INPUT "PRESS ENTER WHEN READY"; ETR$
1820        IF ETR$ = "" THEN GOTO CALBL2 ELSE GOTO CALBL1
1830  CALBL2:
1840        GOTO SEND1                          'TRANSMIT
1850      ELSEIF BN$ = "D" OR BN$ = "d" THEN
1860        MOVE$ = "D"
```

```
1870        GOTO SEND1
1880      ELSE
1890        PRINT "ENTER R OR D"
1900        GOTO REDO1
1910      END IF
1920 '
1930  CASE 4                    'AGV cannot do asked task, msg seq. # 4xx
1940      BEEP                  'There is no load on the lift for delivery
1950      PRINT
1960      PRINT "THERE IS NO LOAD ON THE LIFT OF THE AGV, ";
1970      PRINT "CANNOT DELIVER!"
1980      PRINT "DO YOU WANT TO PUT THE LOAD ON OR CHANGE TO
          PICK-UP?"
1990  REDO2:
2000      INPUT "LOADING (L) OR PICK-UP (P)"; CN$
2010      IF CN$ = "L" OR CN$ = "l" THEN
2020        PRINT
2030        PRINT "PLEASE PUT THE LOAD ONTO THE LIFT OF THE AGV."
2040  CALBL3:
2050        INPUT "PRESS ENTER WHEN READY"; ETR$
2060        IF ETR$ = "" THEN GOTO CALBL4 ELSE GOTO CALBL3
2070  CALBL4:
2080        GOTO SEND1                          'TRANSMIT
2090      ELSEIF CN$ = "P" OR CN$ = "p" THEN
2100        MOVE$ = "P"
2110        GOTO SEND1
2120      ELSE
2130        PRINT "ENTER L OR P"
```

```
2140        GOTO REDO2
2150      END IF
2160 '
2170  CASE ELSE
2180      GOTO READ1
2190  END SELECT
2200 '
2210  AGVMSG:             'After the AGV finished the task, prog. goes to here
2220  CALL CLEARLOC   'Clear the previous messages in the LOC(1)
2230  PRINT "WAITING ON MESSAGE FROM THE AGV TO START THE
      NEXT TASK."
2240  N = 0
2250  DO              'Keep looking for the right kind of message to come.
2260     DMY$ = INPUT$(LOC(1), #1)
2270      M = 0
2280      FOR I = 1 TO 1000
2290      M = M + 1
2300      NEXT I
2310     'PRINT N; DMY$;          'WHEN DEBUG, TURN THIS ON FOR INFO.
2320     N = N + 1
2330  LOOP UNTIL LEN(DMY$) > 14 AND LEN(DMY$) < 17
2340 '
2350  TAKING:
2360  IF LOC(1) < 14 OR LOC(1) > 17 THEN GOTO TAKING
2370
2380  MSG$ = INPUT$(LOC(1), #1)      'Take msg. that was sent from HP-71B
2390  'THE FORMAT OF THE MSG$ = "$200AGVREADY1A"<CR><LF>
2400  N = 0
```

```
2410  FOR I = 1 TO 2000
2420  N = N + 1
2430  NEXT I
2440  PRINT MSG$
2450  IF H = 11 THEN H = 1
2460  H = H + 1
2470  J = J + 1
2480  IF J = 12 THEN J = 2
2490  M(J) = VAL(MID$(MSG$, 2, 3))
2500  IF M(J) = M(H) THEN GOTO TAKING    'Verify the sequence # of msg.
2510  MS = MS + 1
2520  PRINT
2530  PRINT "THE MESSAGE FROM THE ORBITOR TO THE HOST IS: ";
      MSG$
2540  PRINT "THE CHECKSUM STRING IS: "; MID$(MSG$, 2, 11)
2550  C$ = MID$(MSG$, 2, 11)          'Do a checksum on the recv'd msg.
2560  CALL CHKSUM(C$, K$)
2570  CALL CLEARLOC                   'Clear the LOC(1) buffer again.
2580  PRINT
2590  PRINT "THE CHECKSUM FROM THE ORBITOR= "; MID$(MSG$, 13, 2)
2600  PRINT "THE HOST CALCULATED CHECKSUM = " + K$
2610  PRINT
2620  IF K$ = MID$(MSG$, 13, 2) THEN
2630  PRINT "PROPERLY RECEIVED MESSAGE #"; MID$(MSG$, 2, 3); "
      FROM THE AGV."
2640  GOTO THEO2              'Loop back to the front of the program
2650  ELSE
2660  PRINT "BAD CHECKSUM"
```

```
2670  PRINT
2680
2690  BADSUM:
2700  INPUT "DO YOU WANT TO CONTINUE? (Y/N)"; ANS$
2710  IF ANS$ = "Y" OR ANS$ = "y" THEN
2720      PRINT "STOP THE PROGRAM ON THE HP-71B BY PRESSING THE
          [ON] KEY"
2730      K = 1
2740      WHILE K = 1
2750       PRINT
2760       PRINT "PRESS ANY KEY ON THE REMOTE COMPUTER
           KEYBOARD WHEN READY"
2770       PRINT "THEN START HP-71B PROGRAM AGAIN BY PRESSING
           THE [RUN] KEY"
2780        DO
2790         A$ = INKEY$
2800         LOOP UNTIL A$ <> ""
2810         K = 0
2820       WEND
2830        GOTO THEO1
2840     ELSEIF ANS$ = "N" OR ANS$ = "n" THEN
2850      PRINT "PROGRAM TERMINATED"
2860      END
2870     ELSE
2880      PRINT "ANSWER Y OR N"
2890      GOTO BADSUM
2900     END IF
2910  END IF
```

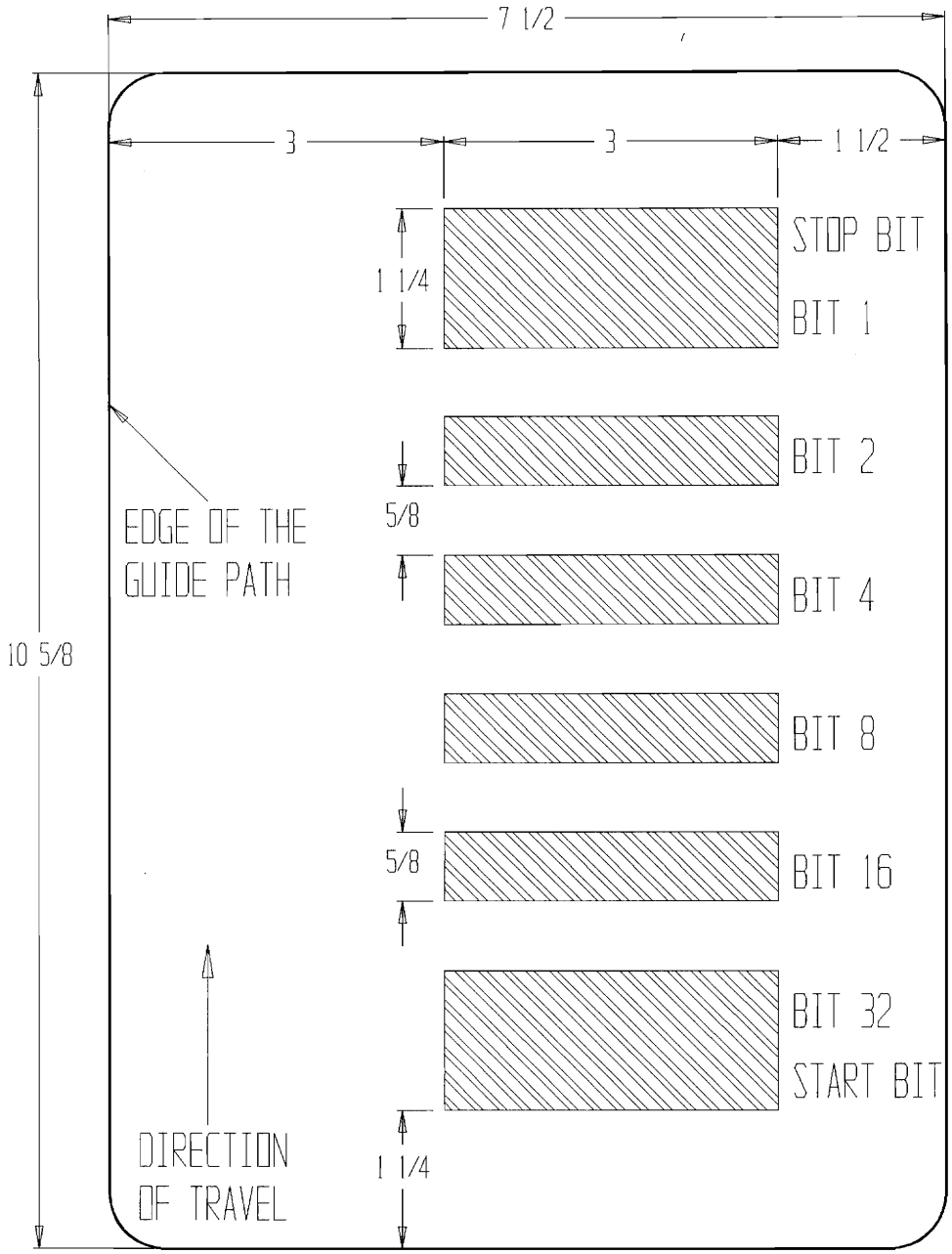
```
2920  END

5000  SUB CHKSUM (Q$, K$)          'Check sum subroutine
5010  DIM Q$(16)                   'Please ref. to the checksum sub. on the
5020  L = LEN(Q$)                  'HP-71B example program.
5030  V = 0
5040  FOR I = 1 TO L
5050  V = V + ASC(MID$(Q$, I, 1))
5060  NEXT I
5070  HBYTE:
5080  IF V > 255 THEN V = V MOD 256
5090  IF V > 255 THEN GOTO HBYTE
5100  V = 255 - V
5110  B1 = V / 16
5120  B1 = INT(B1)
5130  B2 = V MOD 16
5140  IF B1 > 9 THEN B1 = B1 + 55 ELSE B1 = B1 + 48
5150  IF B2 > 9 THEN B2 = B2 + 55 ELSE B2 = B2 + 48
5160  C1$ = CHR$(B1)
5170  C2$ = CHR$(B2)
5180  K$ = C1$ + C2$
5190  END SUB

6000  SUB CLEARLOC          'CLEARLOC subroutine
6010  N = 0
6020  DO
6030  DUMMY$ = INPUT$(LOC((1), #1)
6040  M = 0
```

```
6050  FOR I = 1 TO 2000
6060  M = M + 1
6070  NEXT I
6080  'PRINT "CLEAR LOC BUFFER"; DUMMY$
6090  N = N + 1
6100  LOOP UNTIL LOC(1) = 0
6110  END SUB
```

APPENDIX C

THE ICUTEST PROGRAM OF THE HP-71B COMPUTER

```
10 RESET HPIL @DESTROY ALL          ! HP-71B is controller, clear memory
20 ENDLINE                          ! Automatic <CR><LF>
30 OPTION BASE 1                    ! Set the base line for all the arrays
40 X2$=CHR$(13)&CHR$(10)            ! Chr$(13) is CR, Chr$(10) is LF
50 STD                              ! Standard display format for numbers
60 DISP "ICU TEST" @ WAIT 2
70 S$ = "FTxxxxx"&X2$               ! MSG string that will be sent to ICU
80 ON TIMER #1, 30 GOTO 'SEND'      ! Send a message every 30 seconds
90 'SEND':
100 OUTPUT :2;S$                    ! Sending the S$ message to ICU
110 'CHKBIT':  @T = T+ 1
120 S = BIT(MOD(SPOLL('RS232') DIV 256,256), 2)          ! The ICU buffer bit
130 DISP S; "WAITING"; T @ WAIT 1          ! Display the S and the counter
140 IF NOT S THEN 'CHKBIT'          ! Keep waiting for the Buffer to have a 1
150 ENTER :2;A$                     ! Enter the MSG to the HP-71B in A$
160 DISP "RESPONSE = "&A$ @WAIT 2       ! DISP A$ to verify that theICU
170 END                             ! is ok and end the program
```

APPENDIX D

THE TEMPLATE FOR THE FLOOR BAR CODE