TECHNICAL REFERENCE

MANUAL

TABLE OF CONTENTS

SYSTEM OVERVIEW 1

1 Introduction - 2
2 Major Components - 4

PROPULSION SYSTEM 10

3 Drive System and Base Assembly - 11
4 Power Distribution System - 14
5 Propulsion Control Computer - 18

SENSOR SYSTEM 23

6 Head Assembly - 24
7 Sensors - 28

VOICE INPUT/OUTPUT AND SOUND SYSTEM 40

8 VIOS Computer - 41
9 VIOS Operating System - 46

MAIN CONTROL SYSTEM 53

10 Main Computer - 54
11 Keyboard and Display - 89

LIFE OPERATING SYSTEM 99

12 Memory Organization - 101
13 System Monitor - 107
14 Living Mode Operating System - 130
15 Scheduler - 140
16 Navigation - 143

APPENDIX 151

List of Figures - 152
List of Tables - 153

SYSTEM OVERVIEW

SECTION I

SYSTEM OVERVIEW

CHAPTER 1

INTRODUCTION

This manual contains facts about GEMINI's construction, electronic and mechanical hardware, and some of its software. It is not a manual of methods and is not intended to teach you how to program in any of the languages available on the robot. It is intended to educate you about GEMINI and to provide a convenient reference source of factual information about the robot.

This REFERENCE MANUAL is divided into six major sections. Section 1, entitled System Overview, is divided into two chapters. Chapter 1 is an introduction which you are now reading. It briefly describes the six major sections of this manual. Chapter 2 provides you with a complete "picture" of the entire robot. This section includes a number of drawings to get you familiar with the physical layout of the robot's hardware.

Section 2, entitled Propulsion System, is divided into three chapters. Chapter 3 provides information concerning the base assembly and drive system. Chapter 4 describes the power distribution system and Chapter 5 describes the propulsion control computer.

Section 3, entitled Sensor System, is divided into two chapters. Chapter 6 provides information concerning the head assembly and its sensors. Chapter 7 describes the digital and analog sensors and their control. It also describes the sonar ranging sensors and their control and operation.

Section 4, entitled Voice Input/Output and Sound System (VIOS), describes the VIOS computer and is divided into two chapters. Chapter 8 describes the many aspects of the computer, and Chapter 9 the operating system.

Section 5, entitled Main Control System, is divided into two chapters. Chapter 10 describes the main computer and onboard peripherals. Chapter 11 describes the keyboard and liquid crystal display.

Section 6, entitled Life Operating System, describes the ROM based software and is divided into five chapters. Chapter 12 gives a detailed description of the memory organization. Chapter 13 provides details of the system MONITOR. Chapter 14 gives similar details of the living mode operating system. Chapter 15 describes the SCHEDULER program. Chapter 16 discusses the navigation system and important subroutines.

CHAPTER 2

MAJOR COMPONENTS

Figures 2.1, 2.2 and 2.3 show assembly sketches of the robot´s body and interior. Figure 2.1 shows an exploded view of the ABS, flame retardant plastic shells. These shells provide protection for the robot´s electronics and a smooth exterior surface to prevent the robot from "catching" objects while in motion. The shells purposely give the robot an overall conical shape. This shape is very functional from the standpoint of "snag-free" movement. Notice that the base is circular and that the pressure sensitive bumpers are located at the outer most circumference of the robot. This helps insure that the robot will make contact with an object, missed by its collision avoidance sonars, at its base first. The circular shape allows the robot to turn within its own diameter without snagging an object. The shells can be removed and installed quickly.

Figures 2.2 and 2.3 show two views of the robot and its major components. The interior was designed to provide easy access to all the electronics and room for the addition of peripherals and experimental apparatus. There are three main assemblies that make up the robot; the head assembly, the torso assembly and the base assembly.

The head assembly contains most of the navigational sensors. These include three head ranging sonars, the door edge reflector detector and the room beacon detector. The motion detector, light sensor and microphones are also located on the head. Notice that the navigation sensors are centrally located and at the highest point possible on the robot to give it an "adult´s view" of its world. The head has a stepper motor which can rotate the head at a relatively high speed with little power consumption. The head can rotate through approximately 359 degrees. All wiring passes through a central hole so there is no chance of entangling and breaking signal wires. There is plenty of space on the head for the addition of peripherals and experimental equipment.

The torso assembly is the most complicated part of the robot. Mounted on this lightweight aluminum frame are all of the computers, power supply, signal conditioning, collision avoidance sonars, and user interface devices. All of the electronics is exposed for ease of adjustment or replacement of parts. The main computer is centrally located so interconnecting cable runs are kept to a minimum. As with the head, there is plenty of room for the addition of experimental devices and peripherals.

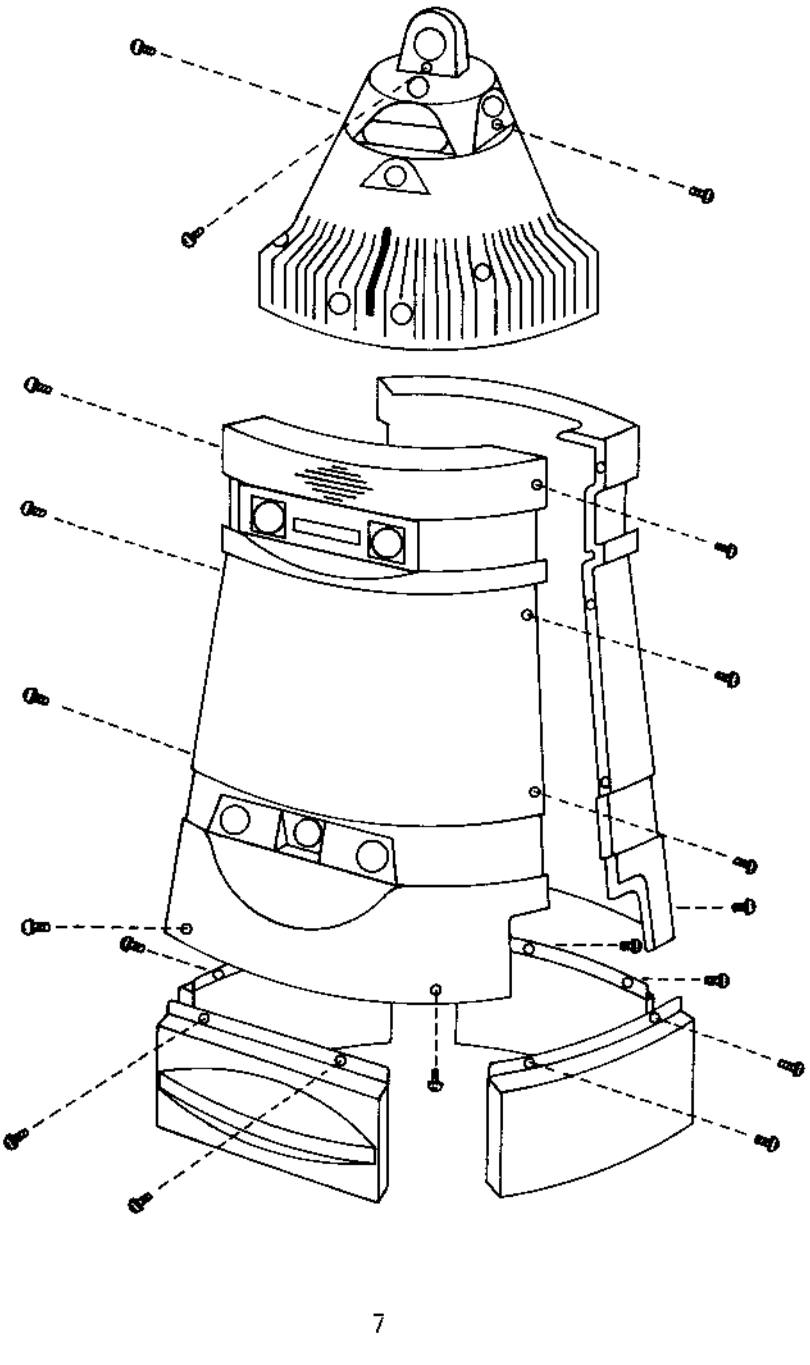
The base assembly contains the drive motors, three batteries, four wheels, frame and bumpers. Nearly 70 percent of the robot´s weight is contained in the base assembly. This insures that the robot´s center of gravity is very low, therefore making the robot difficult to overturn.

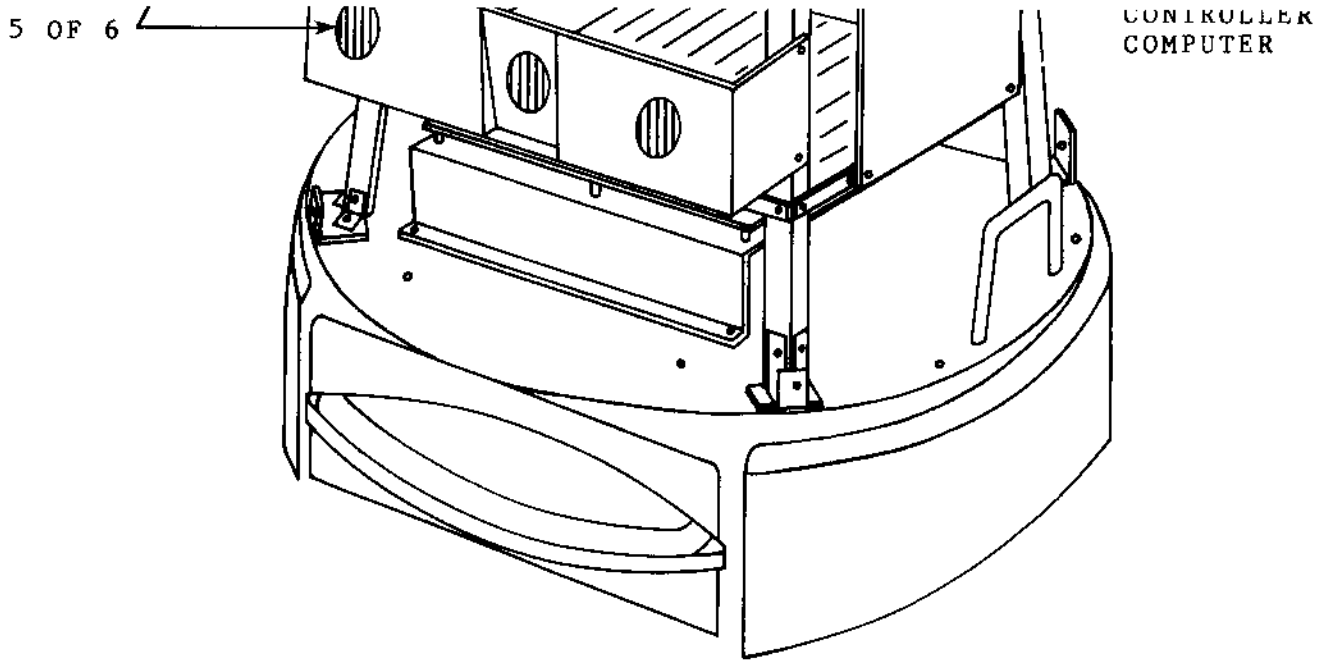
The three assemblies can be quickly dis-assembled. The torso mounts to the base on 8 shock-absorbing screws with hex nuts. The motor cable assembly easily disconnects allowing the torso to be removed from the base. This provides quick access for battery replacement, belt adjustment, etc.

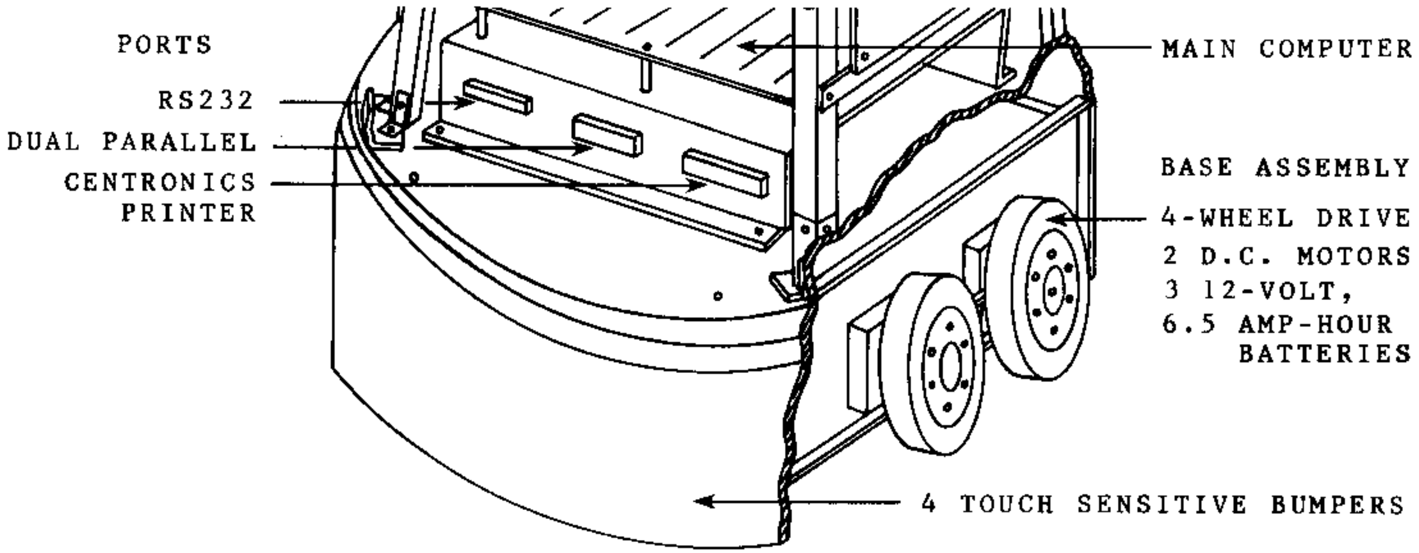
Disconnecting the head from the torso is not as simple, yet it is straightforward. The signal cable is easily disconnected. Then to remove the head, first locate the large brass gear on the underneath side of the head. Using the 1/16" Allen wrench supplied with your robot, loosen the two set screws and remove the gear. The head will lift right off.

FIGURE 2.1

EXTERIOR ABS PLASTIC SHELLS




5 OF 6
CONTROLLER
COMPUTER


PORTS
RS232
DUAL PARALLEL
CENTRONICS PRINTER
MAIN COMPUTER
BASE ASSEMBLY
4-WHEEL DRIVE
2 D.C. MOTORS
3 12-VOLT, 6.5 AMP-HOUR BATTERIES
4 TOUCH SENSITIVE BUMPERS

PROPULSION
SYSTEM

SECTION II

PROPULSION SYSTEM

CHAPTER 3

DRIVE SYSTEM AND BASE ASSEMBLY

The robot´s drive system has the ability to move over most terrain typically encountered in a home, school or office environment. The drive consists of four wheels driven by two independent DC motors. One motor controls the two wheels on the right side and the other controls the two wheels on the left side. The two wheels on either side are connected together with a toothed belt which provides a 4-wheel drive capability.

A schematic of the drive system is shown in Figure 3.1. The aluminum inner frame serves a number of purposes. It provides a mounting surface for the motors, and feed back sensors, it holds the batteries and it provides a mounting surface for the base ring and bottom torso plates. A schematic of the base assembly is shown in Figure 3.2. Four bumpers are attached to the outer ring. The front bumper is special in that it also holds the charger plates.

There are two sets of important sensors located in the base. Each motor shaft has attached to it a slotted disk which fits inside a U shaped IR emitter/detector device. When the motors are running, the IR device generates a pulse train that is used by the propulsion control computer to measure distance and control speed.

There are ribbon type switches near the top hinge of each bumper. Any pressure exerted against a bumper will close one of these switches. Signal wires from these switches are led to the propulsion control computer for monitoring.

FIGURE 3.1

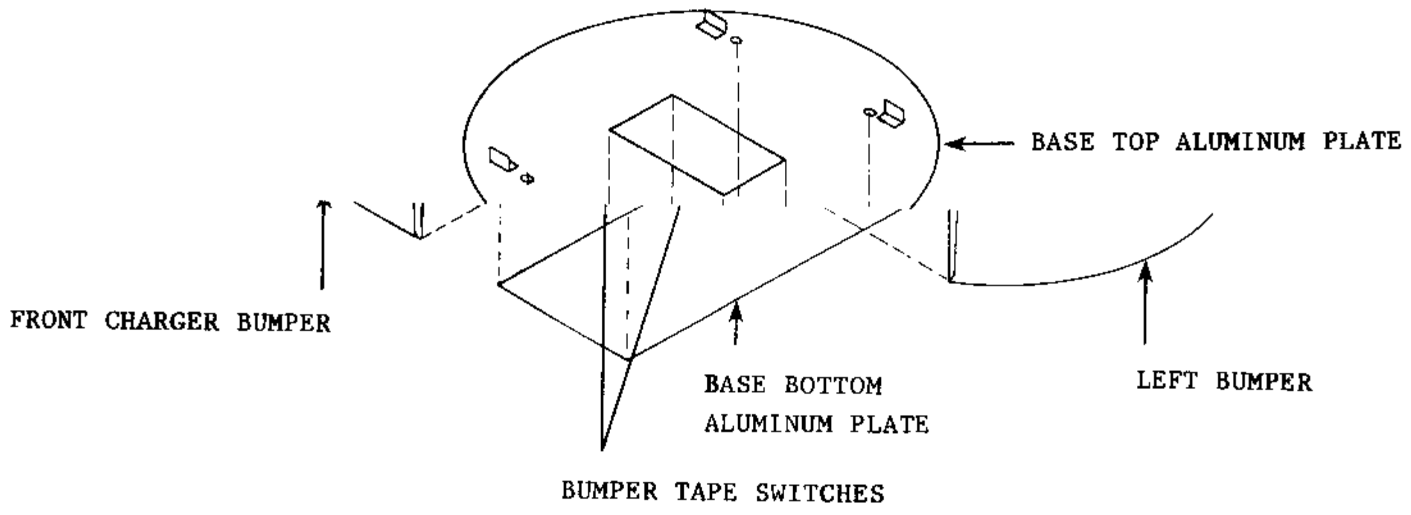
DRIVE SYSTEM SCHEMATIC

ALUMINUM INNER STRUCTURE

TOP VIEW

FIGURE 3.2

BASE ASSEMBLY SCHEMATIC



CHAPTER 4

POWER DISTRIBUTION SYSTEM

This system is comprised of the batteries, the power distribution board and the charger.

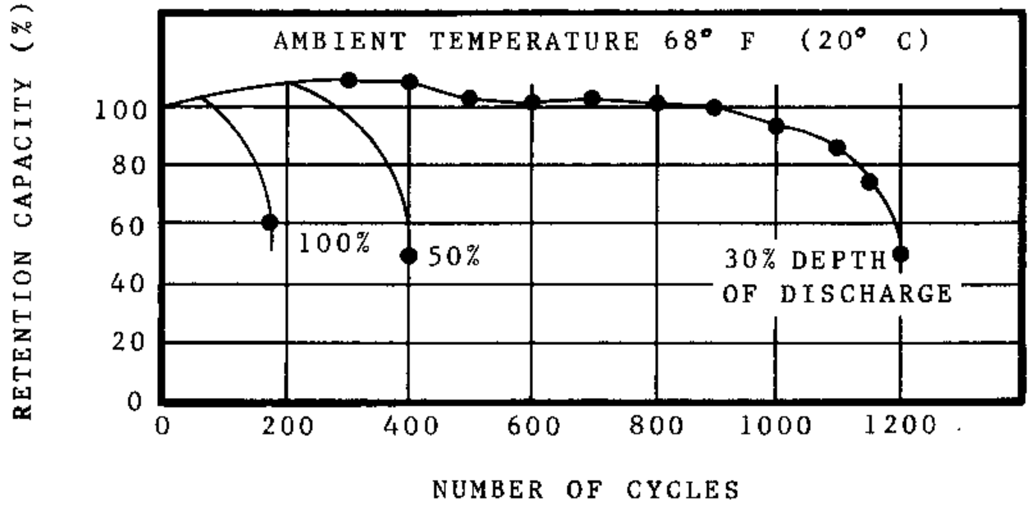
There are three 12-volt 6.5 amp-hour gel cell batteries. One battery is used to power the right motor and sonars, one battery is used to power the left motor and head stepper motor, and the other battery is used to provide power to all logic boards and the keyboard. The logic battery can be isolated from the motor batteries by opening a relay which ties their grounds together.

The charge capacity of a battery is a function of the number of times the battery is discharged and the depth of the discharge. A graph from the manufacturer's catalog is given in Figure 4.1 which shows this function. It can be seen from this figure that battery charge capacity is significantly affected by the depth of discharge. For this reason, it is best not to allow the robot to remain off its charger for long periods of time.

Current draw from the batteries depends on the duty cycle of the motors and the use of auxiliary computers and peripherals. Off charge operating times can therefore only be estimated. Factory measurements show that each motor draws about 350 ma of current when translating and about 700-1000 ma when rotating depending on the surface the robot is rotating on. Observations of the robot navigating indicate a relatively low duty cycle on the motors so 10-12 hours of off charge operations should be normal. CMOS logic has been used wherever possible on the computers and peripherals to reduce current drain

FIGURE 4.1

CHARGE/DISCHARGE CYCLE LIFE



on the batteries. A fully equipped main computer running at 2 MHz (full speed) draws about 100 ma. The voice input/output and sound computer draws around 50 ma when speaking and around 80 ma when using the NMOS sound generator chip. The propulsion computer draws around 50 ma. Such small amounts of current allow the logic battery to power the computers for long periods of time while off its charger.

The power distribution board performs several functions. It regulates the incoming DC voltage from the charger and supplies the charging voltage to the batteries. It also contains an efficient switching regulator which reduces the raw +12 volts to +5 volts for logic. The board also contains short circuit fuse protection for each battery, a relay to connect motor and logic battery grounds and an on-charge detector circuit. NOTE: logic ground is tied to the chassis of the robot.

The battery charger is especially designed to facilitate the robot´s docking itself to the charger. The charger is shaped so it can rest against a flat wall or in a corner. The flat wall location provides the robot with the greatest possible area of access and is preferred. The charger also sits on spring loaded legs. This allows the charger to move up and down and rotate in the vertical plane the amounts necessary to accommodate docking with the charger plates on the robot.

The charger contains a 110 volt 60 Hz to 18-22 volt 60 Hz step down transformer rated at 2 amps. The output of the transformer is full wave rectified and filtered to supply about 18 volts DC. The unit is fuse protected with a 2 amp "slow-blow" fuse to guard against short circuits. A red lamp is mounted on the top of the charger to indicate when it is powered.

WARNING: If the fuse blows, replace it only with a 2 amp "slow-blow" fuse. Also, keep pets and children away from this unit.

CHAPTER 5

PROPULSION CONTROL COMPUTER

ARCHITECTURE

The heart of the Propulsion Control Computer (PROCON) is a Rockwell R65C02 8-bit CMOS microprocessor (CPU) which operates at 1 MHz. It contains 2K of CMOS static RAM, 2K of CMOS ROM, two motor speed/distance controller chips, 18 bit input ports and 5 bit output ports.

The 64K address space of the CPU is only partially decoded. The hex addresses used and their functions are as follows:

HEX ADDRESS	USE
0000-07FF	2K RAM
0801	Bumper Input Port
0802	Baud Rate Switches/Joystick Controller Input Port
1000-1007	Motor Speed Control Chip Registers
1800-1807	Motor Distance Control Chip Registers
C000	Serial Input Data/Control Lines
D000	Serial Output Data/Control Lines
E000	Motor Direction and LED Power Control Lines
F000-FFFF	4K ROM (Jumper Selectable) or 2K ROM (F800-FFFF)

SPEED/DISTANCE CONTROL

There are two RCA CDP1878C Dual Counter-Timer chips used on the PROCON board to perform pulse width modulation, speed control and distance measurement. Each of these chips has two identical 16-bit programmable down counters and several control input and output lines. Each chip can be placed in one of five different operating modes.

The 1878 used for motor speed control is operated in Mode 5 which provides for a repetitive output with programmable duty cycle. The output lines of this chip are used to perform pulse width modulation speed control of the two motors. The clock to this chip comes from a 74HC4040 down counter which divides the 1 MHz CPU clock to a suitable value. The chip has interrupt capabilities and these are used on PROCON.

The 1878 used for distance measure is operated in Mode 1 which is the time out or count down mode. The clock for this chip is the stream of pulses which comes in from the slotted disk on the motor shafts. The chip is loaded with the distance in number of pulses to be counted and when these pulses have been counted, the chip interrupts the CPU.

SERIAL COMMUNICATIONS

Both a CMOS level serial port and an RS232C serial port are available on the PROCON board. In addition there is a 4-position dip switch which allows selection of the transmission baud rate.

The RS232C serial port (J3) is configured as Data Communication Equipment (DCE) and has the following pinouts.

RS232C SERIAL PORT PINOUTS

PIN	USE
1	GND
2	RD
3	TD
4	$\overline{\text{CTS}}$
5	$\overline{\text{RTS}}$
6	pulled high
7	GND
8	pulled high

The CMOS level serial port (J2) is configured to match the corresponding ports on the main computer. The pinouts for this port are as follows:

CMOS SERIAL PORT PINOUTS

PIN	USE
1	GND
2	RD
3	TD
4	$\overline{\text{CTS}}$
5	$\overline{\text{RTS}}$
6	+5V IN
7	GND
8	Not Used
9	Enable ($\overline{\text{ON}}$/OFF)
10	GND

The dip switches are used as follows:

SW4	SW3	SW2	SW1	BAUD RATE
NU	NU	ON	ON	300 bits per sec
NU	NU	OFF	ON	600 bits per sec
NU	NU	ON	OFF	1200 bits per sec
NU	NU	OFF	OFF	2400 bits per sec

Serial communications consists of one start bit, 7 data bits and 2 stop bits. All communications to and from PROCON are with normal ASCII characters (high bit clear).

SOFTWARE

The PROCON operating system uses a command oriented structure. Commands are issued to PROCON and the computer acknowledges the host when the command has been executed. The control program makes use of interrupts which allows constant monitoring of the serial input lines for additional commands even while in the process of executing a command.

The commands are divided into three groups. Group I commands give PROCON direction, speed and distance commands. Group II commands give speed adjustment commands. Group III are special function commands. The format and allowable values for Group I, II, and III commands are given in Table 5.1.

Source codes of the PROCON software are available from ARCTEC SYSTEMS.

TABLE 5.1

PROCON COMMANDS

GROUP I - MOVEMENT COMMANDS

COMMAND	RESPONSE	FUNCTION
0 XYYYY 0	0 or Bumper 1-8	Move robot FORWARD at speed (0-F) X, for distance (16 bit counter) YYYY
1 XYYYY 0	0 or Bumper 1-8	BACKWARDS-same as above
2 XYYYY 0	0 or Bumper 1-8	LEFT-same

COMMAND	RESPONSE	FUNCTION
3 XYYYY 0	0 or Bumper 1-8	RIGHT-same
B XYYYY 0	0 or Bumper 1-8	LEFT, ONE MOTOR -same
C XYYYY 0	0 or Bumper 1-8	RIGHT, ONE MOTOR -same

GROUP II - CONTROL COMMANDS

COMMAND	RESPONSE	FUNCTION
40	0	STOP by ramping down
50	!	RESET stop immediately
60	0	ENABLE BUMPERS
A0	0	CLEAR OFFSET
D0	XXXXXXXX	RETURN LED COUNT
E0	0 or Bumper 1-8	RETURN BUMPER

F0	000 or MLR	SYSTEM TEST
		M=Memory bad
		L=Left motor bad
		R=Right motor bad

GROUP III - CHANGE SPEED OR DIRECTION COMMANDS

COMMAND	RESPONSE	FUNCTION
7 X 0	0	CHANGE SPEED to X(0-F)
8 X 0	0	OFFSET LEFT Motor by X(0-F)
9 X 0	0	OFFSET RIGHT Motor by X(0-F)

SENSOR SYSTEM

SECTION III

SENSOR SYSTEM

CHAPTER 6

HEAD ASSEMBLY

The head assembly is shown without the shell in Figure 6.1. At the base of the assembly are two aluminum plates. The lower one is the shoulder plate, and is part of the torso. The upper one, the head sensor plate, is part of the head assembly. The shoulder plate provides a mounting surface for the head stepper motor and bearings. The head sensor plate is used to mount sensors and anchor the plastic head shell.

The head can be rotated through a 359 degree arc. Continuous rotation of the head is prevented by a thin fiber mechanical stop attached to the shoulder plate and a metal stud attached to the head sensor plate. The fiber stop has copper plates on both sides which are electrically insulated from the shoulder plate. Each copper plate is attached to one end of a resistor that is pulled high (to logic +5V). A signal wire runs from each plate to BITS 6 and 7 of input bit port $E050. Counter-clockwise rotation of the head to the stop will force BIT 7 low and clockwise rotation to the stop will force BIT 6 low. These are referred to, respectively, as the left stop and right stop bits. The left stop position is used as the zero position of the head.

The head drive motor is a 12 volt 0-60 rpm Airpax stepper motor. It has a 4.2:1 built in gear box. The external gear provides another 6:1 gear reduction resulting in a 0-0.31 radian/sec head speed and 0.33 degree steps.

The head motor windings are each attached at one end to +12 volts and at the other to ground through a switching transistor. The +12 V supply is taken from the left motor battery. The base of each switching transistor is driven by a bit from bit output port $E061 through an optocoupler which isolates +5 logic from the motor battery. Bits 4, 5, 6, and 7 of $E061 are drive Phase 3, 2, 1 and 0, respectively, winding. The switching transistors, optocouplers and associated resistors and blocking diodes are housed on the Sonar, Sensor and Stepper board.

Machine language routines are contained in the System Monitor (See Section V, Chapter 14) to control the stepping sequence, speed and direction of the motor. Subroutine ONESTP ($E478) is the basis of all of the routines. This routine will step the motor one physical step (0.33 degree) in the direction specified by BIT 0 of the memory location $50 (0=CW, 1=CCW). Five physical steps are used to define one logical step (1.5 degree) and most programs in the robot use 1.5 degree as a single step. Subroutine TURNHD ($E47E) will step the head one logical step in a specified direction, at a specified speed while monitoring the Room Beacon and Door Edge detector bits (see Chapter 7) and looking for contact with either side of the mechanical stop. This routine also updates the variable which holds the current head position, CURHD ($056A).

There are 240 (decimal) logical steps ($00 through $EF). Position $00 is at the left stop, $78 is straight ahead or center position and $EF is at or near the right stop. Subroutine POSHD ($E47B) will position the head at the location contained in the A-Register on entry. It will do this at the speed specified by the contents of MSPSTP ($0567). This routine will also call subroutine INITHD ($E46F) to initialize the head at the left stop if it has not been previously initialized. The subroutine POSHD should be used by machine language programmers to move the head.

NOTE: The head does not use feedback to determine its current position. Consequently, if the head has been initialized and it is forced from its current head position, the routines will lose track of where the head is. The stepper motor is turned off (using POSHD) after a head movement is complete in order to conserve power. The head can thus be relatively easily moved.

There are several BASIC commands which give direct control over the head motor. INITHD will initialize the head and rotate it to the left stop. ROTO I will rotate the head to the decimal position I (0 to 239). ROT (DIR, STEPS) will rotate the head the number of STEPS in the DIRection specified.

FIGURE 6.1

HEAD ASSEMBLY

FRONT VIEW

STOP

FIBER
STOP

TORSO HARNESS OPENING

CHAPTER 7

SENSORS

Sensors on the robot provide the three onboard computers with information concerning the robot´s internal and external environment. Without this information it would be impossible for the robot to navigate, re-charge its batteries or perform tasks. All sensor information must be presented to one of the computers in digital form so the computers can process the information and take appropriate actions as required. The main computer is the eventual recipient of all sensor information. However, where the Propulsion Computer and Voice Input/Output Computer are involved, only results of information collected from sensors attached to these computers is presented to the Main Computer.

A considerable amount of signal conditioning is required in order to present sensor information in digital form to a computer. On GEMINI, there is a board referred to as the SSS Board (Sonar, Sensor and Stepper), which performs most of the signal conditioning functions. This board, the sensors and certain peripheral chips on the computers all comprise the robot´s SENSOR SYSTEM.

BATTERY VOLTAGE MONITORING AND CHARGE CONTROL

The charge level of the robot´s batteries are essential to its life. The robot must be able to monitor these levels and determine if it is on its charger or not. Each of the three battery voltages are routed to voltage dividers on the SSS board. These voltages are then passed through a differential amplifier and onto the 16 channel, analog to digital converter (ADC) on the Main Computer. The battery voltages are assigned to channels on the ADC as follows:

Battery	Channel Number
Left Motor	8
Right Motor	9
Logic	A

There is a subroutine in the System Monitor (See Section V) called ADCONV located at $E481 which converts this analog voltage into an 8-bit digital number. On entry, the Y-REG of the CPU contains the channel number. On exits, the A-REG contains the 8-bit conversion. This 8-bit number can be converted to actual voltage using the following formula

BATTERY VOLTAGE = CONVERSION RESULT (DECIMAL)/256
x REFERENCE VOLTAGE

The reference voltage for the ADC is +5V logic which usually runs around 5.12 volts. The battery voltages can also be read directly in volts using the BASIC command BAT(I) where I=0=left motor battery, I=1=right motor battery and I=2=logic battery.

Voltage outputs from the three differential amplifiers on the SSS board are passed on to comparators for use in generating a critical battery non-maskable interrupt (NMI) of the Main Computer's microprocessor. Any one of the three batteries can generate an interrupt. The interrupt voltage level can also be adjusted by potentiometer R107 on the SSS board. This has been set at the factory to give interrupts starting at around 10.5 volts (10.5/3 volts at Pin 6 of U2B on the SSS board).

When the voltage of any battery drops below this set point value, it resets the NMI flip/flop on the Main Computer and forces an NMI. There is an NMI service routine starting at $0853 which counts the number of NMI´s that have occurred since the counter ($0829) was last reset and compares it to a limit ($082A). If the limit (default is 20 NMI´s) has not been exceeded, the NMI flip/flop is set and the service routine returns from the interrupt. If the limit is exceeded, then a routine in Living Mode Operating System (See Section VI) is called. This routine compares each battery voltage to a critical value (BATCRT-$082B) and if any are found below this level, the robot issues a warning to the user and gives the user 1 minute to respond. If the user does not respond, the robot will attempt to return to its charger. If the user responds by pressing a function key, then BATCRT is set to zero, the NMI counter is set to zero, the limit is moved to its maximum value ($FF), and the NMI flip/flop is set. Under these conditions, the robot cannot give another warning to the user. You should therefore heed the first warning or you may endanger the life of the batteries.

Also associated with the batteries is an "on-charge" detector circuit located on the Power Supply Board. When the robot goes onto its charger, current flows through the diode side of opto coupler U1 which turns on the transistor and pulls the ON CHARGE line low. This lights the on charge LED at the back of the robot and provides an active low signal at BIT 0 of the bit input port located at address $E050 on the Main Computer. The user can read this port location with either a machine language program or a BASIC program. The BASIC command CHARG will return a decimal 00 if on charge or a 01 if off charge.

Grounds between the logic and motor batteries are separated so voltage spikes during motor operations or sonar ranging do not interfere with the logic supply voltage. The ground can be connected or disconnected by closing and opening a relay on the Power Supply Board. This is accomplished under program control by changing the state of BIT 2 of the bit output port located at address $E060 on the Main Computer. The grounds are connected when this bit is low. The bit can be set with either a machine language program or the BASIC command GNDON. There is also a BASIC command GNDOFF that will disconnect the grounds. The grounds should be connected when on the charger, otherwise, only the logic battery will be charged. In general, the grounds should be disconnected when the robot is off its charger. Many routines in the System Monitor will automatically disconnect grounds when sonar ranging or driving the motors. The Living Loop will automatically connect grounds if the robot is on its charger.

SONAR RANGING AND CONTROL

Measurement of distance from the robot to an object is essential for navigation and collision avoidance. GEMINI is equipped with nine (9) ultrasonic sonar transducers that are located at strategic points on the robot. There are three (3) sonar transducers located on the head so they can be accurately positioned by the head stepper motor over a 359 degree arc. There are five (5) sonar transducers located on the front of the robot and one (1) sonar transducer located at the rear.

In general, the head sonars are used for navigation and the body sensors for collision avoidance;however, the navigation routines (See Section VI) use all of the information for both navigation and collision avoidance. The location of these sensors is very important. Figure 7.1

shows the areas covered by the sonars at various distances from the robot. The Polaroid Sonar transducers have a detection beam width of approximately 15-20 degrees.

Only one sonar transducer can be used to obtain ranging information at a time. Multiple firing of sonar transducers leads to false ranging information and is therefore prevented on GEMINI by multiplexing. The multiplexing technique used on GEMINI involves switching of the transducers so only one sonar ranging module is needed. Relays on the SSS board accomplish the switching in and out of a transducer. Bits 0 through 3 of the output bit port at address $E062 (See Section V), are used to select a transducer. The following bit patterns select the desired transducer.

BIT				SONAR TRANSDUCER
3	2	1	0	SELECTED
0	0	0	0	Head
0	0	0	1	Left side of head
0	0	1	0	Right side of head
0	0	1	1	Left top front
0	1	0	0	Right top front
0	1	0	1	Left bottom front
0	1	1	0	Right bottom front
0	1	1	1	Front center
1	0	0	0	Back
1	0	0	1	Spare (None Connected)

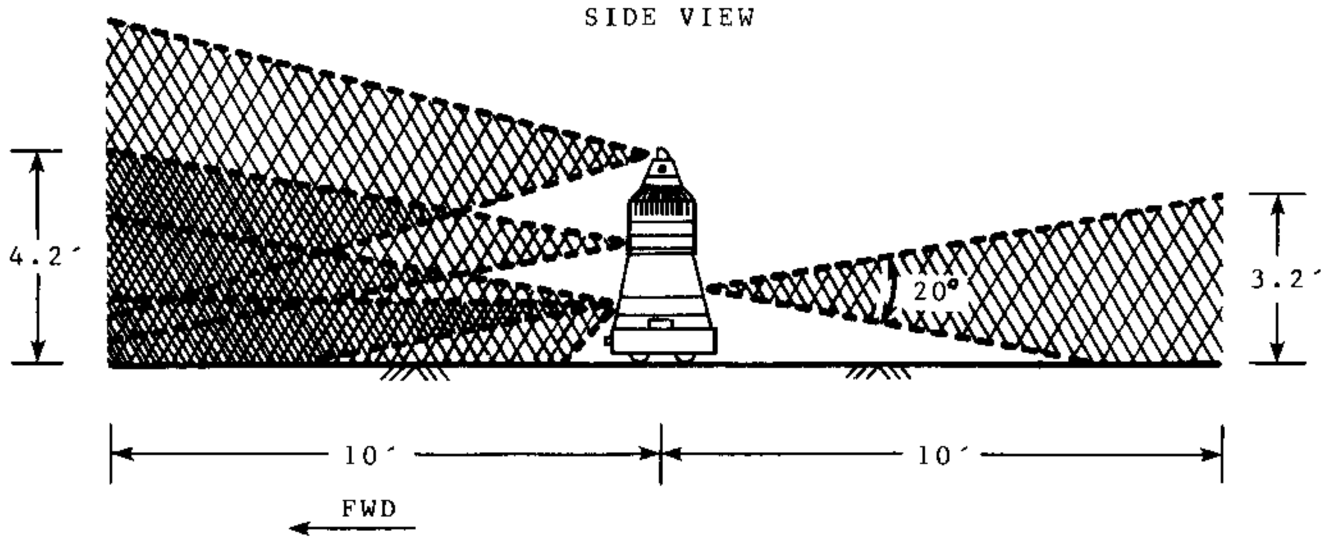
WARNING: No other bit combination is allowed. If you activate the sonar ranging module without a sonar transducer attached, you can destroy the module.

Power to the ranging module is controlled by BIT 3 of the bit output port at address $E060. When this bit is high, power to the ranging module is turned on.

Ranges are obtained from a selected sonar transducer by measuring the time interval between firing a sonic burst and receiving the return echo. It takes approximately 1775 us for sound to travel 2 ft. Thus, for a range of 1 ft (round trip distance 2 ft), 1775 us will elaspe. Accurate measurements of such small time intervals are handled by a 16 bit timer chip on the Main Computer (See Section V). This timer is connected to the sonar ranging module on the SSS board. The timer is first loaded with a hexidecimal FFFF. Then BIT 5 of output bit port $E060 is brought low. This causes the ranging module to pulse the sonar transducer and start the count down of the timer. When the sonar transducer receives the echo, the ranging module generates another signal which turns off the timer. The timer can then be read to determine the elapsed time and hence range. BIT 5 of $E060 must be brought high for about 100 ms before firing the sonar again.

There are machine language routines in the System Monitor to handle obtaining ranges from any sonar. These are described in detail in Section VI. There is also a BASIC command which will return the range in feet from a designated sonar. The command is RANGE(I) where I is the desired sonar as follows:

I	SONAR
0	Head
1	Right top front
2	Right bottom front
3	Left top front
4	Left bottom front
5	Right side of head
6	Left side of head
7	Front center
8	Back


SIDE VIEW
4.2´
3.2´
20°
10´
10´
FWD

ROOM BEACON AND DOOR EDGE REFLECTOR SENSING AND CONTROL

There are two sensors mounted on the head that are an important part of the robot's navigation system. Due to the large beam width of the sonars, it is not possible to detect door openings (edges) from sonar ranging unless the robot is within about 4-5 feet of a door. To aid the robot in locating doors, semi-circular blocks covered with reflective tape capable of reflecting infrared radiation are placed near each door. The robot's Infrared Door Edge Detector can then be used to obtain relative bearings of the reflector.

This sensor is both an IR transmitter and receiver. The unit transmits a signal, which if received, will cause BIT 2 of input bit port at address E050 on the Main Computer to go high. The range of this sensor is about 20 ft (40 ft round trip). Reflections can sometimes be obtained off bright metallic surfaces, glass and mirrors.

The robot must be able to determine what room it is in. Room Beacons are installed in each room which produce a 100% modulated square wave of IR radiation at about 40 KHz. The period of this square wave is selected at each beacon with a 4-position dip switch. The switch positions and approximate period are as follows:

SW4	SW3	SW2	SW1	PERIOD(MS)	ROOM CODE	
0	0	0	0	Not Allowed	N/A	
0	0	0	1	7.26	01	(1)
0	0	1	0	13.07	02	(2)
0	0	1	1	20.33	03	(3)
0	1	0	0	28.31	04	(4)
0	1	0	1	35.57	05	(5)
0	1	1	0	41.38	06	(6)
0	1	1	1	48.64	07	(7)

SW4	SW3	SW2	SW1	PERIOD(MS)	ROOM CODE	
1	0	0	0	59.53	08	(8)
1	0	0	1	66.79	09	(9)
1	0	1	0	72.60	0A	(10)
1	0	1	1	79.86	0B	(11)
1	1	0	0	87.85	0C	(12)
1	1	0	1	95.11	0D	(13)
1	1	1	0	100.91	DE	(14)
1	1	1	1	108.17	DF	(15)

0=OFF, 1=ON

The Room Beacon Detector receives the radiated signals through a narrow slot, and by using a parabolic mirror concentrates the energy onto a photo diode. The signal from the photo diode is then amplified and passed to a phase lock loop circuit which is adjusted to detect a 40 KHz signal. When a 40 KHz signal is present, BIT 1 of input bit port at address $E050 will go high. The narrow slot and location of the sensor in the head allow angular information regarding the location of the beacon to also be obtained.

There are machine language programs in the Navigation program (See Section VI) which will determine the period of the incoming signal at $E050 BIT 2 and look up the correct room code. Entry points for these routines are given in Section VI.

Both the Door Edge and Room Edge Beacon sensors can be turned on and off by BIT 4 of output bit port $E060. These sensors are enabled when BIT 4 is high.

SMOKE DETECTOR SENSING AND CONTROL

This optional sensor is located in the head and is a typical ion chamber type particle detector.

It is activated (power up) by bringing BIT 6 of the output bit port at address $E060 on the Main Computer high. When there is no smoke present, BIT 3 of the input bit port $E050 will be low. If smoke is present, BIT 3 of $E050 will oscillate with a period of about 2.5 ms. The BASIC command SMOKE will return the number 0 if there is no smoke. Due to the oscillation it is necessary to poll this location in a BASIC loop to determine if smoke is present.

MOTION DETECTOR SENSING AND CONTROL

Two ultrasonic transducers are mounted on the front of the head for the purpose of detecting motion in the vicinity of the robot. The circuitry for this sensor is located on the SSS board. Power to this circuit is controlled by bringing BIT 6 of the output bit port $E060 (same as the Smoke Detector) high. If the sensor detects motion, BIT 4 of $E050 will go high. When this sensor is on, it will interfere with the ultrasonic ranging module and cause false ranging. The BASIC command MOTION will return the number 0 if there is no motion or 1 if there is motion.

SOUND LEVEL SENSORS

There are three small microphones located around the head which provide voice input and sound level sensing. Signals from these microphones are routed to the SSS board where they are amplified and added together to obtain one signal. This preamplified signal is then passed on to the Voice Input/Output computer for use in voice recognition and to the analog to digital converter for sound level conversion. The sound level is assigned to channel B of the ADC and can be read by the machine language program ADCONV ($E481) or by the BASIC command SOUND. SOUND will return a value from 0 to 255.

LIGHT LEVEL SENSOR

A cadmium sulfide cell is mounted on top of the head just in front of the head sonar transducer. This signal is conditioned on the SSS Board and then passed to channel D of the ADC. The BASIC command LIGHT will return a value from 0 to 255.

TEMPERATURE SENSOR

An LM334 temperature sensor and conditioning circuitry are located on the SSS Board. The output conditioned signal is passed to channel E of the ADC. The BASIC command TEMP will return the temperature in degrees celsius.

PRESSURE SENSOR

This optional sensor is attached to the SSS board and contains its own signal conditioning. The output from this board is passed to channel C of the ADC. The BASIC command PRESSURE will return the pressure in millibars.

MAIN MEMORY BATTERY BACKUP VOLTAGE SENSOR

A line from the top of the battery leads directly to channel F of the ADC.

REMOTE COMMUNICATIONS SENSING AND CONTROL

The optional Remote Communication System (RCS) computer comes with a modem board and radio transceiver that attach to the robot. Power to this board is enabled by bringing BIT 7 of output bit port $E060 high. Should a carrier be detected from the RCS when this board is enabled, BIT 5 of input bit port $E050 will go low. When the robot is in its Living Loop, this forces a jump to the RCS communications program where communication between the robot and RCS are handled. This is further explained in the users manual that accompanies the RCS.

REAL TIME CLOCK

The robot can obtain the real time by reading the real time clock on the Main computer. This is fully explained in Section V. BASIC command TIME$ and DATE$ will return the time and date respectively as a string variable.

RANDOM NUMBER SENSOR

The Main computer contains a hardware random number generator. This is fully explained in Section V. The BASIC function RND(0) will return a random number between 0 and 255.

BUMPERS

The bumpers are under control of the Propulsion computer and their use is fully explained in Section II.

VIOS SYSTEM

SECTION IV

VOICE INPUT/OUTPUT AND SOUND SYSTEM

CHAPTER 8

VIOS COMPUTER

SYSTEM OVERVIEW

The Voice Input/Output and Sound System consists of a microcomputer (VIOS), three microphones with mixing preamplifier,and two 8 ohm speakers. The system handles speech recognition, speech synthesis and sound synthesis.

Voice input from the three microphones is preamplified and added together on the SSS Board. This preamplifier has a fixed gain of 100. This signal is decoupled and passed to a two stage audio amplifier on the VIOS computer. The first stage of the amplifier has a fixed gain of 100. The second has a variable gain of 0 to 50. This gain is controlled by variable resistor R50. The output of the two stage amplifier is passed to a low pass filter whose gain is approximately 2. From there the signal is sent to two circuits which generate a word available (WRDAV) and a zero crossing detect (ZCDET) digital signal for use in voice recognition.

The speech synthesizer has a 1/2 watt amplifier attached to its output. This amplifier directly drives one of the two 8 ohm speakers. Control circuitry on the VIOS computer allows both the chip and its amplifier to be powered on and off as needed.

The complex sound generator also has a 1/2 watt amplifier attached to its output. This amplifier directly drives the other 8 ohm speaker. Control circuitry on the VIOS computer allows both the chip and its amplifier to be powered on and off as needed.

ARCHITECTURE

The heart of the VIOS computer is a Rockwell R65C02 8-bit CMOS microprocessor (CPU) which operates at 1 MHz. It contains 16K of battery backed CMOS static RAM, 32K of CMOS ROM, a third generation SSI263 speech synthesizer, an AY-3-8913 PSG complex sound generator, 24 bit input/output lines, two 1/2 watt audio amplifiers, RS232C and CMOS level serial communication ports, and various power control and support chips. The VIOS computer is an extremely powerful, multifunction computer with extensive software in ROM.

The 64K address space of the CPU is utilized as follows:

HEX ADDRESS	USE
0000-1FFF	8K CMOS RAM (Battery Backed)
2000-3FFF	8K CMOS RAM (Battery Backed)
4000-4003	82C55 PIA I/O REGISTERS
6000-6004	SSI263 SPEECH SYNTHESIZER REGISTERS
8000-9FFF	8K CMOS ROM
A000-BFFF	8K CMOS ROM
C000-DFFF	8K CMOS ROM
E000-FFFF	8K CMOS ROM

The 24 bit I/O lines are used as follows:

BIT	DIR	USE
PA0	I	RD
PA1	I	$\overline{\text{CTS}}$
PA2	I	SW0
PA3	I	SW1
PA4	I	SW2
PA5	I	SW3
PA6	I	ZCDET
PA7	I	$\overline{\text{WRDAV}}$
PB0	I/O	PSG DATA DA0
PB1	I/O	PSG DATA DA1
PB2	I/O	PSG DATA DA2
PB3	I/O	PSG DATA DA3
PB4	I/O	PSG DATA DA4
PB5	I/O	PSG DATA DA5
PB6	I/O	PSG DATA DA6
PB7	I/O	PSG DATA DA7
PC0	O	PSG BCI
PC1	O	PSG BDIR
PC2	O	PSG RESET
PC3	O	SOUND ON/$\overline{\text{OFF}}$
PC4	N/C	N/C
PC5	O	SPEECH ON/$\overline{\text{OFF}}$
PC6	O	TD
PC7	O	$\overline{\text{RTS}}$

SERIAL COMMUNICATIONS

Both a CMOS level serial port and an RS232C serial port are available on the VIOS computer. In addition there is a 4-position dip switch which allows selection of transmission baud rates.

The RS232C serial port (J6) is configured as Data Communications Equipment (DCE) and has the following pinout:

RS232C SERIAL PORT PINOUTS

PIN	USE
1	GND
2	RD
3	TD
4	$\overline{\text{CTS}}$
5	$\overline{\text{RTS}}$
6	PULLED HIGH
7	GND
8	PULLED HIGH

The CMOS level serial port (J1) is configured to match the corresponding ports on the Main Computer. The pinout for this port are as follows:

CMOS SERIAL PORT PINOUTS

PIN	USE
1	GND
2	RD
3	TD
4	$\overline{\text{CTS}}$
5	$\overline{\text{RTS}}$
6	+5V IN
7	GND
8	+12 IN
9	ENABLE (ON/$\overline{\text{OFF}}$)
10	GND

The 4-position dip switch is used as follows:

SW3	SW2	SW1	USE
0	0	0	MALE VOICE 1
0	0	1	MALE VOICE 2
0	1	0	VILLAIN´S VOICE
0	1	1	CHILD´S VOICE
1	0	0	ALIEN´S VOICE
1	0	1	MALE VOICE 2
1	1	0	MALE VOICE 2
1	1	1	MALE VOICE 2

SW4	USE
0	300 BITS PER SEC
1	1200 BITS PER SEC

CHAPTER 9

VIOS OPERATING SYSTEM

The VIOS operating system uses a command oriented structure. Commands are issued to VIOS and the computer acknowledges the host when the command has been executed.

On power up, VIOS performs certain cold start chores and then transmits a prompt (ASCII 21 "!") to the host computer. It then waits for a mode selection command. The mode selection commands and their functions are as follows:

COMMAND	CHARACTER	MEANING
11 (DC1)	CTRL-Q	Enter Voice Recognition Mode
12 (DC2)	CTRL-R	Enter Speech Generation Mode
13 (DC3)	CTRL-S	Enter Sound Generation Mode
14 (DC4)	CTRL-T	Perform Self Test

VIOS does not acknowledge these select commands. Once in a mode, the host may ask VIOS to return to the mode select state by transmitting an ASCII 09 (EOT) or a CTRL-I from a keyboard. VIOS will then respond with its prompt character.

VOICE RECOGNITION MODE

VIOS has a 256 word/utterance recognition capacity. The recognizer is an isolated word/phrase, speaker dependent, language independent recognizer with adaptive capabilities. These words are divided into 16 groups with 16 words in each group. Following are the commands, VIOS responses and meanings while operating in this mode. All commands are in ASCII.

COMMAND	RESPONSE	FUNCTION
0X	ERROR CODE	Cold start training of group X ($0 \leq X \leq F$)
1	ERROR CODE	Enter words for speaker adaptation pass
2	ERROR CODE	Process data from speaker adaptation pass
3X	ERROR CODE	Enter words for training pass one; group X
4XY	ERROR CODE	Enter word Y for training pass two; group X
5X	WORD NO, ERROR CODE	Recognize word in group X. An "I" (ASCII $49) will be returned as word number, if an invalid match was made.
6XY	ERROR CODE	Retrain second pass word Y in group X
7	ERROR CODE	Keep running mean of correctly recognized words
8XY	ERROR CODE	Receive acknowledge word Y in group X
9	ERROR CODE	Update running mean of previously recognized word
A	ERROR CODE	Update running mean of currently recognized word

ERROR CODE	MEANING
0	NO ERROR
1	VOREC ERROR (Try Again)
2	SPEECH BUFFER OVERFLOW
3	SPEAKER ADAPTATION PASS COMPLETE

Training of the voice recognizer is accomplished as follows. First the group to be trained is initialized by issuing the cold start command (0) followed by the group number (0-F). The

VIOS computer will respond with 0. Next the speaker adaptation phase is conducted by saying each word/phase in the vocabulary once until either all words are spoken or the computer indicates "enough data". The "enter words for speaker adaptation pass", command (1), must be given for each spoken word. After the word is spoken, the computer will respond with the error code.

At the end of the speaker adaptation phase,the "process data from speaker adaptation pass", command (2) must be given. The computer will respond with 0.

The individual word/phrase training is accomplished in two passes. In the first pass the command 3X, where X is the group number, is given prior to speaking each word. The computer will respond with the error code. In the second pass the command 4XY, where X is the group number and Y is the word number in this group, is given prior to speaking the word/phase. During the training passes it is necessary to repeat the words/phrases in the exact same sequence. At the completion of this phase, the training is complete for that group. The process is repeated for the remaining groups. Once trained, the computer will retain its training even when powered down as long as the RAM backup battery is functioning.

Speech recognition is accomplished by issuing the command 5X, where X is the group number of a previously trained group. The computer will respond with the word number recognized in this group and the error code.

The computer also has a limited ability to adapt or "learn" if this feature is selected. To select this feature, two commands are issued. First the Command 7 is issued to tell the computer to start maintaining a running mean of correctly recognized words. Second, the Command 8XY is issued where X is the group number and Y is the "acknowledgement" word number in this group. The group must be previously trained as described above.

Once this has been accomplished, the adaptation or learning feature is invoked after two consecutive uses of the 5X command by issuing the commands 9 and A. For example:

COMMAND ISSUED	WORD SPOKEN	COMPUTER RESPONSE
5X	"LEFT"	Y0 (where Y="LEFT"
5X	"YES"	Y0 (where Y="YES" = acknowledgement word)
9	N/A	0
A	N/A	0

The computer will average in the word patterns for LEFT and YES to its current word patterns for these words.

SPEECH SYNTHESIS MODE

There are two options available once this mode has been entered (CTRL-R). The first option allows the selection of a speaking voice already built into the computer and the second option selects text-to-speech operations.

The select speaking voice option is elected by issuing the command 0X where:

X	VOICE
0	MALE VOICE 1
1	MALE VOICE 2
2	VILLAIN´S VOICE
3	CHILD´S VOICE
4	ALIEN´S VOICE

The select text-to-speech option is selected by issuing the command 1 followed by ASCII text to be spoken and a carriage return ($0D).

After completing a command while in this mode, the computer will issue the ASCII character $3E or ">".

SOUND SYNTHESIS MODE

There are three options available once this mode has been entered (CTRL-S). The options are as follows:

COMMAND PREFIX	USE
0XY	DO A CANNED SOUND
1XY	PLAY A CANNED SONG
2X	ENTER MUSIC EDITOR

For commands 0 and 1, the parameters indicate the sound to make or the song to play as follows:

COMMAND	XY	SOUND/SONG
0	00	CLOCK
	01	MOVING ROBOT
	02	GUNSHOT
	03	OCEAN
	04	SILENCE
	05	TRAIN (CHOO-CHOO TYPE)
	06	HELICOPTER
	07	EXPLOSION
	08	PUMP
	09	POWER GENERATOR
	10	LASER
	11	SIREN
1	00	STAR SPANGLED BANNER (with vocals)
	01	HAPPY BIRTHDAY (with vocals)

COMMAND	XY	SOUND/SONG
	02	HAPPY ANNIVERSARY (with vocals)
	03	JINGLE BELLS (with vocals)
	04	EASTER PARADE
	05	AULD LANG SYNE (with vocals)
	06	WHEN IRISH EYES ARE SMILING
	07	STAR WARS
	08	ROBOTS ARE A MAN´S BEST FRIEND (with vocals)
	09	MUSIC BOX DANCER
	10	THE ENTERTAINER
	11	DAISY, DAISY (with vocals)
	12	RAINDROPS KEEP FALLING ON MY HEAD (with vocals)
	13	HAVAH NAGILAH
	14	O CANADA (with vocals)

Option 2 (enter music editor) provides three additional options. These options are as follows:

COMMAND PREFIX	USE
20	DOWNLOAD A SONG
21	PLAY DOWNLOADED SONG
22	CHANGE TEMPO OF DOWN LOADED SONG

Details on the use of these commands and the parameters which must follow are detailed in the MUSIC EDITOR manual.

When a command is completed while in the Sound Synthesis Mode, the computer will issue the ASCII character $3E or ">".

SELF TEST MODE

When this mode is entered the computer performs a self test and sends back a four character report (WXYZ) followed by the prompt (!). The report and its meaning are as follows:

CHARACTER	MEANING
W	0 - RAM OK
	1 - RAM CHIP 1 BAD
	2 - RAM CHIP 2 BAD
X	0 - ROM OK
	1 - ROM CHIP 1 BAD
	2 - ROM CHIP 2 BAD
	3 - ROM CHIP 3 BAD
	4 - ROM CHIP 4 BAD
Y	0 - SPEECH CHIP AND AMPLIFIER OK
	1 - SPEECH CHIP OR AMPLIFIER BAD
Z	0 - SOUND CHIP AND AMPLIFIER OK
	1 - SOUND CHIP OR AMPLIFIER BAD

Speakers and microphones must be attached to the computer for this test, otherwise the computer will report SPEECH CHIP and SOUND CHIP bad.

MAIN CONTROL SYSTEM

SECTION V

MAIN CONTROL SYSTEM

CHAPTER 10

MAIN COMPUTER

INTRODUCTION

The main control system on the robot consists of the MAIN COMPUTER, LIQUID CRYSTAL DISPLAY and the KEYBOARD. The MAIN COMPUTER is a very powerful, fully CMOS microcomputer with many peripherals on board. This computer is described in this chapter.

User interface to this computer and the auxiliary computers is through the KEYBOARD* and LIQUID CRYSTAL DISPLAY. Both of these I/O devices are described in Chapter 11.

MAIN COMPUTER ARCHITECTURE

The heart of the main computer is a Rockwell R65C02, an 8-bit CMOS microprocessor (CPU). This CPU is a very low power version of the popular 6502 and has the highest expanded instruction set of all CMOS 6502's. The address and data lines of the CPU are fully buffered. The CPU is also configured to allow direct memory access (DMA) by boards in the expansion slots. The CPU operates at a clock frequency of either 1 or 2 MHz and may be changed "on the fly" under software control. This feature allows the CPU to run at high speed when running calculation type programs and at low speed, when accessing "slow" peripheral chips or executing wait loops. Since the CPU's power consumption is proportional to speed, this helps conserve power.

---

* The RS232C port may be used for input in lieu of the keyboard.

The clock circuit consists of a crystal controlled oscillator, a 12 stage divider, and a CPU clock switching circuit. The oscillator frequency is 4 MHz and is divided down by the 12 stage divider for use of the CPU and the keyboard peripheral interface chip.

The 64K address space of the CPU is set up to provide 56K of static CMOS RAM, 7K of CMOS ROM and 1K of I/O space as follows:

HEX ADDRESS	MEMORY ALLOCATIONS	USE
0000-DFFF	56K	RAM (battery backed)
E000-E3FF	1K	I/O
E400-FFFF	7K	ROM

The RAM consists of seven 8K static CMOS RAM chips and the ROM consists of one 8K CMOS ROM chip, 1K of which is not used. Only low power static RAM chips (with "LP" designation) with an access time of 120 ns or less may be used.

The memory located at C000-DFFF is set up to be bank switched. A peripheral card, referred to as the ROM EXPANSION CARD, contains seven 8K sockets (to house either 8K RAM or ROM chips) and a 3 to 8 decoder controlled by three output bits from an I/O space peripheral chip. When this card is in place, any one of the chips on this card may be selected and written to (only if the card contains a RAM chip) or read from. This feature provides an additional 56K of addressable RAM or ROM. A ROM EXPANSION CARD is provided with GEMINI and contains the artificial intelligence software in ROM, which provides the robot with "life". Bank switching routines are contained in the "monitor" ROM which allow programs to jump back and forth between ROMS and access the RAM at this address. These routines are described in Section VI.

Utilization of the I/O space is summarized below, and followed by a detailed description of each.

HEX ADDRESS	MEMORY ALLOCATIONS	USE
E000-E00F	16	LIQUID CRYSTAL DISPLAY (LCD)
E010-E017	8	REAL TIME CLOCK (RTC)
E018-E01F	8	DUAL 16-BIT COUNTER/ TIMER
E020-E02F	16	SERIAL PORTS
E030-E03F	16	PARALLEL PORTS
E040-E04F	16	ANALOG TO DIGITAL CONVERTER (ADC)
E050-E05F	16	BIT INPUT PORTS
E060-E06F	16	BIT OUTPUT PORTS
E070-E073	4	RANDOM NUMBER GENERATOR
E074-E077	4	KEYBOARD
E078-E07B	4	BELL
E07C-E07F	4	NMI RESET
E080-E09F	32	SLOT 1 DEVICES
E0A0-E0BF	32	SLOT 2 DEVICES
E0C0-E0DF	32	SLOT 3 DEVICES
E0E0-E0FF	32	SLOT 4 DEVICES
E100-E17F	128	SLOT 1 I/O
E180-E1FF	128	SLOT 2 I/O
E200-E27F	128	SLOT 3 I/O
E280-E2FF	128	SLOT 4 I/O
E300-E3FF	256	NOT USED

REAL TIME CLOCK

The RTC is an RCA CDP1879C CMOS clock chip. It functions as a time-of-day/calendar with an alarm capability that can be set for combinations of seconds, minutes and hours and which has a capability to cause an IRQ type interrupt on the CPU. It can also be set up to create one of 15 square-wave signals on Pin 39 of each of the peripheral back-plane slots. The period of this square wave is variable from 488.2 us to 1 day and can be used to interrupt the CPU on a negative transition if desired. The RTC also has leap year handling abilities.

Routines for setting the clock, reading the clock, setting the clock alarm and turning off the clock alarm are contained in the monitor and are described in Section VI. Also, there are several zero page RAM locations that are used to store RTC data and to vector interrupts from the chip´s two sources.

To set the clock using these routines, zero page memory locations are set as follows:

HEX ADDRESS	USE
25	SECONDS
26	MINUTES
27	HOURS
28	DAY
29	MONTH
2A	YEAR,LOW (units and tens)
2B	YEAR,HIGH (hundreds and thousands)

using BCD data. Then subroutine SETCLK ($E442) is called.

To read the clock, call subroutine GETCLK ($E445) and then read the above memory locations. The time will be given in BCD. It should be noted that the RTC does not actually maintain years, but through routines in the monitor, a 10,000 year clock is implemented with leap year handling.

The RTC alarm may be set by loading the above memory locations with the desired alarm time and calling subroutine SETALM ($E448). When the time matches the alarm time, the CPU will be interrupted (provided the interrupt request (IRQ) is enabled) and a jump indirect through memory locations $18,19 will be taken. (That is, the program will jump to the address contained in these memory locations.) The monitor sets these locations to the "return from interrupt routine", RETINT (Monitor Jump Table Entry E406), on power up so in order to use the alarm function, memory locations $18,19 must be set to your routine.

In a similar manner if the square wave output referred to above is used to interrupt the CPU, then a jump indirect through memory locations $1A,1B will be taken. These locations must also be set to your routine prior to setting up the RTC for this type of interrupt.

You can use the square wave interrupt capabilities of the RTC by storing one of the following bytes in the RTC's status/control register located at $E017.

HEX DATA BYTE	CLOCK INTERRUPT PERIOD
06	disable
16	488.2 us
26	976.5 us
36	1953.1 us
46	3906.2 us
56	7812.5 us
66	15.625 ms
76	31.25 ms
86	62.5 ms
96	125 ms
A6	250 ms
B6	500 ms
C6	1 sec
D6	1 min
E6	1 hour
F6	1 day

For example, to set up a 1 minute interrupt, store the address of MNIRQ for the below routine in $1A,1B.

```
MNIRQ   SMB   TMTSK,ROSTAT    set bit TMTSK in
                              byte ROSTAT (zero
                              page)
        JSR   SRCLK           reset RTC for 1
                              min, clear interrupt
                              bits on RTC
        JMP   $E406           this is the monitor´s
                              return from interrupt
                              routine
SRCLK   JSR   $E622           shift CPU clock
                              to low speed
        LDA   $D6             set 1 min clock
                              interrupt period
        STA   $E017
        JMP   $E632           resume speed
```

To use this routine

```
        RMB    TMTSK,RDSTAT    clear bit TMTSK
                               in byte ROSTAT (zero
                               page)
        JSR    SRCLK           set RTC for 1 min
                               period
        CLI                    allow interrupts
LOOP    BBR    TMTSK,ROSTAT,   stay here until
               LOOP            interrupt
        BRK                    break on interrupt
```

DUAL 16-BIT COUNTER/TIMER

There is an RCA CDP 1878C Dual Counter-Timer chip with two 16-bit programmable down counters on the main computer that is very versatile and powerful. Each half of this chip can be used simultaneously in one of five different modes as follows:

MODE	FUNCTION	APPLI-CATION
1 - Timeout	Outputs change when clock decrements counter to "0"	Event Counter
2 - Timeout Strobe	One clockwide output pulse when clock decrements counter to "0"	Trigger Pulse
3 - Gate-Controlled One Shot	Output change when clock decrements counter to "0". Retriggerable	Time-Delay Generator
4 - Rate Generator	Repetitive clockwide output pulse	Time-Base Generator
5 - Variable-Duty Cycle	Repetitive output with programmed duty cycle	Motor Control

One half of this chip is used to measure the time it takes a selected sonar pulse to travel the round trip distance to a target. There is a subroutine in the monitor (See Monitor Jump Table Entry at E480) which may be helpful in learning how to use this chip in Mode 1. Also the propulsion computer uses two of these chips. One is used in Mode 1 to count distance pulses and the other in Mode 5 to accomplish pulse-width modulation speed control of the propulsion motors. By studying the routines in the PROCON ROM, you can further learn how to use this chip. You should also obtain a copy of the RCA spec sheet in order to fully understand how to use this powerful chip.

One half of the counter-timer is available for your use and the control lines are brought out to Port J7 for that purpose. There is also a jumper, J22, to allow you to select either the CPU clock (BΦ2) or an external clock source. The chip also has CPU interrupt capabilities but there is no service routine in the monitor. If you use this chip to interrupt the CPU, you must provide your own service routine. This can be accomplished by changing memory locations $14,15 to point to your service routine. After servicing the request, you should jump to the monitor return-from-IRQ routine at $E406 so that saved registers will be restored.

Although the other half of this chip is devoted to the sonar system, it too may be used as a precision 16-bit counter-timer when not using the sonars.

SERIAL PORTS

There are five (5) CMOS level serial ports and one (1) RS232 level serial port on the main computer. Their port numbers and use follows:

PHYSICAL PORT NUMBER	LOGICAL PORT NUMBER	USE
J10	0	Propulsion Computer
J11	1	Voice Input/Output and Sound Computer
J12	4	Expansion Computer-1
J13	5	Expansion Computer-2
J14	3	External RS232
J15	2	Remote Communication System Computer (RCS)

Each port (except J14) contains a 10 pin header with the following lines

PIN NUMBER	SYMBOL	USE	DIRECTION
1	GND	Ground	N/A
2	TD	Transmit Data	out
3	RD	Receive Data	in
4	RTS	Request to Send	out
5	CTS	Clear to Send	in
6	5V SUPPLY	+5 Supply	out
7	GND	Ground	N/A
8	+12V SUPPLY	+12 Supply	out
9	EN	Enable	out
10	GND	Ground	N/A

The serial transmission rate on the serial ports are set by the dip switch on the main computer. Switch 4 of the dip switch controls the baud rate on the CMOS level ports. When this switch is OFF the baud rate is 300 and when ON it is 1200. This rate applies to all CMOS ports. Switch 2 and 3 control the baud rate on the RS232C serial port as follows:

SWITCH 2	SWITCH 3	BAUD RATE
0	0	300
0	1	600
1	0	1200
1	1	2400

Serial communications on all of these ports are implemented by one 82C55A ($E020-E02F) and software serial data transmit/receive routines are included in the monitor for each port. There is also a bit in the output bit port ($E060-E06F) for each CMOS level serial port for use in enabling/disabling the associated auxiliary computer connected to this port. The following table summarizes the information associated with the CMOS level serial ports. For each character I/O routine, the received character is in the accumulator and the character to send is the accumulator.

PORT NAME	LOGICAL PORT NUMBER	ENABLE ADDRESS	ENABLE BIT	MONITOR I/O ROUTINE NAME	MONITOR I/O ROUTINE ADDRESS
PROCON	0	E060	0	PRCIN/ PRCOUT	E4E0/ E4D6
VIOS	1	E060	1	VIOSIN/ VIOSOT	E4F2/ E4E8

PORT NAME	LOGICAL PORT NUMBER	ENABLE ADDRESS	ENABLE BIT	MONITOR I/O ROUTINE NAME	MONITOR I/O ROUTINE ADDRESS
RCS	2	E060	7	RFMIN/ RFMOUT	E50F/ E4FA
EXP1	4	E061	6	EXP1IN/ EXP1OT	E540/ E536
EXP2	5	E061	5	EXP2IN/ EXP2OT	E552/ E548

In addition to the character I/O routines, there is a routine which allows for polling a port to see if a character is ready to be sent from an auxiliary computer. This routine, POLPRT ($E469), polls the port in the accumulator and returns with a hexidecimal 00 if there is no character or the character in the accumulator.

The procedure to use in communicating with an auxiliary computer using the CMOS level serial ports is given in the following example for enabling VIOS.

```
ENVIOS     LDA #$02              Enable VIOS by
                                 making "enable"
                                 high
           TSB $E060
           LDA #$01
           JSR WAIT.A.SECS       A routine to wait
                                 "A" seconds for
                                 VIOS power on
                                 reset
           LDA #$01
           JSR $E469             Try to get prompt
                                 (using POLPRT)
           CMP #'!
```

```
          BEQ.1               Branch if got it
          SEC                 Flag no answer
          RTS
.1        CLC                 Flag got prompt
          RTS
```

A call to this routine will enable VIOS and return with the carry clear if VIOS answered with its prompt, otherwise the carry will be set. If all was well, then communications via VIOSIN and VIOSOT can take place following the expectations written into the software of the auxiliary computer.

The RS232C level serial port has additional software features built into the monitor which makes it attractive for communications with other devices equipped with RS232 serial ports. Ordinarily character input data to the main computer comes from the keyboard and character output data goes to the LCD. However, both character input and character output are vectored to those monitor routines through zero page locations which contain the address of the input and output character handlers. The zero page locations are $1E,1F (CHARIN) for input and $1C,1D (CHROUT) for output. During a power on reset, the monitor checks switch 1 of the dip switch to determine if there is a keyboard (SW1=off) or a terminal (SW1=on) attached to the RS232C port. If the keyboard is attached, then CHARIN is set to point to RDKEY ($E460) which is the keyboard input character routine, otherwise it is set to EXTIN ($E52E). Similarly, if the keyboard is attached then CHROUT is set to LCDOUT ($E45D) which sends characters to the LCD only. If the keyboard is not attached, then LCDOUT sends characters to both the LCD and the external serial port via EXTOUT ($E524).

There is also a monitor command "C" which calls a special program that will allow upload and download of data between the robot and another computer via the RS232C port. This program only requires that the connected terminal have a terminal program that is capable of uploading and downloading data files in ASCII format.

To use this program, the user connects a terminal to the RS232C port, enters the Monitor and types "C". On the screen of the computer will appear

```
* COMMAND?
!
```

The exclamation mark is the prompt. There are two commands to receive or send data to the robot. To send data to the robot, the command is

```
RXXXX,YYYY <CR>
```

This command tells the robot to receive YYYY bytes of data, starting at XXXX in his memory. The robot will respond

```
READY TO RECEIVE
!
```

At this point the user would get his file and transfer it to the robot. After receipt of the file, the robot will return with ROBOT READY FOR COMMAND and the prompt (!).

It is also possible to request the robot to send data to the user´s computer. In this case the letter S is substituted for R and the robot´s response will be

```
                        READY TO SEND
                        !
```

A carriage return must then be given to initiate uploading of the file.

PARALLEL PORTS

PRINTER

Two types of parallel data transfer ports are implemented using a 65SC21 CMOS Peripheral Interface Adapter (PIA). One is a Centronics style printer port, J8, and the other is a dual, high speed, data transfer port, J2.

The pinout for the printer port is given below.

CENTRONICS PRINTER PORT

	STROBE	1	14	NC
DATA	D0	2	15	GND
	D1	3	16	GND
	D2	4	17	GND
	D3	5	18	GND
	D4	6	19	GND
	D5	7	20	GND
	D6	8	21	GND
	D7	9	22	GND
	ACK	10	23	GND
	NC	11	24	GND
	NC	12	25	NC
	NC	13	26	NC

There is a routine in the Monitor, PRTOUT ($E46C) to handle output to a printer. This routine can be used by changing CHROUT to point to this routine and then calling the standard OUTCHR ($E41B) routine. Since all ROM based routines in GEMINI use OUTCHR for output, all output from any of these programs will then be passed to the printer. There is also a monitor command (P) which will set up the vector to the printer port.

DUAL PARALLEL

The dual parallel port is a special and important addition to GEMINI. It provides a means for rapidly transferring data between the robot and a development computer equipped with a similar dual parallel chip. A parallel "communicator" card for the Apple II series computers and the IBM PC series computers is available from ARCTEC SYSTEMS. This equipment provides the capability to rapidly develop software for the robot.

The monitor "X" command enters a set of routines in the monitor that will handle the up/downloading using commands similar to those described for serial upload/download via the RS232C port.

ANALOG TO DIGITAL CONVERTER (ADC)

There is a National Semiconductor (ADC0817) CMOS 16-channel, 8-bit ADC on the main computer. This converter uses successive approximation as the conversion technique and will do a conversion in less than 100us. Its unadjusted total error is ± 1 LSB. Analog signals supplied to this chip MUST be single ended and MUST NOT fall outside the range $-0.1 \leq V_{ANALOG} \leq 5.1$. The chip will be destroyed if this range is exceeded.

ADC channel assignments are as follows:

ADC CHANNEL	HEX ADDRESS	USE
0	E040	PERIPHERAL SLOT 1
1	E041	PERIPHERAL SLOT 2
2	E042	PERIPHERAL SLOT 3
3	E043	PERIPHERAL SLOT 4
4	E044	J1 - PIN 28 (SPARE)
5	E045	J1 - PIN 29 (SPARE)
6	E046	J1 - PIN 30 (SPARE)
7	E047	J1 - PIN 31 (SPARE)
8	E048	LEFT MOTOR BATTERY
9	E049	RIGHT MOTOR BATTERY
A	E04A	LOGIC BATTERY
B	E04B	SOUND LEVEL DETECTOR
C	E04C	BAROMETER
D	E04D	LIGHT LEVEL DETECTOR
E	E04E	TEMPERATURE LEVEL DETECTOR
F	E04F	MAIN RAM BATTERY BACKUP

There are eight (8) analog input channels available for experimentation.

The monitor contains a conversion routine, ADCONV ($E481), which will do a conversion of the analog voltage on the channel number contained in the Y-REG and return the results in the accumulator. This routine selects the correct channel, takes 8 readings of the voltage and returns the average of these readings. Total conversion time is on the order of 800 us.

BIT INPUT PORTS

There are twenty-four (24) bit inputs implemented using an 82C55A Programmable Peripheral Interface (PPI) chip. These bits are located and used as follows:

HEX ADDRESS	BIT	TRUE WHEN	USE
E050	0	low	On Charge
E050	1	high	IR Beacon Detector
E050	2	high	Door Reflector Detector
E050	3	osc*	Smoke Detector
E050	4	high	Motion Detector
E050	5	low	RCS Carrier Detector
E050	6	low	Right Head Stop
E050	7	low	Left Head Stop
E051	0	N/A	Slot Bit Input 2
E051	1	N/A	Slot Bit Input 1
E051	2	N/A	Not Used
E051	3	N/A	Not Used
E051	4	SW4	Internal Baud Rate Switch
E051	5	SW3	External Baud Rate Switch
E051	6	SW2	External Baud Rate Switch
E051	7	SW1	Term/Keyboard Select
E052	0	N/A	Spare to SSS
E052	1	N/A	Spare to SSS
E052	2	N/A	Spare to SSS
E052	3	low	Function Key 5
E052	4	low	Function Key 4
E052	5	low	Function Key 3
E052	6	low	Function Key 2
E052	7	low	Function Key 1

* When smoke is detected, this line oscillates between high and low. It is low when no smoke is present.

BIT OUTPUT PORTS

There are twenty-four (24) bit outputs implemented using an 82C55A Programmable Peripheral Interface (PPI) chip. These bits are located and used as follows:

HEX ADDRESS	BIT	TRUE WHEN	USE
E060	0	high	PROCON Enable
E060	1	high	VIOS Enable
E060	2	low	Battery Ground Connected
E060	3	high	Sonar Pulse Enable
E060	4	low	Navigation Sensors Enable
E060	5	low	Sonar Power Enable
E060	6	high	Motion Detector Enable
E060	7	high	RCS Modem Enable
E061	0	N/A	Slot Bit Output 2
E061	1	N/A	Slot Bit Output 3
E061	2	varies	Bit Select 3 (ROM
E061	3	varies	Bit Select 2 CARD
E061	4	varies	Bit Select 1 SELECT)
E061	5	high	Serial Spare 2 Enable
E061	6	high	Serial Spare 1 Enable
E061	7	low=1MHz high=2MHz	CPU Clock Speed Select
E062	0	varies	Sonar Transducer Select-D
E062	1	varies	Sonar Transducer Select-C
E062	2	varies	Sonar Transducer Select-B
E062	3	varies	Sonar Transducer Select-A
E062	4	varies	Phase 3 Stepper Motor Winding
E062	5	varies	Phase 2 Stepper Motor Winding
E062	6	varies	Phase 1 Stepper Motor Winding
E062	7	varies	Phase 0 Stepper Motor Winding

BIT SELECT FUNCTIONS

BIT SEL 3	BIT SEL 2	BIT SEL 1	RAM/ROM SELECTED
low	low	low	RAM ON MAIN BOARD
low	low	high	ROM/RAM 1 ON ROM EXPANSION CARD
low	high	low	ROM/RAM 2 ON ROM EXPANSION CARD
low	high	high	ROM/RAM 3 ON ROM EXPANSION CARD
high	low	low	ROM/RAM 4 ON ROM EXPANSION CARD
high	low	high	ROM/RAM 5 ON ROM EXPANSION CARD
high	high	low	ROM/RAM 6 ON ROM EXPANSION CARD
high	high	high	ROM/RAM 7 ON ROM EXPANSION CARD

SONAR TRANSDUCER SELECT FUNCTIONS

SEL A	SEL B	SEL C	SEL D	SONAR SELECTED
LOW	LOW	LOW	LOW	HEAD
LOW	LOW	LOW	HIGH	LEFT SIDE
LOW	LOW	HIGH	LOW	RIGHT SIDE
LOW	LOW	HIGH	HIGH	LEFT TOP FRONT
LOW	HIGH	LOW	LOW	RIGHT TOP FRONT
LOW	HIGH	LOW	HIGH	LEFT BOTTOM FRONT
LOW	HIGH	HIGH	LOW	RIGHT BOTTOM FRONT
LOW	HIGH	HIGH	HIGH	FRONT CENTER
HIGH	LOW	LOW	LOW	BACK
HIGH	LOW	LOW	HIGH	SPARE-NO TRANSDUCER

WARNING: DO NOT USE ANY OTHER COMBINATION OR YOU WILL BURN OUT THE SONAR CONDITIONING BOARD ON THE SSS BOARD

RANDOM NUMBER GENERATOR

There is a hardware random number generator on the main CPU which can provide an 8 bit random number from 00 to FF. This random number can be obtained by reading $E070-E073. Care should be taken when using machine language routines to obtain a sequence of random numbers. The random transitions from the noise chip are input to a ripple type counter which takes time to propagate through the counter. Sequential readings should not be less than about 1 ms apart.

KEYBOARD INTERFACE

The keyboard transmits its data to the main computer serially. The serial signal is converted to parallel data using a HD6350 CMOS Asynchronous Communications Interface Adapter (ACIA). Subroutine RDKEY ($E460) in the monitor gets the character from this chip, and leaves it in the accumulator.

BELL

There is a piezoelectric buzzer attached to J4 on the main computer. By referencing $E078-E07B, the buzzer can be caused to click. Programs can be written to cause the buzzer to make various sounds. One such program, BEEP, is contained in the Monitor.

NMI RESET

Should any one of the batteries fall below about 10.5 volts, circuitry on the SSS board will generate a low battery signal that will cause a flip-flop on the main computer to change state. This change of state will force a non-maskable interrupt (NMI) of the CPU. The NMI services routine in the Living Mode Operating System will then cause the robot to issue a critically low battery warning. If after one minute the user does not respond to the warning, the robot will return to its charger. To remove the NMI, the memory locations $E07C-E07F is referenced which resets the flip-flop.

EXPANSION SLOTS

The main computer board contains four peripheral connectors that are physically identical to those on Apple II (tm) series computers. The signals brought out to these pins, however, are not identical to the Apple II (tm) series, although they are similar. There is no guarantee that a peripheral for the Apple II (tm) series computer will work on GEMINI although some do. Experimental (blank) cards are readily available from many sources. Figure 10.1 shows a typical connector and pin lables. Table 10.1 provides a description of each signal.

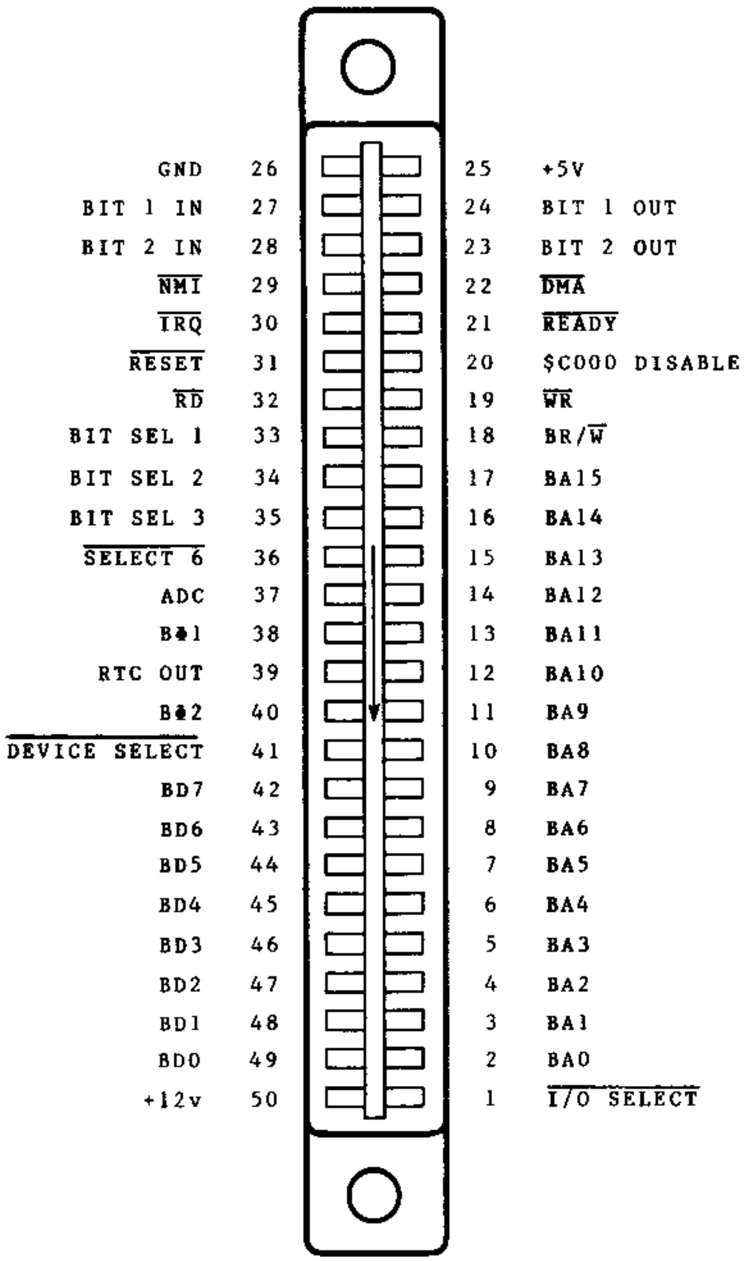
When designing peripherals for this computer, you must keep in mind that all logic signals are CMOS level signals. CMOS and LSTTL/TTL logic levels are in general not compatible. Usually there is no problem when a CMOS output signal drives an LSTTL input because the CMOS level exceeds the requirements of the LSTTL input signal. However, CMOS current drive capabilities are usually not the same as LSTTL so it is easy to overload the CMOS output signal.

When an LSTTL or TTL output signal attempts to drive a CMOS input signal, there is a serious incompatibility. Usually LSTTL output voltage levels are in the "undetermined" region of CMOS logic voltage. Pull up resistors can help to overcome this problem.

If you plan to interface LSTTL to the main computer, then you should give careful consideration to the above and make sure your signals are properly buffered. Whenever possible, we highly recommend the use of high speed CMOS logic (the "HC" series). This logic family carries the same designations as LSTTL and is easy to work with.

FIGURE 10.1

EXPANSION CARD CONNECTOR PINOUTS



GND 26
25 +5V
BIT 1 IN 27
24 BIT 1 OUT
BIT 2 IN 28
23 BIT 2 OUT
$\overline{\text{NMI}}$ 29
22 $\overline{\text{DMA}}$
$\overline{\text{IRQ}}$ 30
21 $\overline{\text{READY}}$
$\overline{\text{RESET}}$ 31
20 $C000 DISABLE
$\overline{\text{RD}}$ 32
19 $\overline{\text{WR}}$
BIT SEL 1 33
18 BR/$\overline{\text{W}}$
BIT SEL 2 34
17 BA15
BIT SEL 3 35
16 BA14
$\overline{\text{SELECT 6}}$ 36
15 BA13
ADC 37
14 BA12
BΦ1 38
13 BA11
RTC OUT 39
12 BA10
BΦ2 40
11 BA9
$\overline{\text{DEVICE SELECT}}$ 41
10 BA8
BD7 42
9 BA7
BD6 43
8 BA6
BD5 44
7 BA5
BD4 45
6 BA4
BD3 46
5 BA3
BD2 47
4 BA2
BD1 48
3 BA1
BD0 49
2 BA0
+12v 50
1 $\overline{\text{I/O SELECT}}$

TABLE 10.1

EXPANSION SLOT SIGNAL DESCRIPTIONS

PIN	NAME	I/O	DESCRIPTION
1	$\overline{\text{I/O SELECT}}$	O	This line, normally high, will become low when the CPU references any of the 128 addresses pertaining to this slot. The signal is active during one complete CPU cycle. See Table 10.1A for address and Figure 10.2 for timing.
2-17	BA0-BA15	O	The buffered address bus. These lines are tri-stated when DMA is active.
18	BR/$\overline{\text{W}}$	O	Buffered Read/$\overline{\text{Write}}$ signal. This line is valid at the same time as the address bus and goes high during a CPU read cycle and low during a write cycle. The line is tri-stated when DMA is active.
19	$\overline{\text{WR}}$	O	A $\overline{\text{Write}}$ signal. This line is a write enable signal that is active only during the half cycle when

PIN	NAME	I/O	DESCRIPTION
			Φ_0 in is high and R/$\overline{W}$ is low. It is not placed in tri-state when DMA is active.
20	$C000 DISABLE	I	When this signal is high, it disables the 8K RAM of $C000-DFFF. It is used for bank switching.
21	$\overline{READY}$	I	The 65C02's RDY input. Pulling this line low during Φ_1 will halt the CPU, with the address bus holding the address of the current locations being fetched.
22	$\overline{DMA}$	I	Pulling this line low halts the CPU and places the address bus, data bus, BR/$\overline{W}$, BΦ_2, and BΦ_1 in tri-state.
23-24	BIT OUTPUTS	O	Two bit output lines available to all slots. $E061, Bits 0 and 1.
25	+5V	O	+5V power supply. The switching regulator on the

PIN	NAME	I/O	DESCRIPTION
			power supply board is capable of sourcing 2 amps. Care must be taken that all peripherals do not cause this value to be exceeded.
26	GND	N/A	Logic electrical ground. Connected to motor grounds only when robot on charger.
27-28	BIT INPUTS	I	Two bit input lines available to all slots. $E051, Bits 0 and 1.
29	$\overline{\text{NMI}}$	I	Non-Maskable Interrupt. When pulled low the CPU begins an interrupt cycle and jumps to the interrupt service routine at the location designated by the contents of memory locations $16,17.
30	$\overline{\text{IRQ}}$	I	Interrupt Request. When pulled low the CPU begins an interrupt cycle (provided the CPU´s interrupt disable flag is

PIN	NAME	I/O	DESCRIPTION
			not set) and jumps to the interrupt service routine at the location designated by the contents of memory locations \$14,15.
31	$\overline{\text{RESET}}$	I	When pulled low, the CPU begins a reset cycle. This interrupt is not vectored because the monitor ROM must execute certain instructions to insure the equipment comes up in a known state.
32	$\overline{\text{RD}}$	O	A $\overline{\text{READ}}$ signal. This line is a read enable signal that is active only during the half cycle when Φ_0 in is high and R/$\overline{\text{W}}$ is high. It is not placed in tri-state when $\overline{\text{DMA}}$ is active.
33-35	BIT SEL 1,2,3	O	These lines are for use in bank switching to ROM/RAM on the ROM EXPANSION BOARD. See Table 10.2B for the selection chart.

PIN	NAME	I/O	DESCRIPTION
36	$\overline{\text{SELECT 6}}$	O	This line, normally high, will go low when the CPU references any address in the $C000-DFFF range. The signal is active during one complete CPU cycle.
37	ADC	I	An input to the analog to digital converter. See Table 10.2C for selections and for use.
38	BΦ1	O	Buffered Φ_1 output of the CPU. The line is tri-stated when $\overline{\text{DMA}}$ is active.
39	RTC OUT	O	A square wave output signal from the Real Time Clock. See Table 10.2D for frequency selection and for use.
40	BΦ_2	O	Buffered Φ_2 output of the CPU. The line is tri-stated when $\overline{\text{DMA}}$ is active.
41	$\overline{\text{DEVICE SELECT}}$	O	This line normally high, will become low when the CPU references any of the 32 addresses

PIN	NAME	I/O	DESCRIPTION
			pertaining to this slot. The signal is active during one complete CPU cycle. See Table 10.1E for addresses on Figure 10.2 for timing.
42-49	BD7-BD0	I/O	The buffered data bus. These lines are tri-stated when $\overline{\text{DMA}}$ is active.
50	+12V	O	A non-regulated, nominally 12V supply directly from the logic battery.

TABLE 10.1A

EXPANSION SLOT $\overline{\text{I/O SELECT}}$ ADDRESSES

SLOT	HEX ADDRESS
1	$E100-E17F
2	$E180-E1FF
3	$E200-E27F
4	$E280-E3FF

TABLE 10.1B

EXPANSION SLOT ROM/RAM BOARD BIT SELECTIONS

	$E061		CHIP	
BIT 2	BIT 3	BIT 4	SELECTED	PROGRAM
0	0	0	RAM 0	MAIN CPU
1	0	0	ROM 1	LMOS
0	1	0	ROM 2	SCHEDULER
1	1	0	ROM 3	BASIC
0	0	1	ROM 4	NAVIGATION
1	0	1	ROM 5	VOCOL
0	1	1	ROM 6	CATALOG
1	1	1	ROM 7	SECURITY

TABLE 10.1C

EXPANSION SLOT
ANALOG TO DIGITAL PORT ASSIGNMENTS

SLOT #	A/D PORT #
1	0
2	1
3	2
4	3

To use: Load the "Y" Register with the A/D port number and jump to the subroutine CONVRT ($E481). The A/D conversion value is returned in the Accumulator.

TABLE 10.1D

EXPANSION SLOT
RTC SQUARE WAVE FREQUENCIES

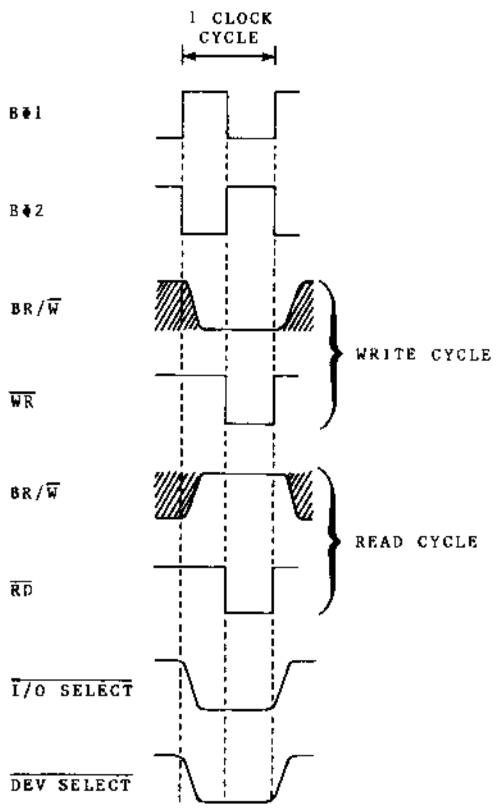
BIT 7	BIT 6	BIT 5	BIT 4	WAVE PERIOD
0	0	0	0	DISABLE
0	0	0	1	488.2 us
0	0	1	0	976.5 us
0	0	1	1	1953.1 us
0	1	0	0	3906.2 us
0	1	0	1	7812.5 us
0	1	1	0	15.625 ms
0	1	1	1	31.25 ms
1	0	0	0	62.5 ms
1	0	0	1	125 ms
1	0	1	0	250 ms
1	0	1	1	500 ms
1	1	0	0	second
1	1	0	1	minute
1	1	1	0	hour
1	1	1	1	day

TABLE 10.1E

EXPANSION SLOT
DEVICE SELECT ADDRESSES

SLOT	HEX ADDRESS
1	$E080-E09F
2	$E0A0-E0BF
3	$E0C0-E0DF
4	$E0E0-E0FF

FIGURE 10.2
EXPANSION SLOT SIGNAL TIMING



NOTE: The 65C02 uses the falling edge of Φ_2 to clock data into (read) the chip from a peripheral or memory device. It also "assumes" a peripheral device or memory device will use this edge to clock in data (write). Do not use $\overline{\text{I/O SELECT}}$ or $\overline{\text{DEVICE SELECT}}$ as a "write" strobe as the peripheral device will miss the data. Use WR instead for this purpose and either $\overline{\text{I/O SELECT}}$ or $\overline{\text{DEVICE SELECT}}$ to enable the peripheral chip.

CHAPTER 11

KEYBOARD AND DISPLAY

KEYBOARD

The keyboard reads the key you press and serially transmits the ASCII code for this key to the main computer. Serial data is transmitted either through a wire or via infrared light waves.

The keyboard contains a Hitachi HD6303 CMOS microcomputer, 2K CMOS ROM, IR transmission circuitry, rechargeable nicad batteries, the keys and miscellaneous CMOS support chips. The keyboard is, in essence, a computer.

The keyboard's microcomputer does not continually scan the keys. Instead it puts itself to sleep (to conserve power) and then awakens when a key is pressed. The microcomputer then quickly scans the keyboard to determine which key (or keys when the SHIFT or CONTROL key is also pressed) was pressed and then transmits the ASCII code for this key. The ASCII characters corresponding to each key are given in Table 11.1. The keyboard does not use lower case. Serial data is transmitted using one start bit, seven data bits and two stop bits. The data transmission rate is 255 bits per second.

There are five 1.2 volt AA (180 ma) nicad batteries inside the keyboard. When the keyboard is connected to the robot and the robot is on, these batteries will be charged.

If the keyboard cord is disconnected from the robot, it can be used to transmit keys to the main computer via the IR link. The IR receiver (on the robot) is located directly above the LCD and the IR transmitting diodes are located at the rear of the keyboard. To use the keyboard in the IR mode, it must be pointing approximately at the LCD. You will need to experiment with the proper height and distance to obtain reliable transmissions. IR transmissions are accomplished through 100 percent modulation of a 33 Khz IR signal.

TABLE 11.1

KEYS AND THEIR TRANSMITTED ASCII CODES

KEY	ALONE	CTRL	SHIFT
!1	31	NONE	21
"2	32	NONE	22
#3	33	NONE	23
$4	34	NONE	24
%5	35	NONE	25
&6	36	NONE	26
´7	37	NONE	27
(8	38	NONE	28
)9	39	NONE	29
0	30	NONE	20
*:	3A	NONE	2A
-=	2D	NONE	3D
↑	5E	1E	7E
ESC	1B	1B	1B
Q	51	11	51
W	57	17	57
E	45	05	45
R	52	12	52
T	54	14	54
Y	59	19	59
U	55	15	55
I	49	09	49
O	4F	0F	4F
P	50	10	50
←	08	7F	7F
→	09	1C	7C
↓	0A	0A	0A
A	41	01	41
S	53	13	53
D	44	04	44
F	46	06	46
G	47	07	47
H	48	08	48
J	4A	0A	4A

KEY	ALONE	CTRL	SHIFT
K	4B	0B	4B
L	4C	0C	4C
+;	3B	NONE	2B
`@	40	NONE	60
RETURN	0D	0D	0D
Z	5A	1A	5A
X	58	18	58
C	43	03	43
V	56	16	56
B	42	02	42
N	4E	0E	4E
M	4D	0D	4D
<,	2C	NONE	3C
>.	2E	NONE	3E
?/	2F	NONE	3F
SPACE	20	20	20

LIQUID CRYSTAL DISPLAY

The LCD is a 40 x 8 alpha-numeric, 5 x 7 dot matrix Epson EA-Y40080AT type of display with a high contrast and a wide, adjustable viewing angle. The display contains its own CMOS segments drivers, controllers and display data RAM. The LCD consumes very little power (5 ma at +5 volts).

The LCD is capable of displaying 96 ASCII characters and 64 Japanese Katakana characters. Those characters which can be displayed are given in Table 11.2.

There are four separate controllers used in the LCD to display data. Each controller controls two separate lines (rows) on the LCD. The addresses of these controllers and the lines they control are as follows (see also Figure 11.1).

HEX ADDRESS	LINE NUMBER
E000-E003	0,1 (0 = top line)
E004-E007	2,3
E008-E00B	4,5
E00C-E00F	6,7

Each controller must be given command or character data in order to display characters and control the cursor. Commands are listed in Table 11.3 along with their meaning. A write of a hex command byte given in this table to $E000 (or E002) will give that command to the controller of lines 0 and 1. A write of a hex ASCII character byte to $E001 (or E003) will place the hex ASCII character byte in the display data RAM at the current location of the cursor and automatically increment or decrement the cursor depending on the SET CURSOR DIRECTION state. LCD lines 2-7 are similarly addressed:

COMMAND ADDRESS NUMBERS	CHARACTER ADDRESS	LCD LINE NUMBERS
E000,E002	E001,E003	0,1
E004,E006	E005,E007	2,3
E008,E00A	E009,E00B	4,5
E00C,E00E	E00D,E00F	6,7

The controllers take time to execute commands and store data so it is necessary to determine if the controller is busy before writing data to it. This can be accomplished by reading from any of the addresses given above. If a controller is busy, the high bit (bit 7) of the data byte read will be high, otherwise it will be low.

The monitor contains many routines used to manipulate characters on the LCD. At the lowest level are subroutines COMDAT ($E4A2), which gives the command contained in the accumulator to the LCD, and CHRDAT ($E4A5), which gives the character data in the accumulator to the LCD. At the highest level is Subroutine LCDOUT ($E45D) which will display the character data in the accumulator and scroll data on the screen as necessary.

A new image of the LCD´s display data RAM is kept in main RAM at $0400-053F. This area of main RAM is specifically reserved for use by the monitor and if destroyed by a user will result in garbage data on the LCD.

Lower 4 bit (D_0 to D_3) of Character Code (Hexadecimal)

Higher 4 bit (D_4 to D_7) of Character Code (Hexadecimal)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	→	←
A		。	「	」	、	・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ
B	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
C	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
D	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	゛	゜

TABLE 11.2

LCD CHARACTER CODE MAP

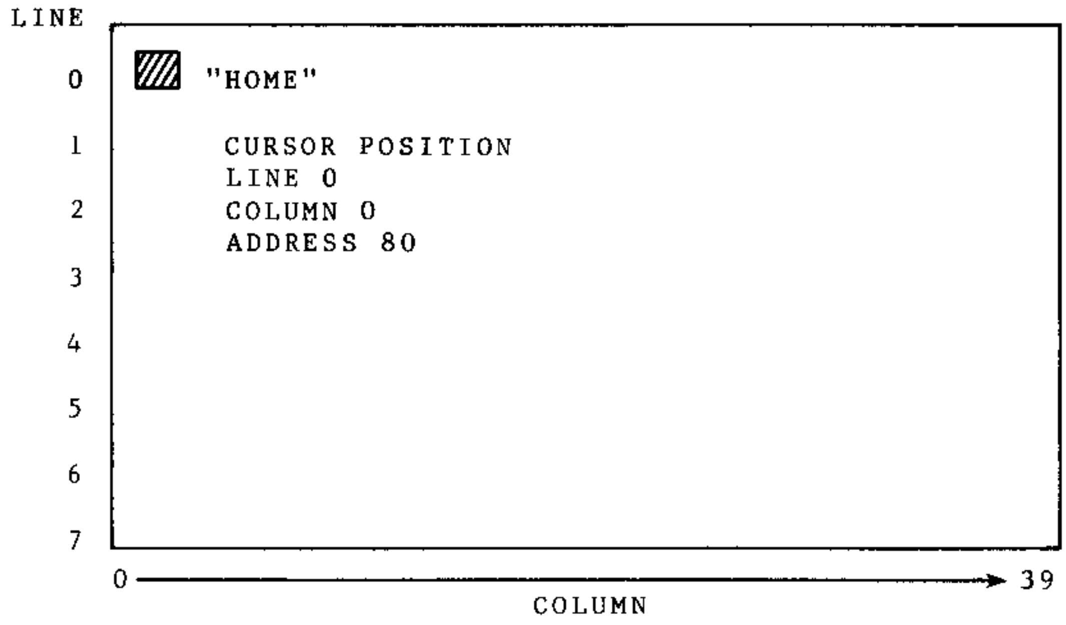
TABLE 11.3

LCD COMMANDS

COMMAND BYTE	COMMAND
01	CLEAR DISPLAY DATA - all display data RAM in LCD is set to ASCII space code ($20) and cursor returns to "home" position (See Figure 11.2)
02	CURSOR HOME - cursor returns to "home" position (See Figure 11.2)
03	CURSOR RETURN - cursor returns to beginning of current line (See Figure 11.2)
04,05	SET CURSOR DIRECTION - 04 = forward, 05 = reverse. Note display suppress on/off = off (display cursor)
06,07	CURSOR INC/DEC - 06 = increment (right), 07 = decrement (left)
08,09	SET CURSOR FONT - 08 = underline, 09 = 5x7 blinking
0A,0B	UNDERLINE CURSOR BLINKING ON/OFF 0A = off, 0B = on
0C,0D	DISPLAY ON/OFF - 0C = off (enable), 0D = on (disable). Display data RAM is not cleared
0E,0F	CURSOR SUPPRESS ON/OFF - 0E = off (enable), 0F = on (disable)
10	SYSTEM RESET - display on/off = off (enable) - cursor suppress on/off = off (enable)

COMMAND BYTE	COMMAND
	- underline cursor blinking on/off = off - set cursor font = underline - display suppress on/off = off - set cursor direction = forward Display data RAM is not cleared
44,45	DISPLAY SUPPRESS ON/OFF - 44 = on (suppress cursor) with cursor direction forward - 45 = on (suppress cursor) with cursor direction reverse
80-A7,C0-E7	SET CURSOR ADDRESS - 80 = column 0, A7 = column 39 of lines 0,2,4,6 - C0 = column 0, E7 = column 39 of lines 1,3,5,7

FIGURE 11.1
LIQUID CRYSTAL DISPLAY



LIFE
OPERATING SYSTEM

SECTION VI

LIFE OPERATING SYSTEM

INTRODUCTION

The software contained in ROM on the main and auxillary computers was designed as a life-imitating operating system. Once the robot is powered up, the software attempts to keep the robot "alive" (batteries charged), while constantly monitoring both internal sensors for possible problems and external sensors for possible user access. The software provides a means for a user to enter tasks for the robot to aperiodically or periodically accomplish at future specified times; something humans do all the time. The software also provides the robot with an important human-like quality; the ability to move from room to room and pass through open doorways.

This software is divided into several major programs which are fully integrated and which make use of one another. In this section, several of the major programs are described in some detail in order to allow the user access to the subroutines contained therein. In addition, commented source code listings of many of the major programs are available for purchase.

CHAPTER 12

MEMORY ORGANIZATION

A fully equipped GEMINI robot contains 107K bytes of 65C02 machine language code spread among all of its computers. The MAIN computer and the ROM Expansion Card contain 63K bytes; the VIOS computer contains 32K bytes; the RCS computer contains 8K bytes; the PROCON computer contains 2K bytes, and the KEYBOARD computer contains 2K bytes.

MAIN COMPUTER ROM AND ROM EXPANSION CARD UTILIZATION

The 63K bytes of ROM on this computer and card are distributed among several major sub-programs as follows:

HEX ADDRESS	MEMORY ALLOCATION	PROGRAM (ABREVIATION)
E400-FFFF	7K	System Monitor (MON)
C000-DFFF	8K	Bank 1-Living Mode Operating System (LMOS)
C000-DFFF	8K	Bank 2-Scheduler(SCH)
C000-DFFF	8K	Bank 3-Basic (BAS)
C000-DFFF	8K	Bank 4-Basic/ Navigation (NAV)
C000-DFFF	8K	Bank 5-Voice Command Language (VOC)/Remote Communications System (RCS)
C000-DFFF	8K	Bank 6-Message Catalog (CAT)
C000-DFFF	8K	Bank 7-Security (SEC) Expansion

MAIN COMPUTER RAM UTILIZATION

All of the programs utilize nearly 4K of RAM out of the total of 56K available RAM. In addition, both the Scheduler and Basic programs utilize RAM to store user schedules and programs. Tables 12.1 through 12.4 provide RAM memory maps showing the utilization of RAM by these programs. Zero page memory on a 65C02 is very valuable real estate, due to the advanced addressing modes of this microprocessor. A detailed map of this area is provided so you do not inadvertently destroy critical variables in this area with your programs.

TABLE 12.1

RAM USAGE MAP (56K CONFIGURATION)

1000	Beginning of free RAM. LOMEM ($12,$13) set to here by monitor. BASIC programs normally begin here.
BFFF	End of free RAM (56K configuration). HIMEM ($10,$11) set to here by LMOS if: (1) RAM at $C000-DFFF is installed, and (2) GEMDOS or GEMTOS is not installed. Highest RAM location available to BASIC programs.
C000	SCHEDULER low memory (SCHLLM-$79,$7A) for a 56K RAM configuration.
DFFF	SCHEDULER high memory (SCHLHM-$7B,$7C) for a 56K RAM configuration.

TABLE 12.2

RAM USAGE MAP FOR VARIOUS RAM CONFIGURATIONS

RAM CONFIGURATION	HIMEM	SCHLLM	SCHLHM	RAM FOR BASIC PROGRAM
8K	1BFF	1C00	1FFF	3K
16K	37FF	3800	3FFF	10K
24K	54FF	5400	5FFF	17K
32K	6FFF	7000	7FFF	24K
40K	8BFF	8C00	9FFF	31K
48K	A7FF	A800	BFFF	38K(1)
56K	BFFF	C000	DFFF	42K

(1) If GEMDOS or GEMTOS is installed, HIMEM will be shifted down in memory. See the GEMDOS/GEMTOS manuals for memory maps when either of these systems is installed.

TABLE 12.3

RESERVED RAM UTILIZATION

HEXIDECIMAL MEMORY RANGE	USAGE
0000-00FF	Zero Page (See Table 12.3)
0100-02FF	CPU Stack
0200-02FF	Line Buffer
0300-03FF	Message Buffer
0400-053F	LCD Display Image
0540-0570	Monitor Variables
0570-0577	LMOS Variables
0578-057F	SCH "
0580-0587	BAS "
0588-058F	NAV "
0590-059F	RCS "
0600-06FF	Integer Math Pack Variables
0700-07FF	Room Tables
0800-087F	LMOS Variables
0880-08FF	SCH "
0900-097F	BAS "
0980-09FF	NAV "
0A00-0A7F	RCS "
0A80-0AFF	VOC "
0B00-0BFF	Disk/Tape Boot Buffer
0C00-0CFF	Tape VTOC Buffer
0D00-0DFF	Tape Catalog Sector 0
0E00-0EFF	Tape Catalog Sector 1
0F00-0FFF	Tape Read/Write Buffer

TABLE 12.4

ZERO PAGE RAM UTILIZATION

HEXIDECIMAL MEMORY LOCATION(S)	NAME	USE
00-0F	-	RESERVED
10-11	HIMEM	Highest RAM location available to BASIC program
12-13	LOMEM	Start of BASIC programs
14-15	IRQLOC	IRQ vector
16-17	NMILOC	NMI vector
18-19	ALMIRQ	RTC alarm vector
1A-1B	CLKIRQ	RTC clock out vector
1C-1D	CHROUT	Output character vector
1E-1F	CHARIN	Input character vector
20-21	CHARPL	Input character (polling) vector
22-23	USRMON	Monitor U-command vector
24	COLDAD	Cold start marker
25-2B	TIMBUF	RTC buffer
2C	DOWBUF	Day of week
2D	CLKMOD	Clock mode
2E	BUFCNT	Line buffer counter
2F	ROWCNT	LCD row counter
30	CURADD	Current address displayed by monitor
31-36	TEMPO	Temporary variables
37-38	INDEX1	
39-3A	ADRPTR	Address pointer for BANK 0 ram ACCESS
3B-3C	DSPIMG	Display image pointer
3D-3E	ENDIMG	End of display image pointer
3F-40	ADDR01	Target address for block move
41-42	ADDR02	End address for block move
43-44	ADDR03	Start address for block move

45	ROMSTA	ROM status byte
46	RAMSTA	RAM status byte
47-48	PCL	Program counter Lo byte
49-4A	PCH	Program counter Hi byte
4B	REGA	A-reg save
4C	REGX	X-reg save
4D	RECY	Y-reg save
4E	REGP	P-reg save
4F	REGS	S-reg save
50	HDSTAT	Head status flags
51	CPUFLG	CPU status flags
52-53	MSGNUM	Message pointer
54-56	QC	Math Pack Variables
57-59	TC	" " "
5A	FLG	" " "
5B	FLH	" " "
5C-5E	RC	" " "
5F	TMP	" " "
60	CNT	" " "
61	ACX	" " "
62-64	AC	" " "
65-67	SC	" " "
68-6C	--	Reserved
6D-6E	USRLIV	Life Loop user vector
6F-70	LIFEAD	Life Loop cold start vector
71-72	USRVOC	VOCOL user vector
73	DEMST	DEMO status flags
74	ROSTAT	Robot status flags
75	STAFLG	Navigation status flags
76	TRNST	Navigation status flags
77-78	MSGPTR	Message pointer
79-7A	SCHLHM	SCHEDULER HIMEM
7B-7C	SCHLLM	SCHEDULER LOMEM
7D-7E	BUFBAS	TOS/DOS buffer pointer
7F	SCHSTA	SCHEDULER status flags
80	SENBYT	Active sensor flags
81-82	DATA	Data pointer
83-84	PRTNUM	Message pointer
85-86	CURRENT	Room table pointer
87-B0	NOT USED	Free
B1-FF	RESERVED	BASIC

CHAPTER 13

SYSTEM MONITOR

The principal "work horse" of all software on the main computer is a program referred to as the "Monitor". It is responsible for starting up the robot and contains numerous utility routines which handle every basic function on the robot. If you wish to develop machine language programs for GEMINI, you should become intimately familiar with the many useful subroutines in this program, and learn how to use its command language. The material in this manual will provide you with a good start in this effort but, eventually, you should study the source code as well. The source code is available from ARCTEC SYSTEMS. Table 13.1 provides a detailed listing of entry points for some of the more useful routines.

MONITOR COMMAND LANGUAGE

The Monitor contains a simple, yet very powerful, program to assist you with program development and diagnosing problems with the robot. You can enter this program in several ways. The easiest way is to simply press Function Key 5 (or Keyboard Key 5) when the robot's "Life Menu" is being displayed. This forces a jump to the Monitor's GETCMD routine at E403. Of course, a BASIC program, or a user's machine language program, can also call this routine.

GETCMD sets up the Monitor's prompt (>) and then calls GETLIN (E41B) to get a command line into the line buffer from the current input device. GETCMD then transfers control to DOCMD (EE75) to handle execution of the command. DOCMD is an alternate entry point available to you. Of course, you must be sure the line buffer contains the command to be executed prior to its call.

Monitor commands can be divided into three groups. There are single letter character commands, single address commands, and multiple address commands. Table 13.2 explains the use of these commands.

The Monitor does not contain STEP or TRACE capabilities. In order to debug machine language programs, you should insert BREAK ($00) opcodes at troublesome points in your program. When the CPU encounters a break opcode, the Monitor will display the address at which the break occurred, and the current contents of the CPU registers as follows:

```
 PC          A      X      Y      P     S

$XXXX        ZZ     ZZ     ZZ     ZZ    ZZ
```

Where PC is the Program Counter, A is the accumulator; X is the x-register; Y is the y-register; P is the processor status register, and S is the stack pointer.

You may then use the Monitor commands to examine memory locations as well. This method of debugging programs is very efficient and allows time critical routines (which must be run at normal speed to prevent damage to the robot) to execute normally.

TABLE 13.1

SOME USEFUL MONITOR SUBROUTINES

ADDRESS	NAME	DESCRIPTION
E400	MNTRO	Warm/cold start monitor If COLDAD ($24) contains $5A then warm start; else cold start
E403	GETCMD	Get a monitor command; get a command line from current input device and try to execute it.
E406	RETINT	Return from interrupt -- Restore A, X and Y registers and perform RTI.
E409	MEMMOV	Move a block of memory. Move range of memory starting at contents of $43,44 and ending at contents $41, 42 to location starting at contents of $3F,40.
E40C	ROMCHK	Test monitor and bank roms. Perform checks on calculation on monitor roms and roms in BANKS 1-7. Set bits in $45 (0=monitor; 1=BANK 1, etc.) if check sums verify.
E40F	CLDISP	Clear LCD display and RAM image. Fill LCD display RAM and RAM display image with $20.
E415	CLRBUF	Clear the line buffer. Fill $0200-02FF with $20 and initialize BUFCNT ($20) to 0.

E418	GETLIN	Get a line from current input device. Clear line buffer; send prompt to current output device; get characters from current input device and return on receipt of a carriage return.
E41B	OUTCHR	Output character; ASCII character in accumulator is sent to current output device. Routine vectors output through CHROUT ($1C,1D) normally set to LCDOUT.
E41E	RDCHAR	Get a character from current input device in the accumulator. Routine vectors input through CHARIN ($1E,1F) normally set to RDKEY.
E421	INCHAR	Poll current input device. Get an ASCII character in the accumulator, if any. Otherwise, return with $00 in accumulator.
E427	OPENP	Open logical serial port. Port number (1-6) must be in accumulator.
E420	OUTCH	Send a character to open serial port. Character to send is in the accumulator. Accumulator is not destroyed. Routine will not return if receiver´s clear to send; doesn´t go low.
E430	INTAPL	Open the dual parallel communications port.
E433	SNDCHR	Send a character to the dual parallel port. Character to send is in accumulator. Routine will not return if character is not taken.

E436	RECCHR	Get a character from the dual parallel port. Character returned in accumulator. Routine will not return if character is not received.
E439	GETFNC	Get the function key. Returns with zero in accumulator if no key pressed, or 01-05 if a key is pressed. Also sounds bell on key press.
E43C	GETMSG	Get a message from catalog; unpack and put in message buffer at $0300.
E43F	MSGXMT	Send message to VIOS and/or current output device. Message in message buffer. Bit flags in DEMST ($73) are used to determine type of display and message routing.
E442	SETCLK	Set the RTC from memory. BCD time data stored in $25-29 (secs/mins/hrs/days/mos).
E445	GETCLK	Put the time from RTC into memory. BCD time data stored in $25-29.
E448	SETALM	Set the RTC alarm from memory. BCD time data stored in $25-27 (secs/mins/hrs).
E44B	ALMOFF	Turn off RTC alarm.
E44E	PRNTAX	Send contents of accumulator and x-register to current output device.
E451	HEXOUT	Send contents of accumulator to current output device.
E454	ASCHEX	Convert ASCII characters in accumulator and X register to hexidecimal.
E457	HEXASC	Convert to hexidecimal nibble in accumlator to ASCII.

E46F	INITHD	Initialize the head stepper motor (see Head Stepper Routines).
E472	LFTHDS	Find the left head step (see Head Stepper Routines).
E475	STPOFF	Turn off all stepper motor windings (see Head Stepper Routines).
E478	ONESTP	Turn head one step (see Head Stepper Routines).
E47B	POSHD	Move stepper motor to position in A-register (see Head Stepper Routines).
E47E	TURNHD	Turn head 1.5 degress (see Head Stepper Routines).
E481	ADCONV	Convert voltage on channel in y-register to hexidecimal.
E484	RDRANG	Read range on sonar in x-register (see Sonar Routines).
E487	ENBSNR	Enable sonar in x-register (see Sonar Routines).
E48A	DISONR	Disable sonars (see Sonar Routines).
E480	GETRNG	Get range on enabled sonar (see Sonar Routines).
E490	AVGRNG	Get average range on enabled sonar (see Sonar Routines).
E493	CROUT	Send a carriage return to current output device.
E496	SETTIM	Set the RTC.
E499	GETTIM	Get time from RTC.
E49C	SPCOUT	Move cursor right number of spaces in y-register.
E49F	TABOUT	Tab to column 20 of LCD display.
E4A2	COMDAT	Send command to LCD controllers.
E4A5	CHRDAT	Send character to LCD controllers.
E4A8	BITHAF	Wait one-half a serial bit time.
E4AB	BITWAT	Wait one serial bit time.

E4AE	SAVALL	Save all CPU registers.
E4B8	GETALL	Get all saved CPU registers.
E4C2	SNDSER	Send character in accumulator to serial port.
E4CA	RCVSER	Receive character in accumulator from serial port.
E4D6	PRCOUT	Send character in accumulator to PROCON.
E4E0	PRCIN	Receive character in accumulator from PROCON.
E4E8	VIOSOT	Send character in accumulator to VIOS.
E4F2	VIOSIN	Receive character in accumulator from VIOS.
E4FA	RFMOUT	Send character in accumulator to RF modem.
E50F	RFMIN	Receive character in accumulator from RF modem.
E524	EXTOUT	Send character in accumulator to external serial port.
E52E	EXTIN	Receive characters in accumulator from external serial port.
E536	EXP1OT	Send character in accumulator to expansion serial port 1.
E540	EX1IN	Receive character in accumulator from expansion serial port 1.
E548	EXP2OT	Send character in accumulator to expansion serial port 2.
E552	EXP2IN	Receive character in accumulator from expansion serial port 2.
E55A	BANKSW	Run a subroutine in another BANK.
E5A1	SWBANK	Jump to a location in another BANK.
E5BC	SAVBNK	Save a byte in BANK 0 RAM.

E5E5	LODBNK	Get a byte from BANK 0 RAM.
E60E	SETLSP	Set CPU default clock to 1MHZ.
E618	SETHSP	Set CPU default clock to 2MHZ.
E622	LOSPD	Set CPU clock to 1MHZ temporarily.
E62A	HISPD	Set CPU clock to 2MHZ temporarily.
E632	RESMSP	Resume default CPU clock speed.

TABLE 13.2

MONITOR COMMAND SUMMARY

SINGLE LETTER COMMANDS

COMMAND	FUNCTION
T	Send the current time from the RTC to the current output device.
S	Send time entry format to current output device and get time from current input device.
X	Enter the Dual Parallel Data Transfer Program.
C	Enter the RS232C Serial Data Transfer Program.
P	Send output to printer port.
E	Enter the Living Mode Operating System.
U	Enter a User Program whose starting address is stored in $22,23 (USRMON).

SINGLE ADDRESS COMMANDS

COMMAND	FUNCTION
$XXXXG	Jump (Go) to the program starting at address XXXX.
$XXXXL	Disassemble (List) six lines of 65C02 code starting at address XXXX and send to current output device. The "L" command may also be used alone and will disassemble code starting at the address contained in $3F,40. After executing $XXXXL, subsequent "L" commands will continue the disassembly.

$XXXXD	Send (<u>D</u>isplay) eight sequential hex characters to the current output device starting at address XXXX. The "D" command may also be used alone, and will send data starting at the address contained in $3F,40. After executing $XXXXD, subsequent "D" commands will continue the display.
$XXXX:YY YY YY	Store the hex characters (YY)into memory starting at address XXXX. The hex characters must be separated by a space, and can be of any length up to the limits of the line buffer.

MULTIPLE ADDRESS COMMANDS

<u>COMMAND</u>	<u>FUNCTION</u>
$XXXX.ZZZZD	Send (Display) the Hex character in memory to the current output device starting at XXXX and ending with ZZZZ.
$XXXX.ZZZZ:YY	Fill the memory starting at XXXX and ending with ZZZZ with the Hex character YY.
$XXXX<YYYY.ZZZZM	Move the Hex characters starting at address YYYY thru ZZZZ to memory starting at XXXX.
$XXXX<YYYY.ZZZZV	Verify the memory contents at YYYY thru ZZZZ with memory contents starting at XXXX. If the memory contents match, the Monitor returns with its prompt. Memory locations which do not match are displayed starting with the first non-match location beginning at XXXX. The corresponding memory in the YYYY thru ZZZZ range is given in parenthesis.

BANK ACCESS UTILITIES

There are four routines in the Monitor which handle jumps and subroutine calls between banks and retrieving/storing of data in the BANK 0 RAM by programs stored in BANKS 1-7. These routines are briefly described in Table 13.1.

To run a subroutine in another bank, and return to the bank from which the subroutine was called, you must use subroutine BANKSW ($E55A). For example:

```
LDY SUBLO     Low byte of subroutine address
LDX SUBHI     High byte of subroutine address
LDA BANKN     Bank number "N"
JSR BANKSW    Monitor routine at $E55A
```

After execution, program flow will return to the statement following the JSR BANKSW. All CPU registers are destroyed. Variables to be transferred must be put into RAM memory.

To jump to a program in another bank from a present bank, you must use subroutine SWBANK ($E5A1). For example:

```
LDY JMPLO      Low byte of jump location
LDX JMPHI      High byte of jump location
LDA BANKN      Bank number N
JMP SWBANK     Monitor routine at E5A1
```

After execution, program flow will continue in BANKN at the location JMPHI,JMPLO.

To save a byte in BANK RAM from another bank, you must use subroutine SAVBNK ($E15C). For example:

```
LDA MEMLO       Low address of Bank 0 Memory
                 Locations
STA ADRPTR      ($39)
LDA MEMHI       High Address of Bank 0 Memory
                 Locations
STA ADRPTR01    ($3A)
LDA VALUE       Value to be stored in MEMHI,
                 MEMLO
JSR SAVBNK
```

The X, Y and A registers remain intact.

To fetch a byte in BANK 0 RAM, you must use subroutines LODBNK ($E5E5). For example:

```
LDA MEMLO       Low Address of Bank 0 Memory
                 Locations
STA ADRPTR
LDA MEMHI       High Address of Bank 0 Memory
                 Locations
STA ADRPTR+1
JSR LODBNK      Value Fetched in Accumulator
```

The X and Y registers remain intact, and the value fetched is in the accumulator.

FRACTIONAL INTEGER MATH PACK

Located in the Monitor from E645-EA1A is a 24-bit, signed, fractional*, integer math pack which is used extensively by the Navigation Program, and is available for your use. This math pack is register-oriented, and has one 3-byte accumulator (AC) and three 3-byte working registers (RC,QC and TC). These registers are located in zero page memory in order to obtain minimum possible computation time.

Each register is stored in memory in reverse order following the 65C02 convention. The fractional part is in the low byte; the low byte of the integer part is in the middle byte, and the high byte of the integer part is in the high byte. These registers are located as follows:

HEX ADDRESS	NAME
62	ACL
63	ACM
64	ACH
5C	RCL
5D	RCM
5E	RCH
54	QCL
55	QCM
56	QCH
57	TCL
58	TCM
59	TCH

Numbers can only range from -32,768.9961 to +32767.9961.

The smallest fraction number which can be represented is .00391.

*A signed fractional integer is represented as follows:

$$\underbrace{\begin{array}{cccccccc} \text{sign} & 2^{14} & 2^{13} & 2^{12} & 2^{11} & 2^{10} & 2^{9} & 2^{8} \\ 23 & 22 & 21 & 20 & 19 & 18 & 17 & 16 \end{array}}_{\text{High Integer Byte}}$$

$$\underbrace{\begin{array}{cccccccc} 2^{7} & 2^{6} & 2^{5} & 2^{4} & 2^{3} & 2^{2} & 2^{1} & 2^{0} \\ 15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 \end{array}}_{\text{Low Integer Byte}}$$

$$\underbrace{\begin{array}{cccccccc} 2^{-1} & 2^{-2} & 2^{-3} & 2^{-4} & 2^{-5} & 2^{-6} & 2^{-7} & 2^{-8} \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{array}}_{\text{Fractional Byte}} \begin{array}{c} \text{WEIGHT} \\ \text{BIT} \end{array}$$

Many mathematical operations can be performed on the contents of these registers, and these operations are listed in Table 13.3, along with the routines in the Monitor which perform the operation. Also, it is possible to move the contents of one register to another, to store certain register contents into memory, and to load certain registers from memory. A special memory section called VAR ($0600-06FF) is set aside for these variables. Locations $0600 thru $0635 are used by the Navigation routines to store constants and 0636 thru 06A4 for variables. Your program should not destroy the constants at 0600 thru 0635. If your program uses the Navigation routines, you should not use the variable locations. The space from 06A5 thru 06FF (20 variables or constants) is free.

Mathematical operations are performed on the contents of the AC register alone (unary opertions), or on the contents of both the AC and RC registers (binary operations). There are two entry points for most unary operations, and three entry points for most binary operations in each subroutine.

In the lowest operation, the mathematical operation is performed on the AC register contents. In the highest unary operation, the contents of VAR storage, pointed to by the Y-register, is first loaded into the AC register and then the indicated operation is performed. In the highest binary operation, the contents of VAR storage, pointed to by the Y-register, is loaded into the AC and the contents of VAR storage pointed to by the X-register is loaded into RC. Then the indicated mathematical operation is performed. *The results are left in the AC register.*

For example, the following routine multiplies the variable A and the variable B, which are both stored in VAR storage, and stores the result in C back into VAR storage.

```
A    .EQ$00        Numeric equates
B    .EQ$03
C    .EQ$06
     LDY #A        Y points to A variable in
                     VAR storage
     LDX #B        X points to B variable in
                     VAR storage
     JSR MUL.2     AC = AC * RC
     LDY #C        Y points to C variable in
                     VAR storage
     JSR STR       Store AC
```

When using the math pack, keep in mind that there is no error checking of operations. You must know beforehand that the results will not overflow the registers. Also, the TC register is used by some routines and will destroy its contents.

The trig functions use degrees for their argument. Positive, and negative angles, and angles greater than 360 degrees can be used. Fractional angles are converted to the nearest whole degree. The inverse trig functions have limited accuracy.

The best way to learn how to use these routines is to experiment and practice. The routines are very powerful and worth the effort. The sonar routines store their results directly into the AC so the distances can be operated upon by these routines.

TABLE 13.3

INTEGER MATH PACK ROUTINES

ADDRESS	NAME	DESCRIPTION
E69F	CLR.AC	Clear AC (AC = 0)
E6A8	INC.1	Increment AC (AC = AC+1) after loading AC with variable designated by Y-register.
E6AB	INC.AC	Increment AC
E6B2	DEC.1	Decrement AC (AC = AC-1) after loading AC with variable designated by Y-register.
EGB5	DEC.AC	Decrement AC
E6C1	INT.1	Take integer value of AC (AC = INT) (AC + .5)) after loading AC with variable designated by Y-register.
E6C4	INT	Take integer value of AC
E6D0	TNC.1	Truncate AC (AC = INT)(AC)) after loading AC with variable designated by Y-register
E603	TNC	Truncate AC
E6DD	ADD.2	Add AC to RC (AC = AC+RC) after loading variable pointed to by Y-register in AC, and variable pointed to by X-register in RC
E6E0	ADD.1	Add AC to RC variable pointed to by Y-register in AC
E6E3	ADD	Add AC to RC
E6F0	SUB.2	Subtract RC from AC (AC = AC-RC) Y variable into AC, X variable into RC
E6F3	SUB.1	Same except Y variable into AC
E6F6	SUB	Same except no fetch
E703	SGN.1	A-REG = SIGN (AC) (A=00 for AC=0, A=1 for AC>0, A=FF for AC<0.Y variable into AC

E706	SGN	Same as SGN.1
E717	CMP.2	Compare AC with RC (A=00 for AC=RC, A=01 for AC > RC, A=FF for AC<RC) Y variable into AC, X variable into RC
E71A	CMP.1	Same except only Y variable into AC
E71D	CMP.AC	Same except no fetch
E740	CHG.1	Change sign (AC = -AC) Y variable into AC
E74D	CHG	Same except no fetch
E746	ABS.1	Absolute value of AC (AC = ABS (AC)) Y variable into AC
E749	ABS	Same except no fetch
E75C	MUL.2	Multiply AC by RC (AC = AC* RC) Y variable into AC, X variable into RC
E75P	MUL.1	Same except only Y variable into AC
E762	DIV.2	Divide AC by RC (AC = AC/RC) Y variable into AC, X variable into RC
E7AD	DIV.1	Same except only Y variable into AC
E7B0	DIV	Same except no fetch
E7FC	INV.1	Invert AC (AC = 1/AC) Y variable into AC
E7FF	INV	Same except no fetch
E812	SQR.1	Square root of AC (AC = SQR (AC)) Y variable into AC
E815	SQR	Same except no fetch
E88C	SIN.1	Sine of AC (AC = SIN(AC)) Y variable into AC
E88F	SIN	Same except no fetch
E8CD	COS.1	Cosine of AC (AC = COS(AC)) Y variable into AC
E8D0	COS	Same except no fetch
E806	TAN.1	Tangent of AC (AC=TAN(AC)) Y variable into AC
E8D9	TAN	Same except no fetch

E8F8	ASN.1	Arcsine of AC (AC = ASN(AC)) Y variable into AC
E8FB	ASN	Same except no fetch
E92C	ACS.1	Arccosine of AC (AC = ACS)(AC)) Y variable into AC
E92F	ACS	Same except no fetch
E946	ATN.1	Arctangent of AC (AC = ATN(AC)) Y variable into AC
E949	ATN	Same except no fetch
E990	LET.2	Var 1 = VAR 2 Y points to VAR 1, X points to VAR 2
E9A1	TMA	Fetch AC (AC = VAR) Y points to VAR
E9B1	TMR	Fetch RC (RC = VAR) X points to VAR
E9BF	STR	Store AC (VAR = AC) Y points to VAR
E9CB	TAR	Transfer AC to RC
E9D5	TAT	Transfer AC to TC
E9DF	TAQ	Transfer AC to QC
E9E9	TTA	Transfer TC to AC
E9F3	TTR	Transfer TC to RC
E9FD	TQA	Transfer QC to AC
EA07	TQR	Transfer QC to RC
EA11	TRA	Transfer RC to AC

SONAR RANGING ROUTINES

There are several subroutines in the Monitor which handle getting ranges from a selected sonar. Subroutine RDRANG (E484) enables the sonar indicated by the contents of the x-reg, gets the range and disables all sonars to conserve power. The range in microseconds round-trip is stored in the math packs AC as follows:

ACL = 0
ACM = Low byte of range (round-trip time)
ACH - High byte of range

It takes sound approximately 1775 (decimal) us to travel one foot and back (round-trip time). Thus, dividing ACH,ACM by this time will give the range in feet. Of course, it is unnecessary to make such a conversion since distance can be measured in "sonic" microseconds. This unit of distance is used throughout the Navigation routines.

Subroutine AVGRNG ($E490) will take the average of eight ranges on an enabled sonar and place the result in the math pack AC. You must first enable the desired sonar through a call to ENBSNR (E487) with the sonar number in the X-register. To conserve power, you should call DISONR ($E48A) afterwards to disable all sonars.

HEAD STEPPER ROUTINES

There are several routines in the Monitor for controlling the head stepper motor. In addition, there are several bytes and bit flags to store data and status relative to this matter. These variables are described in Table 13.4.

Subroutine INITHD ($E46F) is used to initialize the head at the left head stop. This is the counter clockwise most position of the head. The variable CURHD is set to zero. Prior to calling this routine, the head speed variable, MSPSTP ($0567), must be set (see Table 13.4).

Subroutine POSHD (E47B) can be used to position the head at any place within its travel range ($00 < CURHD < $EF). This routine will initialize the head, if not previously done. To use this routine, load the accumulator with the desired position and then call the subroutine. On return, the carry will be clear unless a position ($F0-$FF) has been ordered.

The stepper motor is capable of taking steps as small as 0.33 degrees. Subroutine ONESTP ($E478) takes one physical step in the direction indicated by the DIREC bit. On first call to this routine, STEPNO ($0569) should be set to zero.

The head position is divided into 239 (decimal) positions which are 1.5 degrees (5 physical steps) apart. This covers a range of 358.5 degrees. At position "0", the robot's head is against the left stop facing to the rear. At position 78, the robot's head is centered facing forward, and at position EF, the head is at the right stop facing to the rear. Subroutine TURNHD ($E47E) is used to turn the head one logical step (five physical steps). On entry to this routine, the head must be initialized, head speed must be set, and head direction must be specified. In addition to stepping the head five physical steps, the navigational sensors are energized and checks made at each physical step to see if a door edge reflector or a room beacon was detected. Status bits BECREF and DORREF (see Table 12.7) are set accordingly.

TABLE 13.4

HEAD STEPPER FLAGS AND VARIABLES

LOCATION	NAME	FUNCTION
$50	HDSTAT	Status Bits

BIT	NAME	USE
0	HDINIT	0=Head not initialized 1=Head initialized
1	DIREC	0=Set clockwise head rotation 1=Set counter-clockwise head rotation
4	BECREF	0=Beacon not found 1=Beacon found
5	DORREF	0=Door reflector not found 1=Door reflector found

LOCATION	NAME	FUNCTION
$0567	MSPSTP	Head Stepper Speed

MSPSTP	STOP TO STOP TIME (SECS)
08	10
09	11
OA	12
OB	13.5
OC	14.5
OD	16
OE	17
OF	18
10	19.5
11	20.5

LOCATION	NAME	FUNCTION
$0569	STEPNO	Last logical step number (0-3)
$056A	CURHD	Current head position ($00-EF)

CHAPTER 14

LIVING MODE OPERATING SYSTEM (LMOS)

BANK 1 on the ROM Expansion Card contains a life-imitating GEMINI robot operating system. After the System Monitor performs its housekeeping chores, program control is turned over to this supervisory program which will retain control over the robot until a cold reset is performed.

The heart of this operating system is an endless program loop which constantly monitors a number of sensors, peripherals and user input devices. When GEMINI is in this loop, a LIFE MENU is displayed and the cursor moves back and forth continuously to let you know the robot is ready for you to use. The sensors monitored and their functions are described in Table 14.1. When the program reaches the end of this loop, it is vectored back to the beginning through the address contained in memory locations $6D,6E (USRLIV). You can change these locations to point to your program so you can add additional monitoring functions. This vector will be reset by LMOS on a cold start but not on a warm reset.

In addition to this life-like external and internal system monitoring performed by the robot, there is a life sustaining monitoring function built into the robot. Should any one of its batteries fall below a critical level (10.5 volts), the hardware circuitry will force a non-maskable interrupt (NMI) of the main computer's CPU. This will force program control to a service routine which will cause the robot to issue a warning that it will return to its charger in one minute unless you press one of the Function Keys or corresponding Keyboard key. If you do press a key, the robot will not reset its NMI flip/flop and you will, therefore, not get another

TABLE 14.1

LIFE LOOP MONITORING FUNCTIONS

SENSOR/PERIPHERAL SCANNED	ROBOT'S ACTION TAKEN WHEN DEVICE IS ACTIVE
Smoke Detector	Sounds bell until smoke disappears
Function Key	As indicated on LIFE MENU: 1 -- Jump to DEMO Program 2 -- Jump to SCHEDULER Program 3 -- Jump to BASIC Program 4 -- Jump to Remote Communications System Computer Central Program 5 -- Jump to System Monitor Command Program
Keyboard	Keys 1-5 have same function as Function Keys 1-5. Keys A-U will run individual system check routines, as listed in Table 14.2.
Sound Detector	Enters VOice COmmand Language (VOCOL) if installed when sound level A/D reading exceeds value contained in the variable SNDLVL ($0570).
RCS Carrier Detector	Enters RCS computer communications software program
Real-Time Clock	Run program entered by SCHEDULER at designated time
Battery Levels	Report battery voltage when any one A/D reading falls below value contained in the variable BATLVL ($0800) for five consecutive minutes.

warning. You may endanger the life of the robot´s batteries if you do this. If you do not press a key within one minute, the robot will return to its charger, reset the NMI flip/flop and return to the Living Loop. You may intercept the NMI by changing memory locations $16,17 (NMILOC) to point to the beginning of your program.

When you first power up a new GEMINI robot, or when you hold Function Key 5 down and press the RESET Key (a forced cold start), the robot will run a series of programs designed to test out its systems. The robot does not have the abilities to check out all of its systems by itself so it will require your assistance and attention. The system checks performed during power up/cold start are described in Table 14.2 which also indicates the individual keys to press to get the robot to perform the corresponding check from the Living Loop.

After the system checks have been performed, the robot will run the DEMO program (provided VIOS and the keyboard are working). When you are finished with the DEMO, GEMINI will ask if you wish to enter navigational room data. If so, program control is routed to the program which handles this function; otherwise, the robot enters its Living Loop. You may enter the room data at any time simply by pressing key `R´ on the keyboard while the robot is in its Living Loop.

TABLE 14.2

LMOS ROBOT SYSTEMS CHECK

KEY	OPERATION
A	Check VIOS
B	Check Batteries
C	Check ROM´S on ROM Card
D	Check RAM (except $C000-DFFF)
E	Check for Mass Storage Device; boot GEMDOS/GEMTOS if available
F	Check Keyboard
G	Check Real-Time Clock (only if Keyboard is working
H	Check PROCON
I	Check Base Bumpers
J	Check Sonars
K	Check Head
L	Check Room Beacon Detector (only if head is working)
M	Check Door Edge Reflector Detector (only if head is working)
N	Check Motion Detector (only if head is working)
O	*Speak Temperature
P	*Speak Time
Q	*Speak Pressure
R	*Enter Room Table Entry Program
S	*Enter VOCOL

* Not performed during power up or forced cold start

COLD/WARM START VARIABLE SETTINGS

There are some variables which are set only once during power up/cold start. Some of the variables are changed by the robot during its life. Others may be changed by you to your liking so further warm resets will not affect them. These variables are given in Table 14.3, along with their functions.

There are also variables which are set after every warm reset. These variables are critical to the "life" of the robot. They are described in Table 14.4. The variables located in $0600-0635 are Fractional Integer Math Pack constants used by the Navigation routines, as described under SYSTEM MONITOR. If you use Math Pack, you should be careful not to destroy these constants and you should not use the RESERVED variables.

JUMP TABLE ROUTINES

There are certain routines in the LMOS frequently used by other BANK ROM´s, which were assigned to a JUMP table during program development. You may use these entry points with the assurance that future possible versions of LMOS will retain these entry points. The entry points, and their requirements for proper running of the routines, are provided in Table 14.5.

TABLE 14.3

POWER-UP/COLD START VARIABLES

VARIABLE NAME	LOCATION	FUNCTION
USRLIV	$6D,6E	Set to beginning of Living Loop (LVLP - $C2BD)
USRVOC	$71,72	Set to beginning of VOCOL ($C000)
SNDLVL	$0570	Threshold A/D sound channel reading for VOCOL access ($D0)
BATLVL	$0800	Threshold A/D battery channel readings for first level warnings ($C0=11.25V)

TABLE 14.4

WARM START VARIABLES

VARIABLE NAME	LOCATION	FUNCTION
LIFEAD	$6F,70	Default set to MNLVLP ($C29D) used by other programs to return to Living Loop for alternate route control
MSPSTP	$0567	Head stepper speed; default $0F
SPEED	$0571	Procon Motor Speed (ASCII 8)
NMILOC	$16,17	Non-Maskable Interrupt vector; default is $0864. An important part of the NMI service routine is copied from ROM to RAM beginning here
CLKIRQ	$1A,1B	RTC alarm interrupt; default is $C22E in LMOS ROM. Used to provide one minute interrupt for checking SCHEDULER generated tasks
VAR	$0600-0635	Integer Math Pack Constants used by Navigation
DISFAC	$0600-0602	Distance factor used to convert sonic microsecond ranges to PROCON translation commands

DEGFAC	$0603-0605	Degree factor used to convert degree commands to PROCON rotation commands
N.0.5	$0606-0608	The constant 0.5
N.1.118	$0609-060B	" " 1.118.
N.1.5	$060F-0611	" " 1.5
N.10	$060F-0611	" " 10
N.30	$0612-0614	" " 30
N.63.4	$0615-0617	" " 63.4
N.75	$0618-061A	" " 75
N.90	$061B-061D	" " 90
N.180	$061E-0620	" " 180
N.360	$0621-0623	" " 360
CHGDST	$0624-0626	Distance from head sonar to wall when at charger +0.5 ft. (in sonic microseconds)
N.1FT	$0627-0629	The constant 1 FT (in sonic microseconds)
N.2FT	$062A-062C	The constant 2 FT (in sonic microseconds)
N.4FT	$062D-062F	The constant 4 FT (in sonic microseconds)
NDPR	$0630-0633	Number of degrees per radian
RESERVED	$0634-06A4	Navigation variable space

TABLE 14.5

LMOS JUMP TABLE ROUTINES

HEXIDECIMAL ENTRY ADDRESS	ROUTINE NAME	USE & ENTRY REQUIREMENTS
C000	INITO	Cold/warm start LMOS. Cold start if bit 0 of ROSTAT ($74) is clear.
C003	JMP (LIFEAD)	Indirect JUMP through $6F,6E. Normally, to MNLVLP, the cold entry into the Living Loop.
C006	CHK.BATS	Speak and output (on LCD) the voltages of the three batteries. VIOS must be enabled and in Text-to-Speech Mode on entry.
C009	DEMO	Enter DEMO Program. VIOS enabled and in Text-to-Speech Mode.
C00C	POEMS5	Speak and output (on LCD) five randomly generated poems. VIOS enabled and in Text-to-Speech Mode.
C00F	STRYS5	Speak and output (on LCD) five randomly generated sentences. VIOS enabled and in Text-to-Speech Mode.
C012	ENPROC	Enable PROCON
C015	DSPROC	Disable PROCON
C018	ENVIOS	Enable VIOS
C01B	DSVIOS	Disable VIOS

C01E	DO.TIME	Speak current time and sound gong. VIOS enabled and in Text-to-Speech Mode.
C021	TEMPERATURE	Speak and output the temperature in degrees C. VIOS enabled and in Text-to-Speech Mode.
C024	BAROMETER	Speak and output (on LCD) the Barometric Pressure in millibars. VIOS enabled and in Text-to-Speech Mode.
C027	DETECT.SMOKE	Check for smoke. SMKFLG ($0E1A)=00 if smoke; FF otherwise. Smoke detector must be enabled (Bit 6 of $E060 set).
C02A	SPEAK.A.POEM	Same as POEMS5 except only one poem.
C02D	SPEAK.A.SENTENCE	Same as STRYS5 except only one sentence.

CHAPTER 15

SCHEDULER

BANK 2 ROM contains a very powerful and extremely easy to use program for entering a schedule of activities for the robot to perform at future times in prescribed locations. The program is menu driven, and contains two principal selection menus.

The first menu provides options to: (1) add a schedule; (2) list and delete entries; (3) add BSR module names; (4) list and delete module names; (5) add room names; (6) list and delete room names, and (7) exit (to the Living Loop). The second menu provides several options when the "add a schedule" option from the first menu is selected. These options include: (1) wake up calls; (2) security mode configuration; (3) poetry; (4) stories; (5) songs; (6) daily reminders; (7) BSR module commands; (8) random or autonomous behavior; (9) sensor reports; (10) movement from one room to another; (11) home (go to charger), and (12) run a BASIC program.

There are two principal parts to the SCHEDULER and each has its own entry point that is used by LMOS. The first part provides the menu driven data entry capabilities just described. The second part provides the means of executing the data files created by the first part. LMOS enters the first part (SCHEDLO - $C000) when you push Key 2. It enters the second part (TIMED.TASK,HANDLER - $C003) every minute to determine if there is a schedule task to be executed.

Data files created by the first part of the program are built downward in memory, starting at the address contained in locations $79,7A (SCHLHM). As data files are entered, SCHLHM is appropriately decremented. Should SCHLHM reach SCHLLM ($7B,7C), an OUT OF MEMORY warning will be issued.

There is a JUMP table at the beginning of the ROM which provides access to certain routines which may be useful to you. The entry address, name, and entry requirements of these routines are given in Table 15.1.

The SCHEDULER makes extensive use of the bank switching routines contained in the System Monitor, since it builds its data files (in a 56K RAM version) starting in RAM which occupies its same address space. You will find many interesting and educational subroutines in this program from a study of the Source Code available from ARCTEC SYSTEMS.

TABLE 15.1

SCHEDULER JUMP TABLE ROUTINES

ENTRY ADDRESS	ROUTING NAME	USE & ENTRY REQUIREMENTS
C000	SCHDLO	Main entry point to the build file subprogram. No entry requirements.
C003	TIMED.TASK. HANDLER	Main entry point to subprogram which executes data files. No entry requirements.
C006	SPKMSG	Speak a message pointed to by contents of MSGNUM ($77,78). No entry requirements.
C009	DOPOET	Generate, speak and display on LCD five random poems. No entry requirements.

COOC	DO.SNG	Play the song given by the number in TEMP ($0880).

TEMP	SONG
0	STAR SPANGLED BANNER
1	HAPPY BIRTHDAY
2	HAPPY ANNIVERSARY
3	JINGLE BELLS
4	EASTER PARADE
5	AULD LANG SYNE
6	WHEN IRISH EYES ARE SMILING
7	STAR WARS
8	ROBOTS ARE A MAN'S BEST FRIEND
9	MUSIC BOX DANCER
10	THE ENTERTAINER
11	DAISY, DAISY
12	RAINDROPS KEEP FALLING ON MY HEAD
13	HAVAH NAGILAH
14	O CANADA

COOF	DOSTOR	Generate, speak and display on LCD, five random space travel related sentences.

CHAPTER 16

NAVIGATION SYSTEM

BANK 4 ROM contains a set of programs, beginning at $C5B0 (part of the BASIC interpreter also occupies this ROM), which you will probably find to be the most interesting part of GEMINI. They were the most difficult routines to write and required considerable experimentation and effort to develop them to their present state.

There are two major programs which make up the Navigation System routines. The first routine, called MOVE.TO.RMN ($C5BF), will move the robot from its current room (CURRM-$0997) to the desired room (DESRM-$0999) using the shortest path (least number of doors) found from a search of the room´s navigation data contained in RMTBLS ($0700-07C3). This program calls numberous subroutines which will be described below.

The second major program in the Navigation System software is called HOME ($C5B3). This routine first determines which room the robot is in (sets CURRM) and, knowing that DESRM = 01, calls MOVE.TO.RMN to get it there. It then proceeds to call several routines required to dock the robot to its charger.

ROOM TABLE DATA FORMAT

Before these routines can function, it is necessary to have all hardware (Door Edge Reflectors and Room Beacons) installed, and data concerning the layout of the home or office in the room tables. The format for this data is given in Table 16.1. Note that there are unused bits for additional information, should you desire to experiment with these routines. A routine in LMOS provides an easy way to enter this data. As you gain experience with GEMINI, you may wish to enter this data directly into memory from the monitor.

TABLE 16.1

ROOM TABLE DATA FORMAT ($0700-07C3)

BYTE	MEANING
0	Lo byte of address of next table entry.
1	Hi byte of address of next table entry.
2	Beacon code/room/hallway number (and other information as follows):

BIT	USE
7	1 = All reflectors can be seen from near beacon 0 = All reflectors can be seen from room center
6	1 = "hallway" 0 = "normal" room
5	Unused
4	Unused
0-3	Hex room beacon code 1-F (1 = Charger Room)

BYTE	MEANING
3	Number of doors in this room.
4	Width of door #1 in sonic us clockwise around room facing beacon (in inches)
5	Beacon code of room on other side of this door and reflector locations.

BIT	USE
7	1 = Reflector on right side of door 0 = Reflector on left side of door
4-6	Unused
0-3	Hex room beacon code

BYTE	MEANING
6	Same as Byte 4 for Door #2
7	Same as Byte 5 for Door #2

NOTE: Last entry in table must be designated with 00 bytes for Bytes 0 and 1 (address of next table entry).

DEGREE FACTOR DATA TABLE

Surfaces vary from room to room in a typical home or office and this affects the turning accuracy of GEMINI. The navigation routines have a "learning" mechanism built into them which cause the robot to compute the degree factor when it enters a room for the first time. This information is stored in the Degree Factor Data Table at 07C4-07FF. Four bytes for each room are alloted and provide the following:

BYTE	USE
0	1 = Degree factor has been computed. 0 = Degree factor not known.
1	Fractional part of degree factor.
2	Lo byte of degree factor.
3	Hi byte of degree factor.

Note the degree factor follows the 3-byte data format of the Fractional Integer Math Pack described in the System Monitor.

PRINCIPAL OF GEMINI´S NAVIGATION SYSTEM

A brief explanation of the navigation concepts used on GEMINI will help you better understand and use the Navigational System subroutines. The robot´s height, width, conical shape, and location of sensors were all carefully designed, based on a study of a typical home/office environment and the principles under which humans navigate. The robot uses the room beacons to find out what room it is in much like we use our eyes and past experience to do the same thing. The robot uses data in the room tables to determine connectivity relationships between rooms.

Door edge reflectors were found necessary to practically implement a reliable navigation system with inexpensive (relatively) sonar ranging systems, because of the wide beam width of the sonars. If the sonar module is more than approximately five feet away from an open doorway 28"-30" wide, the sonar ranger data cannot be used to detect door edges. To get around this problem, door edge reflectors are used. These reflectors also improve the robot's navigational efficiency by making it relatively easy to count the doors in a room for comparison with the room table data.

When the robot is directed (or directs itself) to move to another room, it first determines what room it is in by making a full 359 degree scan of the room until the beacon is found. Once found, it extracts the code and saves this in CURRM. During this scan, the robot also counts the number of door reflectors and compares this with the data given for CURRM in the room tables. If this does not match, the robot will move to another position and repeat the search.

The robot then uses this information to perform a breadth first graph search of the room tables to find a "path" of least number of doors to the desired room (DESRM). This path (PATH-$02C2 & up) consists of the door number in use of CURRM, room number on the other side of this door, door number to use in that room, etc.

The robot then turns its head to the designated door (determined by counting clockwise from the beacon) and proceeds to align its body so it squarely faces the reflector.

The robot then takes ranges with its head sonar to determine how far it must move to get within four feet from the door. It orders PROCON to move the computed distances and, while PROCON is doing this, it tracks the reflector with its head while ranging continuously with its front collision avoidance sonars.

If, due to an obstacle, the robot cannot make it to within four feet of the reflector by following a straight line, it will try to find a path around it. In order to do this, it will move its entire body while taking a series of ranges with its front sonars and calculate the course which deviates the least from its present course.

Once this course has been found, the robot turns its head to the reflector, orders PROCON to move half of the new clear distance, tracks the beacon with its head and watches for collison with its body sonars and base bumpers.

The robot then tries again to make it to within four feet of the door reflector. If necessary, the process is repeated until the robot finds itself within the desired range of the door. However, if it determines that a clear path is not possible, then it abandons the attempt and returns an error code to the calling program.

Assuming that a clear path is found, the robot then uses the right/left side information about the door reflector to calculate a position approximately one door width in front of the door. This is accomplished by first determining the included angle of door edges using the head ranger and the ranges to these edges. This information is used to compute the new position using the law-of-sines, cosines and tangents for oblige triangles. The computations are extensive but are performed very quickly and without any noticeable delay.

The robot moves to that position and uses angle data to the reflector to make precise turns and neccessary corrections. Once in front of the door, the robot takes a new set of ranges which will be used to: (1) fine tune its passage course through the doorway; and (2) to make certain the door is open.

The robot proceeds to the doorway center and then moves two feet into the next room. There it stops and makes a limited scan of the new room for its beacon. If it cannot find the beacon, it will move another two feet into the room and try again. If there is still no beacon to be found, the robot will back up the exact distance it went forward and try another door. If the beacon was found, the robot proceeds to scan that room and the process repeats until it reaches its destination.

Although GEMINI is not as fast as we are, it is an incredible experience watching the robot navigate. Table 16.2 gives a brief description of the routines used to perform the above tasks. You can call these routines directly from your program, or you may copy them into a RAM chip on the "ROM" EXPANSION CARD so you can alter and experiment with them. We highly recommend you obtain a copy of the source code for these programs.

TABLE 16.2

NAVIGATION SYSTEM SUBROUTINES

ENTRY ADDRESS	ROUTING NAME	FUNCTION
CB68	MOVE.TO.RMN	Move robot to room contained in DESRM.
CC21	LEAVE.CHARGER	Move robot back 2′ from charger. Check if beacon giving code 01; if so, exit with C=0. Else go back on charger and exit with C=1.
C9C7	FIND.RM.BEC	Search head position A-reg (lo) to X-reg (hi). Find RMCODE and BECANG. Exit with head at BECANG & C=0. C=1 if not found.
D09A	MOVE.TO.RM.CTR	Find approximate room center from head ranges. Check with backup ranges and then move there.
CC61	FIND.PATH	Find most efficient PATH between CURRM and DESRM using RMTBL data.
CE73	ENTER.ADJRM	Move robot from CURRM room to ADJRM.
CF4B	GET.DEG.TAB	Fetch degree factor ($0603) for this room from Degree Factor Table.
D082	GET.DOOR.COUNT	Fetch door count for CURRM in A-reg.
C865	SEN.SCAN	Search X-reg steps on either side of A-reg for door reflectors.

D3BC	MOVE.4FT.FM	Move the robot to within 4 ft. from door edge reflector to beacon.
C7D7	FIND.DOOR.EDGES	Search the sector from A-reg to X-reg and find NDREDS and order of doors in DEBUF.
D997	DOOR.SECTOR.SCAN	Find door edges and angles using sonar.
D36D	REPOSITION	Turn robot 75 degrees in specified direction.
D1FD	ORDER.DOORS	Order door edges found and arrange in CW direction from beacons.
D5C0	MOVE.TO.DR.CTR	Solve for position using data for DOOR.SECTOR.SCAN and move thru.
D23D	MOVE.THRU. DOORWAY	Move robot thru door and 2 ft. inside next room.
D6E4	PASS.THRU.DOOR	Position robot in center of door if door is open.
C62A	TRACK.AVOID	Track the door edge reflector or beacons while ranging with body sonars.
DBF0	HOME	Move robot from CURRM and place on charger.
DC71	HOOK.UP	Dock robot on charger.

APPENDIX

LIST OF FIGURES

FIGURE 2.1	EXTERIOR ABS PLASTIC SHELLS p.7
FIGURE 2.2	INTERIOR STRUCTURE FRONT VIEW p.8
FIGURE 2.3	INTERIOR STRUCTURE REAR VIEW p.9
FIGURE 3.1	DRIVE SYSTEM SCHEMATIC p.12
FIGURE 3.2	BASE ASSEMBLY SCHEMATIC p.13
FIGURE 4.1	CHARGE/DISCHARGE CYCLE LIFE p.15
FIGURE 6.1	HEAD ASSEMBLY - FRONT VIEW p.27
FIGURE 7.1	SONAR FIELDS OF VIEW -TOP VIEW p.34
FIGURE 10.1	EXPANSION CARD CONNECTOR PINOUTS p.78
FIGURE 10.2	EXPANSION SLOT SIGNAL TIMING p.88
FIGURE 11.1	LIQUID CRYSTAL DISPLAY p.98

LIST OF TABLES

TABLE 5.1	PROCON COMMANDS p.22
TABLE 10.1	EXPANSION SLOT SIGNAL DESCRIPTIONS p.79
TABLE 10.1A	EXPANSION SLOT I/O SELECT ADDRESSES p.85
TABLE 10.1B	EXPANSION SLOT ROM/RAM BOARD BIT SELECTIONS p.85
TABLE 10.1C	EXPANSION SLOT ANALOG TO DIGITAL PORT ASSIGNMENTS p.86
TABLE 10.1D	EXPANSION SLOT RTC SQUARE WAVE FREQUENCIES p.86
TABLE 10.1E	EXPANSION SLOT DEVICE SELECT ADDRESSES p.87
TABLE 11.1	KEYS AND THEIR TRANSMITTED ASCII CODES p.91
TABLE 11.2	LCD CHARACTER CODE MAP p.95
TABLE 11.3	LCD COMMANDS p.96
TABLE 12.1	RAM USAGE MAP (56K CONFIGURATION) p.102
TABLE 12.2	RAM USAGE MAP FOR VARIOUS RAM CONFIGURATIONS p.103
TABLE 12.3	RESERVED RAM UTILIZATION p.104
TABLE 12.4	ZERO PAGE RAM UTILIZATION p.105
TABLE 13.1	SOME USEFUL MONITOR SUBROUTINES p.109

TABLE 13.2 MONITOR COMMAND SUMMARY p.116

TABLE 13.3 INTEGER MATH PACK ROUTINES p.124

TABLE 13.4 HEAD STEPPER FLAGS AND VARIABLES p.129

TABLE 14.1 LIFE LOOP MONITORING FUNCTIONS p.131

TABLE 14.2 LMOS ROBOT SYSTEMS CHECK p.133

TABLE 14.3 POWER-UP/COLD START VARIABLES p.135

TABLE 14.4 WARM START VARIABLES p.136

TABLE 14.5 LMOS JUMP TABLE ROUTINES p.138

TABLE 15.1 SCHEDULER JUMP TABLE ROUTINES p.141

TABLE 16.1 ROOM TABLE FORMAT DATA ($0700-07C3) p.144

TABLE 16.2 NAVIGATION SYSTEM SUBROUTINES p.149