RB5X: MISSION TO MARS

AN INTERPLANETARY ADVENTURE FOR ALL GRADE LEVELS

Teacher's Manual



GENERAL ROBOTICS CORPORATION

1978 South Garrison Street, Unit 6 ■ Lakewood, Colorado 80227-2243

BLANK PAGE

RB5X: MISSION TO MARS

An Interplanetary Adventure for all Grade Levels

Teacher's Manual

Version 1.0, May 2000
General Robotics Corporation

General Robotics Corporation
1978 South Garrison Street, Suite 6
Lakewood, Colorado 80227-2243

Toll-Free: 800.422.4265
Local: 303.988.5636
Fax: 303.988.5303

Information: rb5x@generalrobotics.com
Technical Support: support@generalrobotics.com

Products & Corporate Information: www.generalrobotics.com
The RB5X and Education: www.edurobot.com

"RB5X," "RB5X The Intelligent Robot" and "RCL" (Robot Control Language) are trademarks of General Robotics Corporation.
"NSC Tiny Basic" is a trademark of National Semiconductor Corporation.

COVER PHOTO: The RB5X robot is superimposed over the desert landscape of Chryse Planitia, latitude 25N, longitude 45W, on Mars. In an effort to make the image realistic, RB5X's shadow, and the glare on its dome, have been adjusted for the position of the Sun at the time the photo was taken. This is how RB5X would really appear on the surface of Mars!

The ground in this area is strewn with rocks ejected from volcanoes, and has numerous small sand dunes that shift daily with the winds. The large rock, nicknamed "Big Joe," is about 2 meters in diameter. This image (minus RB5X, of course) was radioed back to Earth by the Viking 1 Lander on June 21st, 1976. Photo courtesy of NASA/JPL.

About the Author

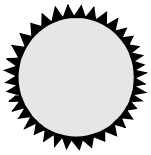
Dan Heim is a freelance consultant in educational multimedia and energy efficiency, and writes instructional materials in the fields of astronomy, physics and ecology. As a teacher of science for 25 years he enjoys a practiced insight into how students learn. That experience and insight was essential to the creation of RB5X: Mission to Mars.

Dan and his wife Sandi live at Heimhenge, a "home of the future" located in the Sonoran desert near New River, Arizona. Earth-sheltered in the foothills of rim country, their home incorporates passive solar design, photovoltaic electricity, high-tech insulation and energy-efficient appliances. The HVAC energy demand at Heimhenge is half that of a conventional structure.

RB5X: Mission to Mars was produced on a computer powered by free solar energy. All electrons were recycled, and no living creatures were harmed in the process.

They don't yet have a robot butler, but when they do it will be a solar-powered RB5X. Until then, Dan will continue answering his email personally at: heimhenge@compuserve.com.

May 2000



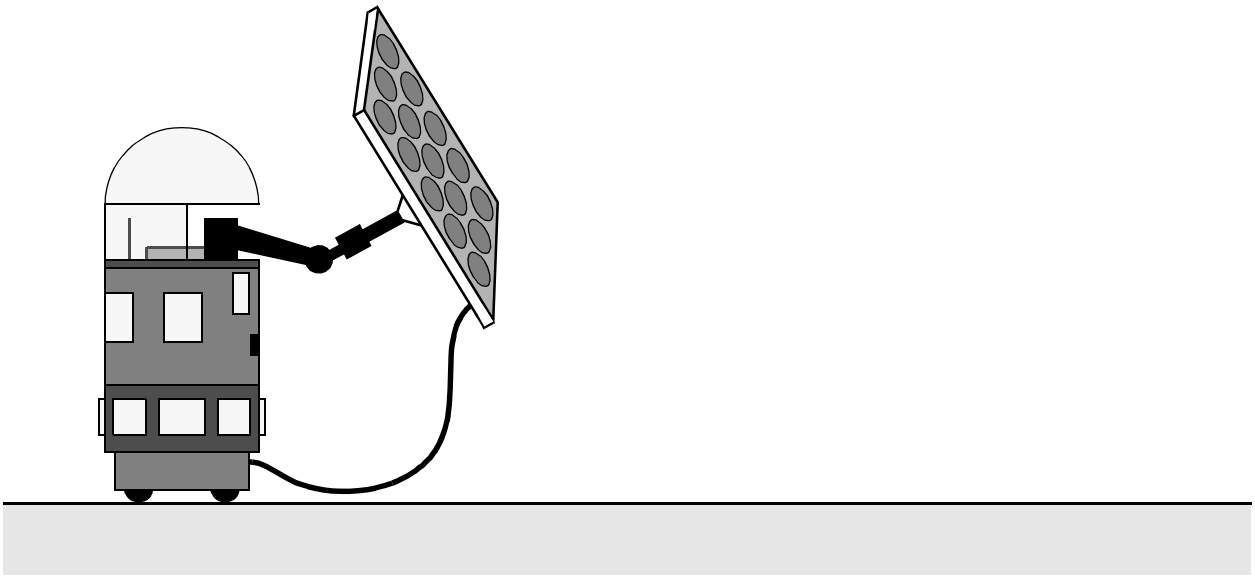


Table of Contents

		page
Mission Briefing:		**7**
To the Teacher:		**9**
Using the Manual:		
Grades K-6		**10**
Grades 7-12		**11**
Grades 12+		**12**
Preparation and Planning:		**13**
Sample Diorama		**15**
RCL Programming — Setup and Methods:		**16**
Initial Robot Calibration:		**30**
URL's You Can Use:		**35**
Glossary:		**36**
Mission Solutions:		
An Important Message Regarding Solutions:		**40**
Mission 1:	Creating the Mars Diorama	**41**
Mission 2:	Robot Calibration	**44**
Mission 3:	Mapping	**51**
Mission 4:	Hazard Avoidance	**60**
Mission 5:	Navigation and Piloting	**67**
Mission 6:	Energy Management	**77**
Mission 7:	Sample Acquisition	**83**

BLANK PAGE

Mission Briefing

Introduction:

NASA's recent success with Mars Pathfinder and Sojourner proved that robotic exploration of distant worlds, once the realm of science fiction, is now a valuable tool of science. Earlier missions to the Moon and planets used machines in place of astronauts quite effectively. The Viking landers on Mars (1976) had working arms that allowed them to do experiments in *exogeology* and *exobiology*. But these earlier robots lacked the mobility and *autonomy* of the Sojourner Rover. Sojourner was controlled from a computer workstation at JPL. The mission "pilots" drove Sojourner using a joystick and keyboard while "seeing" on the monitor through the robot's camera eyes. This type of remote control is known as *teleoperation* with *telepresence*, since the pilots see and reacts as though they were actually there. This is not to be confused with *virtual reality* systems; the world on the other end of the line at JPL was reality.

Your RB5X can duplicate many of the functions of Sojourner and perform a few additional tasks as well. In this series of Missions your class will set up a simulated Mars landscape for RB5X to explore. Along the way you will learn about the nuts and bolts of remote robotic exploration, do some real creative problem-solving and become intimately familiar with one of our nearest neighbors in space — planet Mars.

Assigned Missions:

Creating the Mars Diorama (an exercise in modeling and exogeology)
Robot Calibration (engineering and applied mathematics)
Mapping (exploring and charting unknown territory)
Navigation and Piloting (using maps and teleoperation)
Hazard Avoidance (safely negotiating a hostile environment)
Energy Management (planning exploration within energy constraints)
Sample Acquisition (identifying and retrieving rock samples)

Required Materials:

Your RB5X is ready and willing to assist you in this simulated exploration of Mars. Grade levels K-12 will need no special materials other than a few inexpensive common household items. For post-secondary classes, access to basic tools and construction materials is required. Missions are designed for the base RB5X package and do not require the purchase of any RB5X accessories. If you have the voice/speech synthesis option, *RF* com-link or video camera, or robotic arm, you will find relate exercises and suggested applications included in most Missions.

Caution!

Don't neglect reading these sidebars. The Student Manual has quite a few of them. They'll tell you things you want to know, didn't know, or absolutely need to know ... just like the three examples you see on this page.

Did You Know?

*The letters JPL stand for the **J**et **P**ropulsion **L**aboratories, located in Pasadena, CA. JPL is an operating division of Caltech (California Institute of Technology), and a major contractor with NASA since the early days of space science.*

FAQ:

***Q:** Why are some of the words I'm reading printed in italics?*

***A:** Words that might be unfamiliar have been italicized, and are defined in the Glossary (p. 36). In the online version of your Student Manual the first occurrence of each such word is a Glossary hyperlink. Margin notes like this are always in italics.*

Did You Know?

The calculations for interplanetary space travel were first done by Walter Hohmann, in his book titled: The Possibility of Reaching Celestial Bodies (1925).

Did You Know?

To reach Mars from Earth a rocket must achieve a speed of 11,200 m/s. That's about 25,000 mph!

FAQ:

Q: *Ok, I'm getting used to the margin notes, but it seems as if there's still a lot of unused space on some pages. Isn't this a real waste of paper?*

A: *Marginal notes are typically added to subsequent editions in response to feedback and questions. Building this space into the document now makes later revisions easy.*

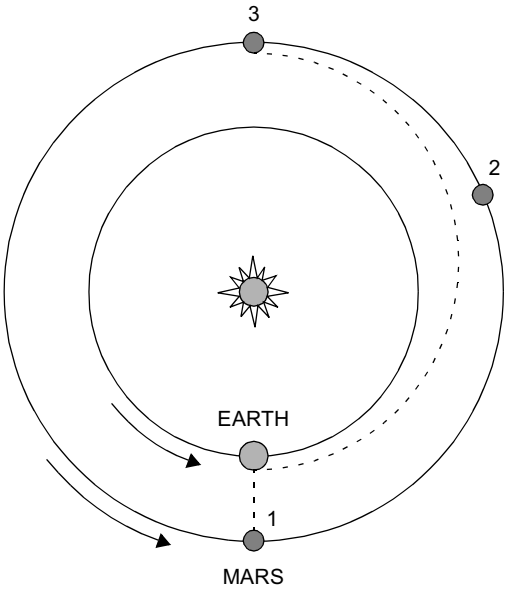
And until the margin is completely full, feel free to use this space for your own notes.

The Flight Plan:

Space travel takes a lot of fuel.

Earth is 150 million kilometers from the Sun; Mars 228 million kilometers. The 78 "Mkm" gap cannot be navigated easily, as the planets move at different speeds. The fastest possible path (to 1) would require more fuel than can be carried by current spacecraft. The long path (to 3) requires the least fuel, but would take about 8½ months one-way and would launch when Mars is at (2).

This interplanetary maneuver is known as the Hohmann Transfer.



The Delta II:

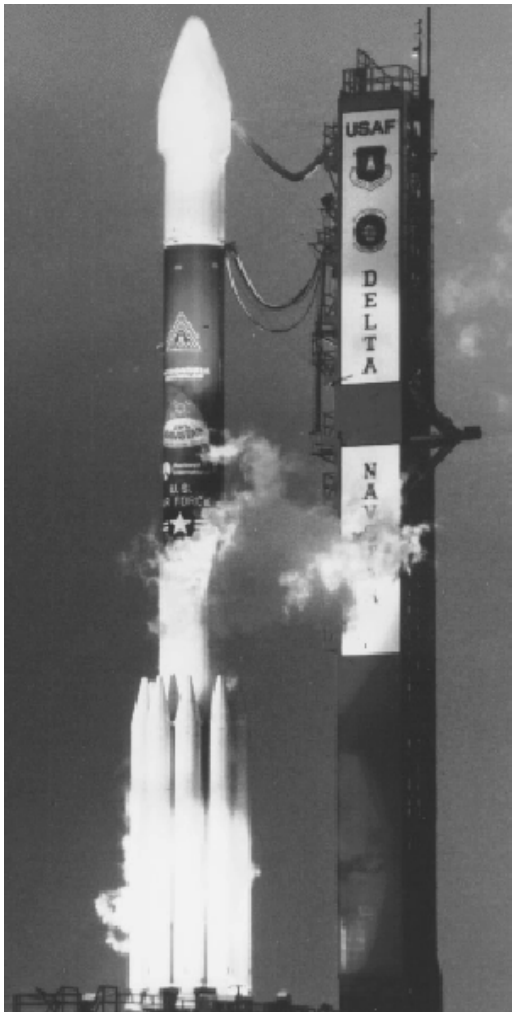
This photo shows the launch of a Delta II rocket, currently our only vehicle capable of the 78 million kilometer journey to Mars.

The rocket is 38 meters tall and, at its waist, 2 meters in diameter. Its engines produce a total thrust of 2.63 MN (mega-*newtons*) or approximately 600,000 pounds.

The rocket reaches a top speed of about 11.2 km/s; still, the journey to Mars lasts 8-9 months.

The payload (satellite or robotic explorer) is housed in the white conical shell at the top. The eight devices arranged around the base are solid-fueled rocket *boosters*.

The Delta II is made by McDonnell Douglas Space Systems.



DELTA II IMAGE COURTESY OF JPL/CALTECH/NASA

To the Teacher

Special Note:

If you skipped over the Mission Briefing, please go back 2 pages and read it before continuing.

Long and Short-Range Learning Objectives

Students of all ages will benefit from Mission to Mars. Expect improvements in general problem solving skills, interdisciplinary insights and spacial visualization. At the fundamentals level, the discovery of real-world applications for mathematics and science will increase student appreciation for the tools and methods of these disciplines, and their skills in the use thereof. Engaging and thought-provoking essay questions provide numerous opportunities for creative writing.

And, of course, as with all RB5X activities, your students will be motivated by the prospect of seeing their ideas carried out by a mobile, autonomous machine.

- **Mission Questions** give students a chance to apply skills acquired from other academic areas. Watch their "thought-bulbs" light as they discover these interdisciplinary connections.
- The open-ended **Mission Problems** are a challenge and a stimulus to their creative talents. Working alone or in groups, the evolution and interplay of ideas will foster verbal and written communication and problem-solving skills.
- **Exploring Further** is where your more technologically advanced students can be challenged to prove their worth. You will hear no complaints of "Boring!" with these assignments.

This is what "active learning" is all about — getting the students involved in their own education. Its superiority to traditional modes of instruction is well supported in theory and practice. The old "straight lecture" approach is giving way to newer methods more effective with today's students. Modern curricula must recognize visual, tactile and kinesthetic experiences as complementary to traditional "oral-to-aural downloads."

Heuristics, the study of general problem-solving methods, is only rarely addressed in traditional curricula. This is ironic, and one of the great failings of our educational system. Heuristics provides a strategy for finding the solution to any problem, regardless of subject area. That such a strategy even exists may come as a surprise to some, but it's been around for awhile, and is really just a generalized variation of "The Scientific Method."

Mission to Mars is an excellent activity for developing students' heuristic skills; this should be emphasized to them at the start. Appendix 2 (The Heuristic Method) should be distributed to all students regardless of how else you use this material.

Mission to Mars provides you the luxury of working with material that is intrinsically exciting to the technologically adept students in our 21st century classrooms — robotics, computers and the exploration of other worlds. This curriculum package includes everything you need to build a highly effective course of study around your RB5X.

Using the Manual: Grades K-6

Prerequisite Knowledge

Students should be familiar with RB5X and its basic capabilities.

Process

The Student Manual rated 10.2 on the Dale-Chall readability test, thus it should not be used for assigned readings in grades K-6. Only the Teacher's Manual is required; it includes all material from the Student Manual plus guidelines and solutions.

You should read the Introduction and Scenario for each Mission to your class, simplifying terminology and providing additional explanation as needed. Encourage questions and discussion, as this will better set the mood and increase motivation for the actual problem-solving process.

Overhead transparencies printed from the file IMAGES.PDF will clarify explanations of many concepts, and also provide visual material that adds to class discussions. The same is true of the many images contained in the Mission Solutions section. You may print them and use them.

Students should answer the Mission Questions either verbally or in writing, and as algorithms or flowcharts. Since the computer code solutions to Mission Problems are provided, you may want to program RB5X to demonstrate these solutions in action.

The Appendix includes material suitable for copying and distribution to your students as part of the Mission activities. At this grade level you must be prepared to act as translator and facilitator. You may even create your own activities using the handouts. As with all learning experiences, it is the innovation of the individual teacher, acting within sound pedagogical guidelines, that makes all the difference. Mission to Mars is a flexible curriculum package and will adapt well to your individual needs, as well as to those of your students.

Behavioral Objectives

The students will learn the physical differences between Earth and Mars, and come to appreciate the environmental factors that make our planet unique in the Solar System. They will be able to use the basic vocabulary of exogeology and teleoperation. They will understand how the environment constrains our options, and why "what is possible" is not necessarily the same as "what is desirable." They will recognize the importance of stating instructions clearly, and its implications for successful problem-solving and cooperation between humans and machines.

Using the Manual: Grades 7-12

Prerequisite Knowledge

Students should have completed our "Programming the RB5X Robot" curriculum, and have basic algebra, geometry and research skills.

Process

The Student Manual rated 10.2 on the Dale-Chall readability test, and is targeted mainly for this grade level. It can be used as-is, and provides all the materials needed for full student engagement and individualized study.

Mission Questions are best used for generating class discussion, and will also promote insights applicable to solving the Mission Problem and Exploring Further exercises.

The Mission Problems are your students' primary tasks. Most involve writing and implementing computer code that allows RB5X to accomplish stated objectives. Depending on your students' programming skills, the creation of an algorithm or flowchart could also be assigned.

Exploring Further is an excellent source for extra credit or exam problems. You can also use it for in-class "think tank" sessions, with verbal or written solutions shared and critiqued as follow-up.

As with most heuristic exercises, the given solutions to Mission Problems are not unique. You are encouraged to post innovative student solutions on our website to share with other educators.

Behavioral Objectives

In addition to the K-6 objectives, students will be able to use cartographic coordinate systems, and create kinematic models for robot operation within those systems. Students will improve in their ability to discover solutions while working under externally imposed constraints. They will also understand how the Laws of Nature set limits on what is possible, and develop optimization and efficiency procedures to avoid those limits. Student will learn how to use geometric structures as models for the physical world.

Using the Manual: Grades 12+

Prerequisite Knowledge

Students should have completed our "Programming the RB5X Robot" curriculum, have strong algebra, geometry and research skills, and be familiar with basic tools and fabrication processes.

Process

The Student Manual rated 10.2 on the Dale-Chall readability test, thus it is well within the reading ability of your students. It can be used as-is, and provides all the materials needed for full student engagement and individualized study.

The content of this curriculum is applicable to courses in basic programming, robotics, computer science, industrial design and astronomy. Specific needs may best be met by selecting individual Missions to use within your overall lesson plan.

For example, you could easily skip Mission 1 as a class activity, create the diorama ahead of time on your own, and proceed to Mission 2. Mission 1 activities do not directly address behavioral objectives for your grade level.

Mission Questions can be used for generating class discussion and will promote useful insights for solution of the Mission Problem or Exploring Further.

The Mission Problem works well as a lab exercise, demonstration, or exam question. Depending on student abilities, you can require solutions in the form of algorithms, flow charts or code.

Exploring Further is designed primarily for your grade level. This section can be used as a source for semester projects or extended lab sessions. The problems are real exercises in heuristics, and the universe of possible solutions is virtually infinite. The provided solutions are not unique; you are encouraged to post innovative alternatives on our website to share with other educators.

Behavioral Objectives

In addition to the K-12 objectives, students will improve their ability to create hardware solutions that interface with existing devices. They will understand how artificial intelligence can enable autonomous action, and learn to write the programs that accomplish it. Students will be able to explain the operation of currently available electronic equivalents to our human senses, and how this hardware is integrated for teleoperation of a remote, mobile robotic explorer.

Planning and Preparation

Lesson Planning

The entire Mission to Mars could take as long as 4-6 weeks, depending on depth of coverage and available class time. If you choose to employ only a single Mission (3-7 below) this reduces the time to about 1-2 weeks, allowing 1-2 days for creation of the Mars diorama.

Note: Missions 1 and 2 together provide an excellent learning experience for students of all grade levels, and could comprise your entire lesson. But if you decide to include any or all of the other Missions, you must first complete Missions 1 and 2. Alternatively, you could select individual activities from the Missions that do not require a diorama or calibration, write your own custom Mission, and dispense with Missions 1 and 2 entirely.

There is a considerable amount of flexibility in how you can choose to adapt Mission to Mars for your own needs. Included in each Mission is a special section titled Exploring Further, presenting several open-ended problems that will challenge even advanced students. As always, having an RB5X in your classroom provides an infinity of learning experiences.

Review all the material first to gain an appreciation for its scope. Then decide which Missions best meet your needs. You might even invent your own Mission to try out with your class. If you want to share your ideas with other educators, please consider posting them on our website.

Student Work Groups

Again, you have considerable flexibility on how to assign actual student tasks. The entire class could be guided through every step of each Mission. Alternatively, students could be grouped into independent teams and work out their own unique solutions to compare or compete with the other teams. You could also group students into seven "Mission Specialist" teams, having each complete one of the following included Missions:

Mission 1: Creating the Mars Diorama
Mission 2: Robot Calibration
Mission 3: Mapping
Mission 4: Hazard Avoidance
Mission 5: Navigation and Piloting
Mission 6: Energy Management
Mission 7: Sample Acquisition

Materials Needed

Much of what you will need is readily available in most classrooms or homes. The ubiquitous "kitchen junk drawer" can often provide useful items. Standard supplies such as colored markers, tape and paper are taken for granted. For grades 12+ basic lab and shop resources are required, including hand and power tools, electrical meters, fabrication materials such as wood, plastic, and metal, and fastening hardware. Additional items will be specified as needed.

Preparing the Mars Diorama

A good diorama is essential for Mission to Mars. Appendix 1 includes a suggested layout along with some important constraints. Whatever your chosen layout, it will take up a considerable amount of floor space — something that is at a premium in most classrooms. Your best location may be separate space in a gym, basement or unused room. Outdoor dioramas are not recommended unless a covered area with protection from strong winds and rain is available. Even then, outdoor dioramas invite extracurricular modification unless the area is continuously secured.

There are no graffiti on Mars.

If you decide to use only a single Mission, or you are willing to rearrange the diorama between Missions, the space requirements can be scaled down considerably. For example, if you choose to run only Mission 3 (Mapping) you could easily get by with as little as a 2x3 block grid.

Preparing RB5X

In addition to having your RB5X fully charged and in good health, its motion must be calibrated. You need to check the robot for accurate rotation and linear travel, both of which are essential for Mission to Mars. Procedures for calibrating these motions are explained in Mission 2, Mission Problems B, C and D. Depending on your plans for this curriculum, you may want to run these tests now, or you may want to leave them for Mission 2.

Note: you must also run the Initial Robot Calibration outlined on page 30. This is a good time to decide whether you want to use metric or imperial units in this curriculum, as the calibration can accommodate either option. We highly recommend using metrics, unless your approach to this curriculum will be strictly qualitative — and that is certainly a valid option.

Get Ready!

Preparations for Mission to Mars are complete — but hold the countdown for a few final matters. Before you begin, you may want to assign students the task of inventing a "mission name" for RB5X. Most of the good names have already been used by NASA (Voyager, Explorer, Pioneer, Prospector, Surveyor, Sojourner) so this may take a little research. Then again, you may think RB5X is already a pretty cool name and decide to keep it.

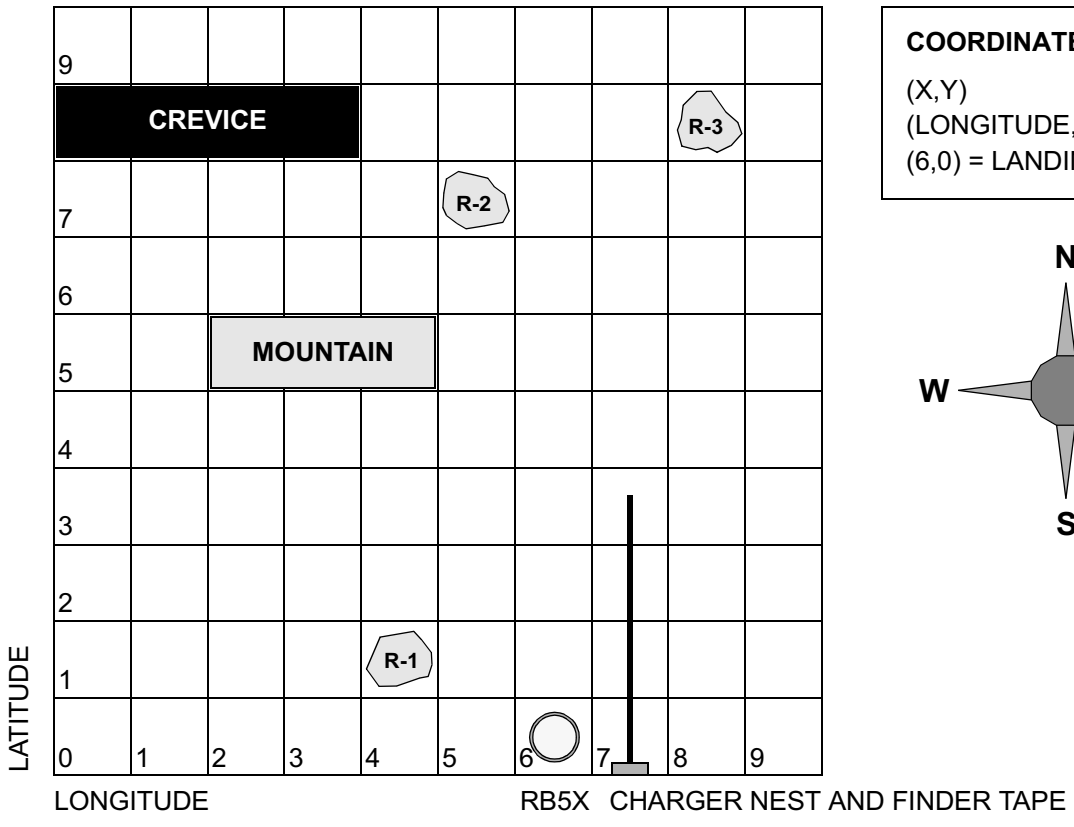
You should definitely assign the task of selecting a name for RB5X's landing site. Sojourner's landing site was named "Sagan Station" in honor of the late Carl Sagan, lifelong popularizer of Mars exploration and *SETI*. Students should research other individuals connected with Mars and choose a name to honor. Some good choices here include: Giovanni Schiaperelli (mapped Mars with a telescope and "discovered" what he thought were canals), H.G. Wells (wrote the science fiction classic "War of the Worlds," about an invasion of Earth by Martians), and Percival Lowell (built his own observatory and studied Mars' polar caps, seasonal changes and dust storms).

Good luck to you, your students and RB5X on your 50 million mile field trip.

Sample Diorama

Your diorama must meet the following criteria. Shown below is one example of a configuration that would work well in all Missions. If available floor space does not allow a 10 x10 square grid, other configurations that meet the criteria are acceptable. If you elect to complete only some of the Missions the diorama can be considerably smaller and simpler.

1. Grid blocks must be 50 cm square to fully contain RB5X's diameter.
2. The mountain may be from 1x2 to 1x4 grid blocks in size.
3. The mountain must near the diorama's center, oriented with its length in the E-W direction.
4. The rocks cannot be adjacent to the mountain, the crevice, or each other.
5. The crevice may be any size from 1x1 to 1x4, but may not be adjacent to a rock or mountain.
6. The crevice may be oriented in either the E-W or N-S direction.
7. Distance to crevice > distance to mountain (as measured from the landing site).
8. The row and column that RB5X starts on must be empty (no rocks, crevice or mountain).
9. Origin (0,0) must also be left empty.
10. RB5X starts on (6,0) facing north adjacent to the charger nest at (7,0).
11. The charger nest must be backed up by a wall (or very heavy object) so that it cannot move.
12. Leave at least 2 grid blocks open to either side of the charger nest.
13. Do not place objects such that any part of the grid is totally cut off from access.
14. The grid need not be square; a rectangle of minimum width 5 blocks wide may also be used.
15. Number the grid along two edges using a 2-axis coordinate system with origin as shown.



COORDINATES:

(X,Y)
(LONGITUDE, LATITUDE)
(6,0) = LANDING SITE

RCL Programming — Setup and Methods

Platform Conventions

The procedure for interfacing your RB5X with a computer is detailed in your Reference Manual and is platform-specific. The procedure given here is for the PC DOS environment, running any version of DOS ≥ 2.0. In other environments, the user will have to interpret these procedures with respect to platform-specific instructions in the Reference Manual. If you have some students advanced in technology skills, you may want to assign them this task.

Connecting RB5X to Your Computer

Before you do anything else, make a backup copy of your RCL disc. Programs can only be saved to this disc. When it's full (1.44 Mbytes) you will need another to continue. Do this now.

The first step is to connect RB5X to your computer and establish a two-way communication link. Follow these steps in the exact sequence indicated:

1. Connect RB5X to your computer's COM1 port using the RS-232 cable. COM1 is RCL's default setting. If COM1 is already used by another device, you can reset RCL to use COM2 by running the routine described on page 30 (Initial Robot Calibration).
2. Ensure the RS-232 switch next to RB5X's battery level indicator is set to "STD" (standard).
3. Carefully remove any EPROM installed in the socket on RB5X's interface panel. It may be left installed if, after powering-up RB5X, you press bumpers 2 and 4 to enter command mode.
4. Turn on RB5X.
5. Turn on your computer and boot to DOS.
6. Insert the "RCL for DOS" disc into the floppy disc drive.
7. Type: A: <RET>
8. Type: RCL <RET>

This will launch RCL and display the title screen along with system information. RCL will then prompt you for a command by printing:

9. *WHAT WOULD YOU LIKE ME TO DO NOW?*

This is the main command mode prompt from RCL. You could, at this point, create a new program or edit an existing program. However, before proceeding you should verify that two-way communication between your computer and RB5X has been established.

10. Type: DUMBO <RET>

If you get the error message *ROBOT AND COMPUTER ARE NOT COMMUNICATING!!!*, first be sure the robot is turned on and in "command mode" (press bumpers 2 and 4). Then check the cable connections, communications port calibration, and your Reference Manual (in that order). If you get the message *COMMUNICATION WITH ROBOT IS WORKING*, all systems are go. Enter F7 (the function key) to return to RCL.

About RCL

RCL (Robot Control Language) is an easy and intuitive programming environment. It is powerful enough for most robot control functions without having to resort to NSC Tiny Basic subroutines. We strongly recommend you acquire and refer to the GRC publication "Robot Control Language with Savvy® — Programmers Manual" for a complete listing of RCL commands and syntax.

If you've done programming in other languages you'll need to relearn some vocabulary. In RCL the words "program" and "command" are replaced by the single word "task," since its language commands are themselves programs (in NSC Tiny Basic). Other differences will be come clear as you work with the language. It is strongly recommended that even experienced programmers go through the steps of the sample program provided later in this section. It will acquaint you with the most important procedures and vocabulary for RCL programming.

An important feature of RCL is the way it accepts and processes individual program lines as they are entered. RCL is a dynamic programming environment, thus when you enter a command you often need enter only the first word — RCL will fill in the rest and provide the input fields for the parameters needed. In some cases, a few more words may be required to specify the command uniquely. For example, to command the robot to move forward 5 feet you would enter:

1. MOVE DISTANCE FORWARD <RET>

and RCL will respond with:

2. *MOVE DISTANCE FORWARD FOR THIS MANY FEET ___*

You then enter a "5" in the distance field, hit <RET>, and the line is complete. In fact, if you try to enter the entire line with distance included, RCL will not accept it.

To program a loop, you need to enter more than the just first word since there are three different types of loops, all of which start with the word BEGIN, i.e.,

BEGIN A LOOP
BEGIN A COUNTED LOOP
BEGIN A LIMITED LOOP

Programming Conventions Used in this Document

- Text typed in Arial caps is text you actually type, e.g., PREPARE THE ROBOT.
- Text typed in Arial italics is a prompt or output from RCL, e.g., *INVENT A TASK CALLED ___*
- The symbol <RET> means you should press the "Return" or "Enter" key.
- Anything in square brackets is an argument (number, variable, or function) you can specify, e.g., MOVE DISTANCE FORWARD THIS MANY FEET [5]. Note: The brackets are not entered.

- Though not shown in the examples provided, all variables entered in an RCL command must be placed in quotes, e.g. SET "B" EQUAL TO 3. The same rule applies to the logical operators (<, >, =, <>). Numbers can be entered without quotes.

Using Variables in RCL

RCL by itself is fine, provided all you need to do is program motions or voice events. However, the use of variables (the key to higher-level programming) requires inclusion of NSC Tiny Basic subroutines. Many of the RCL commands that are designed to handle variables are not functional and generate errors when used. Fortunately, NSC Tiny Basic provides the means for arithmetic and logical operations not otherwise possible.

There are 26 single-letter variables (A-Z) available for use by RCL. Some may already be taken by primary RCL tasks, so to avoid conflicts use only A,B,C,E,F,G,H,J and U. Nine variables are usually enough for all but the most complex problem solving. If you really need more, you can easily check which variables are in use by entering the command:

```
SHOW VARIABLES
```

which will display a screen showing the variables in use, and those still available, in the program most recently listed or edited. Other variables will occasionally be available, depending on what internal tasks RCL is using to run your program.

Two important functional commands for manipulating variables in RCL are:

```
RESERVE THE TINY BASIC VARIABLE [VARIABLE]
CLEAR ALL ITEMS
```

The first command is required at the start of a program to enable the use of a variable. The second sets all variables equal to zero (including any variables currently in use by NSC Tiny Basic).

One of the most powerful functional commands in RCL is:

```
SET THE TINY-BASIC VARIABLE [VARIABLE] EQUAL TO [EXPRESSION]
```

which can be used to set a variable equal to any valid integer (-32767 to +32767) or to a variable or variable expression. It will generate a non-fatal *REPORT MATH ERROR!* message in the latter two cases, but this will not affect the operation of the program (i.e., it's non-fatal). Examples of legal SET statements include:

```
SET THE TINY-BASIC VARIABLE [A] EQUAL TO [3]
SET THE TINY-BASIC VARIABLE [A] EQUAL TO [B]
SET THE TINY-BASIC VARIABLE [A] EQUAL TO [(B+C)/2]
```

The following command sequence is also functional, and can be used to create loops that repeat or terminate on variable values. Again, the use of a variable in more than one field of the same RCL command (i.e., line 3) will generate a non-fatal *REPORT MATH ERROR!* message.

```
1. BEGIN A LIMITED LOOP
2. (do something)
3. REPEAT THE LIMITED LOOP UNLESS [VARIABLE or VALUE] IS [<] TO [VARIABLE or VALUE]
```

Another functional loop sequence is the counted loop:

1. BEGIN A COUNTED LOOP CALLED [VARIABLE] BEGINNING AT [VARIABLE or VALUE] ENDING AT [VARIABLE or VALUE]
2. (do something)
3. REPEAT THE COUNTED LOOP FOR THE COUNTER [VARIABLE]

Note that when you use a counted loop, the underlined VARIABLE must be the same in both lines.

Unfortunately, there are some non-functional RCL commands that would seem to severely restrict what can be done with this language. In case you haven't already discovered them, they are:

```
CALCULATE [VARIABLE] = [VARIABLE or VALUE] [ + , - , * , / ] [VARIABLE or VALUE]
TEST IF [VARIABLE or VALUE] IS [ = , > , < , <> ]  COMPARED TO [VARIABLE or VALUE]
```

Fortunately, we can recover these important operations. By using the COMPILE command we can insert equivalent NSC Tiny Basic statements into the program. The form of this command is:

```
COMPILE THE BASIC STATEMENT [STATEMENT] WHICH MEANS [COMMENT]
```

The argument STATEMENT can be a calculation using variables and integers, such as

```
A=2
A=A+1
A=(A+B)*C/100
```

or any other legal operation. All calculated values will be truncated (rounded down) to integers. It is possible to include more than one STATEMENT in a single COMPILE command by using a colon (:) to separate individual STATEMENTS. For example:

```
COMPILE THE BASIC STATEMENT [A=3:IF A.>B GOTO 42] WHICH MEANS [COMMENT]
```

There is no limit to the number of individual STATEMENTS that can be included within a single COMPILE command, save for that imposed by program size restrictions.

The argument STATEMENT can also be a logical conditional such as

```
IF A<=10 GOTO 50
IF A=B GOTO 50
IF (A>3) AND (C=3) GOTO 42
```

Note that in compound conditionals such as the last example, each conditional must be enclosed in parentheses. Conditional statements can be used to simply jump to another line number in the program, or to terminate a loop by jumping out of it.

When using a GOTO statement in a COMPILE command you cannot GOTO an RCL line number. The line number specified must be the line number in the compiled NSC Tiny Basic version of the program. To see what that line number will be, temporarily enter a dummy line number like 333.

Then go into the RCL command mode and enter:

1. BUILD <RET>
2. *BUILD A PROGRAM FROM THE ROBOT TASK* ___
3. [NAME OF YOUR PROGRAM] <RET>

with the name of your program placed in quotes. Your RCL program will be compiled into NSC Tiny Basic. To list the program showing its NSC Tiny Basic commands and line numbers, enter LTP <RET> and scroll around to discover the actual line numbers. You can then return to RCL and edit the dummy line numbers accordingly.

Finally, the argument COMMENT in the command

COMPILE THE BASIC STATEMENT [STATEMENT] WHICH MEANS [COMMENT]

is simply a comment by you regarding the function of the command, such as COMPARE A&B or CALCULATE C. These comments can be helpful when reviewing and debugging programs.

Some of these methods may seem awkward, but they are the only way to regain full programming functionality. As you acquire experience using variables you will see the real problem-solving potential of this language, for it is the logical processing of variables that allows an autonomous machine to respond intelligently to changes in its internal state or external environment.

Using Random Numbers in RCL

It is sometimes desirable to incorporate random elements into a program. Artificial intelligence problem-solving algorithms often require this feature. The COMPILE statement provides a means to enter a random factor into an RCL program by using the RND function. The command:

COMPILE THE BASIC STATEMENT [A=RND(X,Y)] WHICH MEANS [RANDOM INTEGER]

will assign a integral value from X to Y inclusive to the variable A. The value of integer X must be less than the value of integer Y, and Y can be any integer from 1 to 32767.

The RND function can be used without a “seed” value, in which case it will return the identical sequence of pseudo-random numbers each time the robot is activated. To generate a somewhat less “pseudo” sequence, use the following command before calling the RND function:

RANDOMIZE USING A SEED VALUE OF: [U]

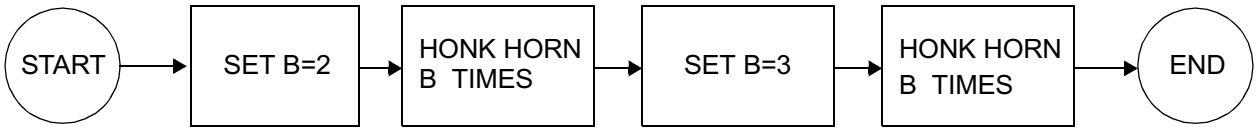
where U is an integer between 0 and 1000 inclusive. By using a variable that takes on different values based on some input (e.g., a counted loop for U that is exited on some external event) the random number generated becomes truly unpredictable.

Using Subroutines in RCL

The COMPILE command also allows the inclusion of subroutines, something that cannot be done in RCL alone. The use of subroutines for procedures that are executed often in the same program can tremendously reduce program size and complexity.

The advantage of a subroutine over an [IF something GOTO somewhere] statement is that the program execution sequence can jump into the subroutine from anywhere in the program and then return to where it jumped from without reference to a specific line number. You can also use a GOTO or JUMP statement to exit a subroutine before reaching the RETURN command.

This concept is best illustrated with a specific example. Consider a program that requires the robot to honk its horn a number of times equal to the variable B. Consider also that this variable could be set to different values at different points in the program. The flowchart below shows a simple program that follows this scheme.



To execute these steps in an RCL program we can use the following code.

HONK: An RCL Program to Demonstrate Subroutine Use

```
1.  PREPARE THE ROBOT
2.  RESERVE THE TINY-BASIC VARIABLE [A]
3.  RESERVE THE TINY-BASIC VARIABLE [B]
4.  SET THE TINY-BASIC VARIABLE [B] EQUAL TO [2]
5.  COMPILE THE BASIC STATEMENT [GOSUB 36] WHICH MEANS [HONK]
6.  SET THE TINY BASIC VARIABLE [B] EQUAL TO [3]
7.  COMPILE THE BASIC STATEMENT [GOSUB 30] WHICH MEANS [HONK]
8.  JUMP TO THE LINE CALLED [DONE]
9.  CALL THIS ROBOT TASK LINE [HONK]
10. BEGIN A COUNTED LOOP CALLED [A] BEGINNING AT [1] ENDING AT [B]
11.    HONK THE HORN FOR [1] SECONDS
12.    WAIT FOR [1] SECONDS
13. REPEAT THE COUNTED LOOP FOR THE COUNTER [A]
14. COMPILE THE BASIC STATEMENT [RETURN] WHICH MEANS [RETURN]
15. CALL THIS ROBOT TASK LINE [DONE]
```

Note how lines 9-13 are executed twice {by the GOSUB), but written only once. In this trivial example only four lines of code are eliminated, but the savings in a more complex program can be substantial. Note also that the subroutine itself is called as line 36 — this is the NSC Tiny Basic line number in the compiled version of the program. By starting the subroutine with a named line (HONK) we make its line number easier to find when scrolling through the compiled program listing. Remember that you must BUILD a program before you can LTP (list) it.

Data Input

The ability to accept input and provide output is crucial for many robotics applications. Input can come from the robot's local environment or from a remote programmer. Sensory information in the form of bumper contacts, sonar echoes, or IR (infrared) readings allow the robot to respond to what it encounters. Output in the form of horn, LED or voice signals can provide feedback to the programmer when acting in a teleoperational mode.

Unfortunately, there is no way to enable I/O (Input/Output) via onscreen prompts or printouts while a program is executing, even with the robot connected to the computer by cable (or the RF communications link option). There are, however, several alternate means for accomplishing I/O.

RB5X can sense its environment in three ways: bumper contacts, sonar and IR. In each of these cases, internal memory registers or NSC Tiny Basic variables take on certain values that can be used for decision-making or calculations within a program.

Bumper input is the easiest data to handle, as it requires nothing more than a few standard RCL commands. The bumpers are numbered clockwise, starting with #1 at the front of the robot, and will display an LED signal (on the correspondingly numbered red LEDs) when pressed. Here is an example of RCL code that uses bumper contact for decision-making in a program:

1. BEGIN A LOOP
2. EXIT IF BUMPER NUMBER [1] IS TOUCHED
3. REPEAT THIS LOOP
4. (do something)

where the bumper number entered can range from 1-8. Alternatively, line 2 could be replaced by

2. EXIT IF ANY BUMPER IS TOUCHED

If you need the robot to react to a specific bumper contact for manual data input while a program is executing, individual bumper values can be incorporated into a COMPILE statement. When any bumper is activated, its ID value is stored in a register accessible through NSC Tiny Basic. Here is an example of RCL code that uses bumper contact for numerical data input:

1. COMPILE THE BASIC STATEMENT [Y=@#7800:IF Y=251 GOTO 3] WHICH MEANS [BUMPER #1]
2. JUMP TO THE LINE CALLED [NEXT BUMPER]
3. SET THE TINY-BASIC VARIABLE [B] EQUAL TO [1]

where the value of the variable Y depends on which bumper is activated (in this case bumper 1), and the variable B is assigned whatever value is needed. Numerical data can be entered into a program by a series of such commands. Individual bumper values are shown in the adjacent chart.

BUMPER	Y=?
1	251
2	254
3	247
4	127
5	239
6	191
7	223
8	253

The bumper value is initially stored at hexadecimal address @#7800, and then transferred to the variable Y (already reserved by NSC Tiny Basic).

Sonar input can be accessed by RCL code modeled after the following example:

```
1. BEGIN A LOOP
2. EXIT IF SONAR DISTANCE IS [MORE] THAN [0.1] FEET
3. REPEAT THIS LOOP
4. CALL THIS ROBOT TASK LINE [MEASURE]
5. COMPILE THE BASIC STATEMENT [LINK#1100:DELAY20] WHICH MEANS [GET D]
6. COMPILE THE BASIC STATEMENT [A=(30*D - 30*73)/22] WHICH MEANS [CALC CM]
```

The loop in lines 1-3 is required to activate the sonar, and only needs to be used once near the start of the program. After that, any time the code in line 5 is used the variable D is assigned a value equal to the number of "counts" between sonar transmission and echo reception. To convert this value into an actual distance we need line 6, the specific details of which depend the units desired ("cm" in this case) and the calibration parameters of the robot (see Calibrating the Sonar).

IR input can be accessed by RCL code modeled after the following example:

```
1. COMPILE THE BASIC STATEMENT [X=#02] WHICH MEANS [TURN ON IR]
2. COMPILE THE BASIC STATEMENT [@#7801=@#7801 OR X] WHICH MEANS [SET BIT]
3. COMPILE THE BASIC STATEMENT [IF (@#7802 AND #40)=0 GOTO 6] WHICH MEANS [LIGHT]
4. SET THE TINY-BASIC VARIABLE [A] EQUAL TO [0]
5. JUMP TO THE LINE CALLED [CONTINUE]
6. SET THE TINY-BASIC VARIABLE [A] EQUAL TO [1]
7. CALL THIS ROBOT TASK LINE [CONTINUE]
```

The conditional "=0" in line 3 will sense light colors against a dark background. If you need to sense dark colors against a light background, change this to "=64." In this example, if the robot senses a light color, the variable A will be assigned a value of "1." If not, the variable A remains at a value of "0."

Direct, real-time, numerical input via keyboard is, as stated earlier, not possible once a program is executing, but by using the hierarchical nature of RCL it can be accomplished, albeit circuitously. The robot must, of course, be connected to the computer by cable or RF link. Consider a program named MOVE that calls, as a task, the sub-program named INPUT consisting of the following lines:

```
1. CALL THIS ROBOT TASK LINE [INPUT]
2. SET THE TINY-BASIC VARIABLE [A] EQUAL TO [42]
3. SET THE TINY-BASIC VARIABLE [B] EQUAL TO [33]
```

where the variables represent, for example, distances to move. When executed, MOVE terminates at a point intended for new data input. On termination, the operator/programmer can easily edit INPUT to include new variable values, and then re-BL (build and load) MOVE. Since INPUT is a task in MOVE, as MOVE is re-compiled these new values will be incorporated as input.

MOVE must be structured to accommodate this type of cyclic data input by including INPUT as a command line preceding any operations on the variables A and B.

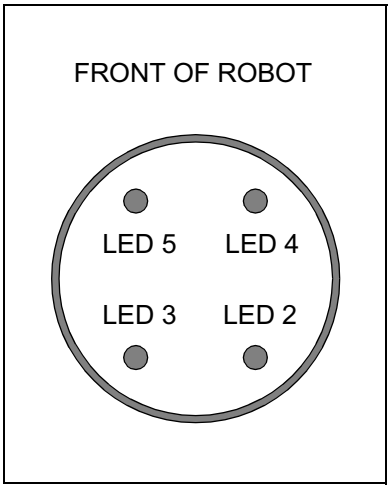
Data Output

As stated in the preceding section, there is no way for RB5X to report data directly to the screen of your computer, even when it is connected via cable or RF com link. Again, there are several alternative means by which the robot can communicated with its programmer. These include the horn, the LEDs, and the voice (if available).

The easiest method here is to use the horn. Conditional loops with included horn commands can be used to count off integral values, or signal any predefined code — even Morse if "text" output is required. Here is an example of a loop that signals the value of a calculated value B:

1. CALL THIS ROBOT TASK LINE [HONK]
2. SET THE TINY-BASIC VARIABLE [B] EQUAL TO [C+E/2]
3. BEGIN A LOOP CALLED [A] BEGINNING AT [1] ENDING AT [B]
4. HONK THE HORN FOR [1] SECONDS
5. WAIT FOR [1] SECONDS
6. REPEAT THE COUNTED LOOP FOR THE COUNTER [A]

Though not as omnidirectional as the horn, the robot's LEDs, if visible in the application, can also be used to signal output codes. There are four individually controllable LEDs numbered as shown in the adjacent diagram. This provides, essentially, four bits of information and allows for much more complex signal codes.



As with the horn, conditional loops are used for converting the values of internal variables into observable output. In practice, line 4 of the preceding example would be replaced by some combination of

```
TURN ON LED NUMBER 2
TURN ON LED NUMBER 3
TURN ON LED NUMBER 4
TURN ON LED NUMBER 5
```

Note that LED 1 is the IR emitting diode located on the underside of the robot. You can turn it on with this command, but it provides no visible output.

The final output option is the voice, available only through the optional voice/sound synthesis package. Using the voice eats up more memory, but provides a method of output most efficiently interpreted by humans. The RCL code for creating words from phonemes, and for speaking those words, is explained clearly in the manual included with that option. Suffice it to say that using a series of SPEAK commands in the form

```
SPEAK THE PHRASE CALLED:  [DIGIT or WORD]
```

with conditional JUMPs provides a powerful means for reporting <u>any</u> type of data.

Creating Your Own RCL Program

You are now ready to begin writing your own RCL programs. This is done by using an onscreen text-based editor. Programs can then be saved to the RCL disk, or downloaded to the robot and executed. If you select "Mode: Free" (as opposed to "Mode: Literal") RCL will assist you by correcting faulty language syntax. This may be helpful to novice programmers. The following example will step you through the programming process. Try this one for practice, and you will soon be sufficiently fluent in RCL to create your own RB5X programs.

You will be creating a simple test program called BEEPLITE. This program will cause RB5X to sound its horn and turn its LED lights on and off. It performs no useful function beyond verifying correct programming procedures. Start out by typing

1. INVENT <RET>

RCL will respond with

2. *INVENT A TASK CALLED* ___

prompting you for the name of the program you want to create. Note that you cannot have two tasks with the same name. If you want to reassign a name, the RENAME function (described later in this example) allows you to do just that. Enter

3. BEEPLITE <RET>

The RCL screen will change, list your program name as "Task BEEPLITE," and prompt:

4. -> ___

You are actually in the edit mode at this point, and can create the program by entering each line in sequence. The flashing cursor is where you begin entering your program. As you enter the lines, RCL will number them sequentially for you. Enter the following line of code:

5. PREPARE THE ROBOT <RET>

This command resets the memory registers to receive a new program. The most current version of RCL (v2.5, 1994) is said to no longer require this line, but programs seem to run more consistently when it is used. The command only costs you 5 bytes, so go ahead and use it.

It is frequently advisable to incorporate a delay between the loading and execution of a program. This allows you time to disconnect the cable (essential if any robot motion is involved), prepare for data acquisition, get out the way, or simply sit back and anticipate, with dramatic pause, your creative masterpiece in action. To include a controlled delay between the loading and execution of this program, enter

6. WAIT FOR ANY BUMPER PRESS <RET>

RCL will respond with

7. *WAIT FOR ANY BUMPER PRESS AND THEN WAIT FOR ___ SECONDS MORE*

allowing you to input the desired delay time. For now, enter

8. 3 <RET>

For programs without any physical motion, the above commands are the only ways to insert such a delay. For programs that do involve motion, RCL provides a calibration parameter that will do this for you automatically without requiring a command within the program. This parameter will be explained in the next section (Initial Robot Calibration).

Now we get to the fun part. Time to program some real actions. Enter

9. HONK <RET>

RCL will respond with

10. *HONK THE HORN FOR ___ SECOND(S)*

prompting you here for the number of seconds to honk the horn. Enter

11. 3 <RET>
12. TURN ON ALL THE LIGHTS <RET>
13. WAIT <RET>

RCL will respond with

14. *WAIT FOR ___ SECOND(S)*

prompting you here for the number of seconds to wait. Enter

15. 3 <RET>
16. TURN OFF ALL THE LIGHTS <RET>

This completes the entry of BEEPLITE. You must now save the program to disc, download it to the robot, and run it. Enter

17. Q <RET>

This quits the edit mode, saves your program to the floppy disc, and returns you to the command mode. RCL will respond with

18. *WHAT WOULD YOU LIKE ME TO DO NOW?*

Note that you will not see this program listed on the disc if you list its directory in DOS — it has been assimilated into the file named RCL.SVY which acts as the RCL task library. You could see it in RCL, however, by using the SHOW TASKS command. Try this now. Then enter

19. BUILD AND LOAD (or more simply) BL <RET>

The RCL screen will change again and prompt you with

20. *PLEASE ENTER THE NAME OF THE ROBOT TASK* ___
21. BEEPLITE <RET>

The RCL screen will change several times as BEEPLITE is compiled into NSC Tiny Basic and download to the robot. When the download is complete the program would normally be executed immediately — in this case you'll need to press a bumper to begin the show. RB5X should wait three seconds, beep its horn for three seconds, and then turn on its LED lights for three seconds. If the cable is still connected, you can easily execute the program again. Just type RUN and the program will repeat.

Editing an RCL Program

If you want to experiment with RCL commands on your own, BEEPLITE can be modified by using the edit mode. Enter

1. LIST <RET>

RCL will respond with

2. *LIST THE* ___
3. BEEPLITE <RET>

RCL will list the program and also display its size (38 bytes). This is a small program. RCL will accommodate programs up to 1000 bytes, so fairly complex problem-solving is indeed possible.

To edit this program enter

4. EDIT <RET>

RCL will prompt with

5. *EDIT THE* ___
6. BEEPLITE <RET>

RCL will list your program and place the cursor after the program's final line. Look it over and review the lines you entered earlier. If you need to add, delete or modify any program lines, this is the place to do it. You are now in the edit mode.

While in the edit mode you can use the following tools:

CURSOR KEYS: to move cursor up and down through the program lines
TAB: to move cursor to the next input field in the current line
SHIFT + TAB: to move cursor to the previous input field in the current line
NUMBER: moves cursor to program line with that number

When the cursor is positioned in the right place, simply enter the desired program line and RCL will renumber all the other lines accordingly. You also have these options:

E <RET>: to edit the line above cursor
D <RET>: to delete the line above cursor
L <RET>: to list the program showing changes
Q <RET>: to quit the edit mode and save the program

There is no way to directly save an edited program under a new name. If you want to modify an existing program and rename it, you must first create a copy of the original with the desired new name before editing the original. Follow these steps from the main command prompt:

1. *WHAT WOULD YOU LIKE ME TO DO NOW?*
2. CLONE A TASK <RET>
3. *MAKE A CLONE FROM THE TASK NAMED .. ___*
4. [NAME OF TASK YOU WANT TO EDIT] <RET>
5. *GIVE THE NEW TASK A NAME .. ___*
6. [NAME YOU WANT FOR NEW TASK] <RET>

RCL will process your request, create a copy the original program, rename it, and save it to disc. You can then edit the cloned program as desired using the EDIT command explained earlier.

To simply rename as existing program, follow these steps from the main command prompt:

1. *WHAT WOULD YOU LIKE ME TO DO NOW?*
2. RENAME <RET>
3. *RENAME OLD NAME ___*
4. [OLD PROGRAM NAME] <RET>
5. *RENAME OLD NAME [OLD PROGRAM NAME] TO NEW NAME ___*
6. [NEW PROGRAM NAME] <RET>

Program names can be up to 256 characters long including spaces. For convenience, keep them short and DOS-like or your SHOW TASKS screen can become very confusing. Remember that you cannot have two programs with the same name on the same RCL disc.

When you're done with RCL and want to return to DOS, enter

BYE <RET>

Program Size and Memory Limits

BEEPLITE was a small program (38 bytes) that barely scratched the surface of RB5X's potential. Individual programs up to 1000 bytes (≈72 lines) in size can be created, saved and run as needed. Even larger programs can be written by exploiting the hierarchical command structure of RCL.

Perhaps you recall reading that RB5X has 8000 bytes (8k) of standard RAM, and were wondering what the other 7k is doing. The short answers is: waiting for more programs. The 1k size limit on programs is built into RCL itself, and has nothing to do with physical memory space. If you need more than 1k, RCL provides a solution.

We explained earlier how the term "task" can mean either "program" or "command." This is the essence of RCL's hierarchical structure. Programs you have already created and saved to disk can be used as commands within other programs.

Using this capability of RCL is simple. All you need to do is include, as an RCL program line, the name of a previously created program. When that line executes, the program it names will run like a subroutine. When complete, it will return control to the following program line. Of course, the "called" program must be present on the RCL disk currently in use. In addition, it must not use any line names (as defined by the CALL THIS ROBOT TASK LINE ... command) already in use by any of the programs preceding it.

In other programming languages this is known as "linking" or "chaining." It provides a way to utilize most (but not all) of RB5X's 8000 byte memory — there's a few hundred bytes taken up by internal tasks relating to the robot's self-diagnostic startup routine.

Here's an example of how to do this. Suppose you have already created a program called MATH that performs several calculations needed for determining whether RB5X will move. Suppose also that during the execution of MATH the variable A is assigned some value.

1. PREPARE THE ROBOT
2. MATH
3. BEGIN A LOOP
4. COMPILE THE BASIC STATEMENT [IF A>10 GOTO 37] WHICH MEANS [CHECK A]
5. REPEAT THIS LOOP
6. MOVE DISTANCE FORWARD FOR THIS MANY FEET [3]

Note that the variable A needs to be reserved in the program MATH, but not in the program that calls MATH as a subtask. Note also that the statement IF A>10 GOTO 37 refers not to an RCL program line, but to the NSC Tiny Basic program line corresponding to RCL line 6. Enter this RCL program, BUILD it, and then list it in NSC Tiny Basic (using LTP) to see the line numbers.

To paraphrase the ever-prescient Mr. William Gates, 8k should be enough memory for anybody. But for extremely complex programs you may still run into the 8000 byte memory limit. If this starts to occur too often in your own robot applications, consider purchasing the 16k RAM option offered by General Robotics. 24k should be enough memory for anybody.

Initial Robot Calibration

About Calibration

There are a few initial calibrations that must be done before you can program RB5X to perform specific tasks. "Calibration" is the process of setting values or states for internal processes that are normally invisible to the user during operation. It controls how RB5X, RCL and NSC Tiny Basic respond to input and generate output.

Calibration settings are saved to the RCL floppy disc and are reapplied every time you BUILD AND LOAD a program. They are, however, specific to whatever serial number RB5X is currently being used. If you are fortunate enough to have more than one RB5X, and require different calibration settings for each, you should have and use separate "RCL for DOS" discs for each robot.

See your Reference Manual for a complete listing of calibration parameters. For most purposes, only a few are important. If you have students advanced in technology skills, you might want to assign them the responsibility of calibration.

To begin calibration, connect RB5X to your computer and go to the RCL command mode. If you don't understand what this means, return to the section titled Connecting RB5X to Your Computer and read it start to finish. Then come back here, empowered by your new-found fluency in RCL.

Calibrating the Com Port

If your computer has a serial (15 pin) mouse it is probably connected to COM1. Even though the mouse is not used while working in RCL, this can cause conflicts. RB5X must be connected to a free serial communications port on your computer. Since the default port for RCL is COM1, you will probably need to change it to COM2 and connect the robot to that port. Enter:

1. CALIBRATE <RET>

RCL will respond with

2. *CHANGE THE CALIBRATION PARAMETER NAMED __*
3. COMMPORT <RET>

RCL will respond by telling you which port it is currently using (1,2,3 or 4) and prompt:

4. *SET THE VALUE TO __*
5. 2 <RET> (or whatever port you need to use)

RCL will verify your setting and return to the command mode.

If your computer has no available serial communication ports you will need to make one available and set it for 1200 baud. Consult your operating system's documentation for instructions.

Calibrating the Bumper-Start Delay

This calibration setting will save you some programming steps later. It requires a program loaded into the robot to be activated by a bumper press, giving you time to disconnect the cable before motion begins. It also creates a three second delay (default value) between the bumper press and program execution. With this parameter set to Y (yes), an initial RCL program line of WAIT FOR ANY BUMPER PRESS is no longer required in programs that involve robot motion.

(In programs that do not involve robot motion this parameter has no effect. To create a delay in these cases, use the WAIT FOR ANY BUMPER PRESS command at the start of the program.)

To enable this "bumper start" feature for programs involving robot motion, enter

1. CALIBRATE <RET>

RCL will respond with

2. *CHANGE THE CALIBRATION PARAMETER NAMED __*
3. BUMPER.START <RET>

RCL will respond with

4. *BUMPER.START IS N* or *BUMPER.START IS Y*
5. Y <RET>

If you want to change the three second delay between bumper press and program execution, enter

6. CALIBRATE <RET>
7. *CHANGE THE CALIBRATION PARAMETER NAMED __*
8. DELAY.START <RET>
9. *DELAY.START IS 3*
   SET THE VALUE TO __

at which point you can specify the exact delay time.

There is another calibration parameter called BUMPER NEEDED, but I don't recommend using it as its value Y or N is not saved. All it seems to do is generate an onscreen reminder about disconnecting the cable. It does not create the bumper press requirement or program execution delay that BUMPER.START so conveniently provides.

Setting Other Calibration Parameters

Most of the other parameters have acceptable default settings. But if you need RB5X to execute very accurate motions, some of these parameters may need to be tweaked slightly. One motion that sometimes needs adjustment is rotation. This can be done by changing the parameters:

```
CALIBRATE SPIN.DEG/SEC = [62]
CALIBRATE SPIN.SECS/90DEG = [1.38]
CALIBRATE SPIN.SECS/TURN = [4.95]
```

For the first parameter, decrease the setting if the robot doesn't turn far enough, and increase it if the robot turns too far. Reverse this logic for the second two parameters. Note that each of these parameters affects specific rotation commands. The first affects commands that involve specified rotation angles, such as SPIN CLOCKWISE FOR [45] DEGREES. The second affects commands that invoke right-angle turns, such as SPIN RIGHT 90 DEGREES. The third affects only the SPIN AROUND CLOCKWISE (or COUNTERCLOCKWISE) [VALUE] TIMES command.

For some robot applications you may want to use metric values in commands involving distances, e.g., the command

```
MOVE DISTANCE FORWARD FOR THIS MANY FEET [VALUE]
```

This can be done by setting the following parameter:

```
CALIBRATE MOVE.SECS/FT = [9.84]
```

Note: If you make this change, movement commands will still prompt you for distances in FEET (as shown above). This text is built into RCL. Ignore it, and treat all distance prompts as if they actually read METERS.

If your intended use of RB5X does not include calculations, or if you just feel more comfortable working with imperial units, leave this parameter at its default setting of 3. If your intended use requires very accurate movements, further fine-tuning of this parameter may be necessary.

For sonar applications, the loop exit command

```
EXIT IF SONAR DISTANCE IS [LESS or MORE] THAN [VALUE] FEET
```

also expects the VALUE to be in units of FEET. To use a VALUE in units of METERS instead, change the following parameter from its default setting of 22 to:

```
CALIBRATE SONAR.COUNTS/FT=[72.6]
```

Again, after making this change, the command will still prompt for distances in FEET. If your intended use requires very accurate movements, further fine-tuning may be required. See the following section for detailed instructions on sonar calibration.

Calibrating the Sonar

If your robot applications require very accurate sonar range measurements, fine tuning of the sonar offset may be necessary.

```
CALIBRATE SONAROFFSET = [73]
```

Note that this command does not require a period after the word SONAR.

The default setting of 73 is the "usual" minimum value of the variable D returned by a series of sonar range measurements. However, we have found this value can vary from one robot to the next. To check your robot, you need a program that reports D as a series of horn beeps or other easily counted signals. Enter the following program:

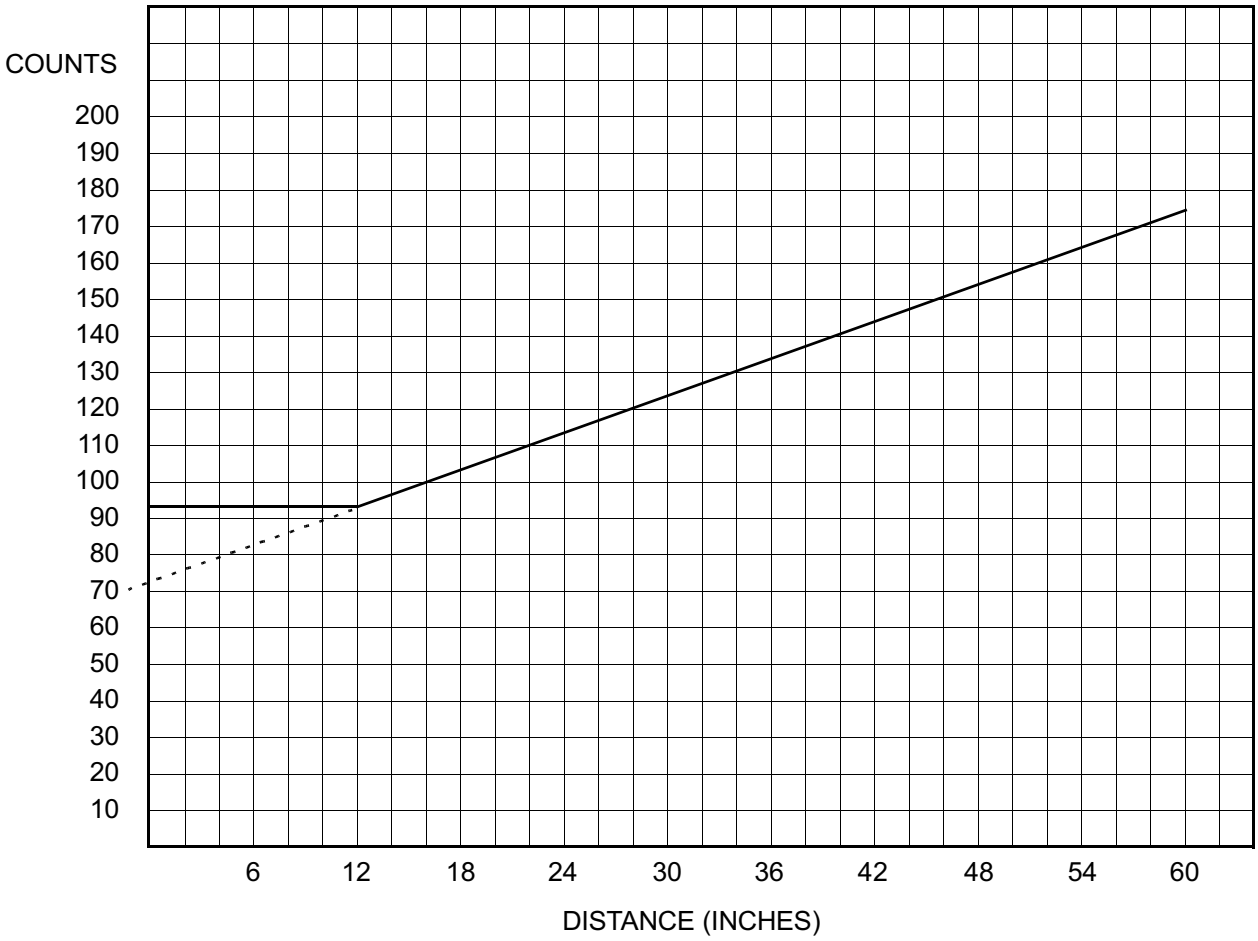
D-TEST: An RCL Program to Measure Sonar Offset

```
1.  PREPARE THE ROBOT
2.  RESERVE THE TINY-BASIC VARIABLE [A]
3.  BEGIN A LOOP
4.  EXIT IF SONAR DISTANCE IS [MORE] THAN [0.1] FEET
5.  REPEAT THIS LOOP
6.  CALL THIS ROBOT TASK LINE [START]
7.  COMPILE THE BASIC STATEMENT [LINK#1100:DELAY20] WHICH MEANS [GET D]
8.  SET THE TINY-BASIC VARIABLE [A] EQUAL TO [0]
9.  BEGIN A LOOP
10. HONK THE HORN FOR [0.1] SECONDS
11. WAIT FOR [0.2] SECONDS
12. COMPILE THE BASIC STATEMENT [A=A+1] WHICH MEANS [INCREMENT A]
13. COMPILE THE BASIC STATEMENT [IF A=D GOTO 47] WHICH MEANS [EXIT LOOP]
14. REPEAT THIS LOOP
15. WAIT FOR ANY BUMPER PRESS AND THEN WAIT [1] SECOND MORE
16. JUMP TO THE LINE CALLED [START]
```

Note: The initial loop in lines 3-5 are needed to simply activate the sonar; it isn't the most elegant method, but none of the LINK codes work by themselves. For convenience, you can write a program called SONARON consisting of these three lines. In future programs, the sonar can then be activated by the simple one-word command SONARON.

Set the robot in an open area where the test can be run. Place a flat, smooth vertical barrier to reflect the sonar 60 inches from the transmitter. Load D-TEST and get ready to count beeps. Record data from 60 inches to 12 inches in 6 inch increments. Run each distance a few times to make sure you get a representative value — spurious readings sometimes occur.

Next, graph your data with "counts" on the vertical axis and "distance" on the horizontal axis. Although details will vary from one robot to the next, you should end up with a plot that looks something like the following.



Finally, calculate the slope and Y-intercept of your plot.

Note that the data bottoms out near 90 counts, due to limitations in the sonar response time. The Y-intercept is not this bottomed-out value, but the one extrapolated from the plot itself (either mathematically or graphically). In this case the correct Y-intercept is about 72.

Likewise, when you calculate the slope, ignore this bottomed-out data. Express the slope in units of "counts/foot" (not counts/inch) to three significant figures.

You are now ready to recalibrate the sonar. Set the following parameters:

```
CALIBRATE SONAR.COUNTS/FT=[SLOPE]
CALIBRATE SONAROFFSET = [Y-INTERCEPT]
```

RB5X's sonar can now be used to measure distances with errors of 1-5%. Residual errors are due to variations in the air through which the sonar must propagate.

URL's You Can Use

Research on the internet can be time-consuming. If you are looking for information related to the activities in Mission to Mars, these websites will be a good starting point. There is an incredible amount of robotics research currently underway. Much of this work is in areas on the cutting edge of technology. Examples include:

- artificial intelligence
- visual pattern recognition
- self propulsion
- synthetic muscle
- sample acquisition

The following URL's were valid at the time of this printing (5/00). Get to any one of them and you'll probably find links to all the others.

Sites Related to Robotics:

Carnegie Mellon University Robotics Institute: http://www.ri.cmu.edu
JPL Robotics Lab: http://robotics.jpl.nasa.gov
Harvard Robotics Lab: http://yellowstone.harvard.edu
Stanford Robotics Lab: http://robotics.stanford.edu
Caltech Robotics Lab: http://robby.caltech.edu
MIT Artificial Intelligence Lab: http://www.ai.mit.edu
Sandia Labs Intelligent Micromachine Initiative: http://www.mdl.sandia.gov/Micromachine

Sites Related to Planetary Exploration:

NASA Home Page: http://www.nasa.gov
NASA Photo Archive: http://photojournal.jpl.nasa.gov
NASA Historical Archive: http://www.ksc.nasa.gov/history/history.html
National Space Sciences Data Center: http://nssdc.gsfc.nasa.gov/planetary/planetary_home.html
JPL Mars Exploration Center: http://mpfwww.jpl.nasa.gov

Sites for General Robotics Corporation:

Products and corporate information: http://www.generalrobotics.com
RB5X educational forum: http://www.edurobot.com

Glossary

aerobraking	the use of air drag to decelerate a spacecraft in preparation for orbital insertion or landing, and an alternative to fuel-guzzling retro-rockets
amp-hour	quantity of charge delivered or received when a current of 1 ampere flows for one hour; exactly equal to 3600 coulombs; usually used for rating the storage capacity of batteries
articulated	able to flex, bend or rotate; opposite of rigid; used to describe mechanical components that mimic the action of human limbs
artificial intelligence	computer processing that mimics the action of the human mind, including such functions as learning, intuition, modeling, visual processing
ascent stage	separable component of a spacecraft used to regain orbit after landing
atm	abbreviation for "atmosphere," a quantity of pressure; 1 atm equals the average sea-level pressure of Earth's atmosphere; $\approx 1.0 \times 10^5$ N/m^2
atmosphere	the gaseous part of a planet (see also *lithosphere* and *hydrosphere*)
autonomy	ability to function without outside assistance, including, but not requiring, artificial intelligence, hazard avoidance, and self-maintenance
booster	temporary thrust-producing module used for brief speed changes, often ejected after use for disposal or refilling
calibrate	the process by which numerical performance specifications are set or obtained for mechanical and electrical devices
carbon-dioxide	CO_2, a component of most atmospheres; can be liquid or solid under suitable conditions of pressure and temperature
cardinal direction	one of the 4 primary compass directions (North, East, South, West)
cartographer	a person who designs or studies maps, globes or geological models
center-of-gravity	point in space representing the average position of matter in a solid body; equivalent to "center-of-mass" in most situations
conjunction	point in the motion of two astronomical bodies where they have equal celestial longitudes, but not necessarily equal celestial latitudes
convention	universally adopted agreement about how to measure or define an otherwise arbitrary quantity, e.g., the decision to label the equator latitude 0° is an international cartographic convention
coordinate system	geometric reference structure in any number of dimensions that allows sets of numbers to represent the locations of points
cosmic ray	very high energy atomic nuclei or electrons traveling nearly at the speed of light; believed to originate from catastrophic cosmic events
cybernetics	area of science concerned with similarities and differences between the human brain and electronic processors designed to model it
effluvial	related to the flow of water and its effects on topography
exogeology	area of science concerned with the geology of other planets, especially in the sense of comparison with Earth's own geology
exobiology	area of science concerned with the biology of other planets, especially in the sense of comparison with Earth's own biology
exolexicologist	not an official word, but an extrapolation of a legitimate word meaning "someone who studies words related to things beyond the Earth"
feedback	in general, the use of an output to modify or control an input

flywheel	device that stores energy in the form of rotational kinetic energy, most often through the use of high-rpm, high-density discs
friction	resistive force produced when two bodies in contact move relative to one another; can be rolling, sliding or viscous in nature
fuel cell	type of battery that stores energy in the form of dissociated chemical compounds, releasing electrical energy by chemical reaction; easily recharged and more efficient than standard electrolytic batteries
g (or gee)	1g is the amount of acceleration equal to 9.8 m/s^2, or amount of inertial reaction (also called g-force) equal to 9.8 N/kg; normal Earth gravity
geology	the area of science that investigates the relationships and evolution of lithospheric features on Earth (*geo* ≡ earth and *ology* ≡ study)
gravimetric	related to the measurement of gravitational forces or anomalies
heuristics	area of study concerned with the process by which the human mind learns how to solve problems; also applied to machine AI
hydrosphere	the liquid part of a planet (see also *lithosphere* and *atmosphere*)
igneous	rocks or minerals formed by high-temperature geologic processes
impact basin	flat, depressed, usually circular area created by the impact of a meteor
inferior planet	planet in an orbit closer to the Sun than the observer's planet
internal friction	friction generated within the drive train and motor of a machine
I/O channel	data link between a computer and some outside device
iron-oxide	most often Fe_2O_3, commonly known as "rust"
isotope	atomic nucleus having more or less than the average atomic weight due the an excess or deficiency of neutrons; typically less stable and hence more radioactive than the average nucleus
joule (J)	amount of energy required to exert a force of 1 N over a distance of 1 m [1 N·m, mechanical], or supply 1 W of power for 1 s [1 W·s, electrical]
kilowatt (kW)	amount of power equal to 1000 watts
kinematics	area of study concerned with the mathematical description of motion
kips	abbreviation for "thousands (kilo) of instructions per second;" usually used to rate the command processing speed of a computer
lithosphere	the solid part of a planet (see also *atmosphere* and *hydrosphere*)
laser	acronym for "light amplification by stimulated emission of radiation"; optical device that produces a beam of monochromatic coherent light
latitude	measure of distance, in degrees north or south of the equator
light-minute	distance traveled by light (or by a radio wave) in a time of 1 minute; equal to 1.8×10^7 km or about 11 million miles
longitude	measure of distance, in degrees east or west of the prime meridian
magnetometer	device used to measure the strength of a magnetic field; is most often calibrated to provide readings in units of "gauss" or "tesla"
metamorphic	rocks or minerals formed by high-pressure geologic processes
N_2 propulsion	system that produces thrust by venting high-pressure nitrogen gas; used for low thrust applications; safe, non-reactive exhaust; highly reliable
newton (N)	amount of force required to accelerate 1 kg of mass at a rate of 1 m/s^2
odometer	mechanical or electrical device used to measure total distance traveled
opposition	point in the motion of an astronomical body where its celestial longitude is 180 degrees away from the Sun

ordered-pair	set of two numbers representing the location of a point in a plane, and measured with respect to a 2-dimensional coordinate system
pattern recognition	sub-area of AI concerned with the process by which visual information is acquired, processed and interpreted
permafrost	water frozen permanently into the soil at a depth not affected by seasonal temperature variations
photovoltaic	physical process by which the radiant energy of a photon is converted into electrical energy; principle of operation underlying solar cells
precession	the circular drift or wobble of the axis of a rotating body caused by the application of a force off-axis to the body's rotation
primordial	relating to the distant past, usually in the era during which astronomical bodies were formed, e.g., primordial Earth 4.5 billion years ago
Pythagorean Theorem	mathematical relation between the sides of a right triangle relating the sides A and B to the hypothenuse C with: $C^2 = A^2 + B^2$
radar	acronym for "radio navigation and ranging"; process by which radio waves reflected from a remote body can be used to determine its distance and location relative to the radio transmitter
ranging	determination of the distance to a remote object by optical, acoustic or electromagnetic reflection
receiver	device designed to receive and process a signal from a transmitter
regenerative braking	method of deceleration accomplished via transfer of kinetic energy to a mechanical or electrical storage device, as opposed to frictional braking where pads or discs are pressed together converting energy into heat
renewable	as applied to an energy source, one that is replenished by natural processes with no action on our part, e.g., solar, wind or hydro energy
retrograde	rotation or revolution in the opposite direction defined by convention
retro-reflector	tetrahedronical mirror-like device capable of reflecting a signal exactly back to the source regardless of the angle of incidence
retro-rocket	thruster designed to fire in the direction opposite to normal motion and used for deceleration of spacecraft
rolling friction	friction created by the action of a wheel rolling, but not sliding, on a surface; function of both surface smoothness and wheel diameter
RF	abbreviation for "radio frequency"; that part of the electromagnetic spectrum below about 10^{13} Hz (or with wavelengths > 1 cm)
RTG	abbreviation for "radioisotope thermoelectric generator"; device that produces power by converting the heat from a radioactive substance, usually thorium, directly into electrical energy with no moving parts
sedimentary	rocks or mineral formed by erosive and layering geologic processes
seismometer	device designed to measure ground movement relative to an inertial system, usually a large mass mechanically attached to a recorder
SETI	acronym for "Search for Extra-Terrestrial Intelligence"; the ongoing process, begun in the 1950s, to find hard scientific evidence for the existence of other technological civilizations in our galaxy
sidereal	measured with respect to the motion of the stars
simple machine	mechanical device used to multiply force or speed; classic examples include the lever, inclined plane, wheel and axel, pulley and screw

sol	one apparent day measured with respect to the Sun; on Earth, 1 sol is 86,400 seconds; on Mars, 1 sol is 88,643 seconds
solar cell	thin, flat wafer of material, usually silicon doped with germanium, or gallium doped with arsenic, in which the photovoltaic effect converts the energy of an absorbed photon into electrical energy
solar panel	flat, usually rectangular, array of solar cells; used to convert sunlight into electrical energy at 12, 24 or 48 VDC
sonar	acronym for "sound navigation and ranging"; process by which sound waves reflected from a remote body can be used to determine its distance and location relative to the sound transmitter
STS-X	abbreviation for "Space Transportation System - mission #X", and the official NASA designation for the Space Shuttle
superior planet	planet in an orbit farther from the Sun than the observer's planet
synodic	measured with respect to the motion of Earth
talus	rocky debris eroded from a mountain by wind, rain or temperature
teleoperation	the process of controlling a device in a remote location, usually by means of 2-way radio communication
telepresence	the process of observing a remote location with multiple sensory input, including visual, aural and sometimes tactile information, usually by means of 2-way radio communication
terrestrial	relating to the Earth (Terra)
transmitter	device designed to process and send a signal to a remote receiver
triangulation	geometric method for locating observer's position in a plane; requires directional and coordinate information for two remote points
tropical year	amount of time for Earth to complete one orbit with respect to the Vernal Equinox; basis for the definition of the astronomical "second"
UHF	abbreviation for "ultra-high frequency"; that part of the RF spectrum with frequencies between 3×10^8 and 3×10^9 Hz; useful for video
URL	Universal Resource Locator, an internet address, e.g., http://www.etc.
Vernal Equinox	point on the celestial sphere of 0° longitude and 0° latitude, currently located in the constellation Pisces; changes slowly due to precession
virtual reality	artificial environment produced by a system of computers and sensory feedback transducers (for sight, hearing and touch) that interacts with the user in a real-time mode
volcanic shield	topographic feature consisting of a broad, flat, gently uplifted layer of igneous material, usually basalt, through which volcanoes vent
volt	amount of electromotive force that imparts an energy of 1 J to 1 C of electric charge
watt (W)	amount of power equal to an energy consumption rate of 1 J/s or 1 V·A
watt-hour (W·hr)	amount of energy equivalent to 1 W of power acting for 1 hr of time; exactly equal to 3600 J
x-ray	that part of the electromagnetic spectrum between frequencies of about 10^{17} and 10^{20} Hz; harmful to most life forms
x-ray spectrometer	device used to detect and measure x-rays; provides data in the form of x-ray intensity vs. frequency; used to identify species of nuclei present in a sample of matter

Mission Solutions

An Important Message Regarding Solutions

The following pages are copyright free for use within this licensed course. You may duplicate them for distribution to students as needed, cut and paste them into related materials, or convert them into overhead transparencies for group presentations.

Be aware that numerical parameters in some solutions are dependent on the overall mechanical condition of your RB5X and the state of charge of its battery. Your mileage may vary. Numbers based on measured data have been expressed to only 2-3 significant figures, reflecting anticipated variations between robots. Getting "correct" solutions is more a matter of discovering successful procedures, and general problem solving, than of matching a specific numerical value.

Solutions to some of the essay questions are left open. For others, suggested content has been provided. In a few cases, complete (though not unique) answers are given as representative of the author's approach or interpretation. Encourage and be open to innovative approaches — getting your students to think outside of normal constraints is a primary goal of this curriculum. If you discover a really creative alternate solution, please share it with other educators on our website.

The RCL code solutions for the Mission Problems are likewise non-unique. Those provided have been tested on an actual diorama and demonstrated in practice as operable solutions. If you can solve the Problem in fewer lines of code, again, please share it on our website. To help you read our program solutions, a list of all the notation conventions used is provided in the section titled RCL Programming — Setup and Methods. You may want to review these now.

RCL code solutions are provided without embedded comments, however, code functions that may not be immediately obvious are explained. In many cases, code sequences used are recycled from previous Mission solutions, so if you are doing the Missions out of sequence, recognize that some code sequences may have been explained earlier. We assume user fluency in RCL programming as a prerequisite for this curriculum.

Exploring Further contains the most open-ended questions, specifically those relating to hardware modification or performance calibration. You can expect the greatest divergence between student solutions and provided solutions in these sections.

Disclaimer: Some of the provided solutions involve modifications to the robot and/or its official accessories. These modifications may void your warranty from GRC. If your warranty is still in affect, check with the manufacturer before attempting any modifications. Regardless of warranty status, we strongly recommend all modifications be done either by the Instructor, or by students working under the Instructor's direct personal supervision.

Mission 1: Creating the Mars Diorama

A. open topic
B. open topic
C. [crossword solutions on following two pages for easy duplication]

D. The following names are good starting points for research:
- Carl Sagan (planetary scientist, proponent of manned Mars mission)
- Percival Lowell (erroneously "observed" a canal system on Mars)
- H.G. Wells (wrote War of the Worlds, about a martian invasion)
- Edgar Rice Burroughs (wrote Tales of Barsoom, a martian fantasy)
- John McCall (space artist of Mars and other planet-scapes)
- Francisco Fontana (first accurate telescopic drawings of Mars)
- Camille Flamarion (improved Mars maps, believed Mars hospitable)
- Giovanni Schiaperelli (mapped and named features on Mars)
- Johannes Kepler (discovered Mars' elliptical orbit)
- Asaph Hall (discovered Mars' two moons, Phobos and Deimos)

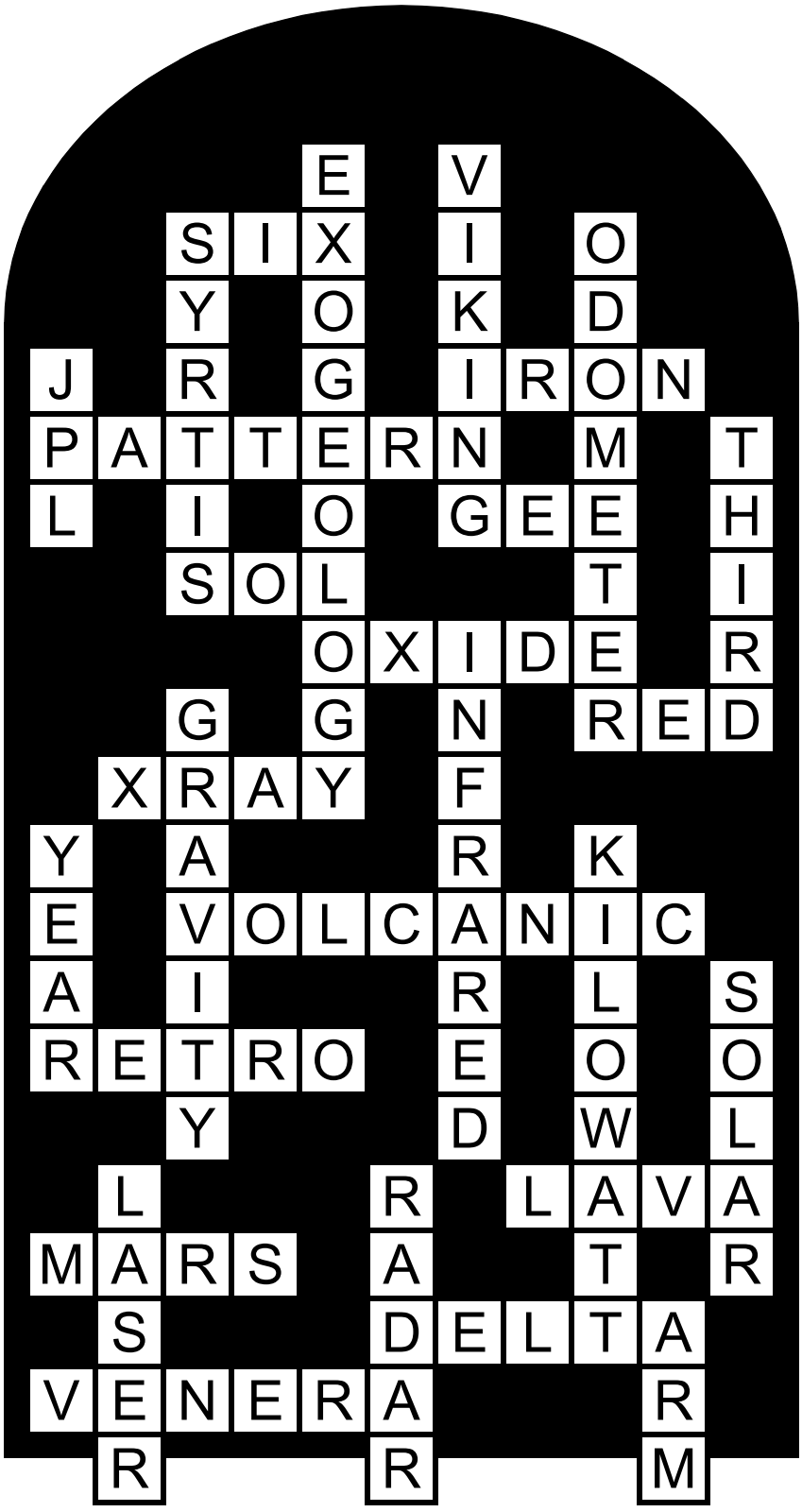
E. Some of the environmental conditions requiring special design are:
- launch and landing g-forces and impact forces
- exposure to vacuum of space in transit
- high and low temperature extremes
- radiation (particulate and ultraviolet)
- abrasive dust on ground and in wind
- formation of condensation or frost
- traction difficulty in loose sand
- traction difficulty due to low gravity
- uneven, boulder-strewn terrain
- extreme distance from earth-based teleoperator

F. open topic

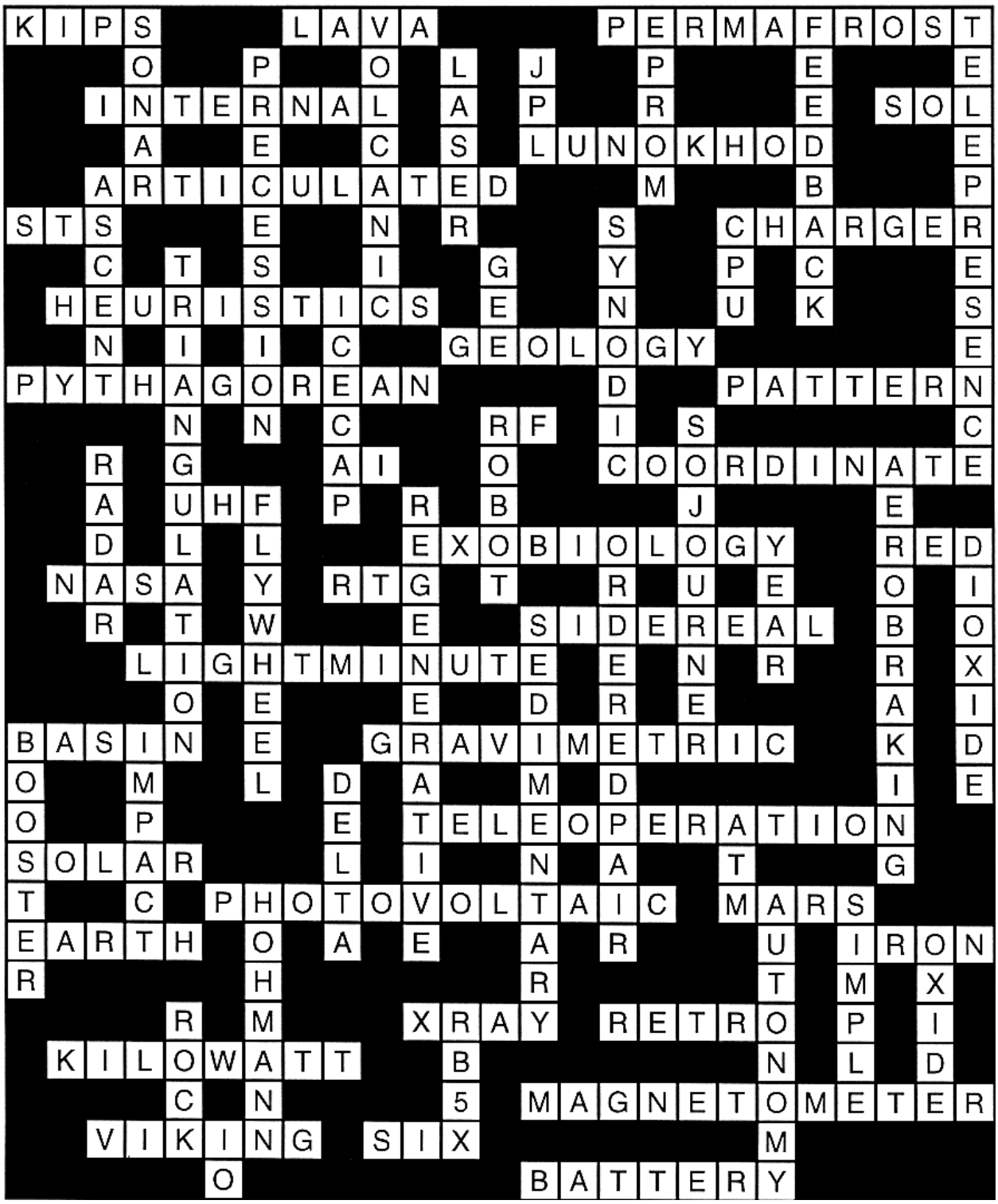
G. The matches between rocks and names are as follows:

Flat Top	(J,5.5)
Flute Top	(J,6.5)
Grommit	(V,8)
Halfdome	(I,3)
Hippo	(X,1.5)
Moe	(K,3)
Pumpkin	(L,3)
Shark	(FG,3)
Stimpy	(KL,4)
Wedge	(CD,5.5)

Mission to Mars Crossword Puzzle Solution



Exolexicologist Certification Puzzle Solution



Mission 2: Robot Calibration

Mission Questions

A. Examples of autonomous machines familiar to most people:

- stoplight — senses when cars are nearby and changes signals accordingly
- streetlight — senses when its day or night and turns itself on and off
- motion sensor — turns room lights on or off in response to activity
- weather station — measures and transmits changing environmental data
- automatic door — detects your presence, opening and closing within safe margins

B. If RB5X senses a critical situation and calls for help, its call will be received 4.3 minutes later. The travel time for the radio signal can be calculated as follows:

$$t = \frac{d}{v} = \frac{78{,}000{,}000 \text{ km}}{300{,}000 \text{ km/s}} = 260 \text{ s} = 4.3 \text{ min}$$ (See also Mission 4, Mission Question A.)

When Mission control hears the call and responds, their instructions will take another 4.3 minutes to travel back to RB5X on Mars. So the total time is 2 x 4.3 minutes = 8.6 minutes. Of course, this assumes they respond immediately. Additional delays could occur depending on how long it takes them to process the information and solve the problem.

C. When Earth and Mars are on directly opposite sides of the Sun the total distance between them will be 150 Mkm + 228 Mkm = 278 Mkm. The one-way travel time for a radio signal is now:

$$t = \frac{d}{v} = \frac{278{,}000{,}000 \text{ km}}{300{,}000 \text{ km/s}} = 927 \text{ s} = 15.4 \text{ min}$$ (See also Mission 4, Mission Question A.)

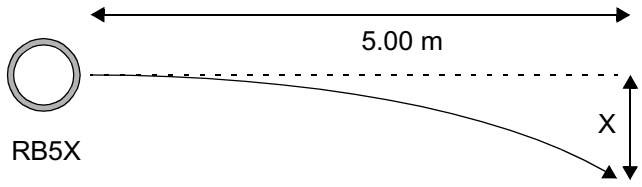
D. RB5X needs to know its own speed in order to relate distance to time. Speed, distance and time are connected by the formula "v = d/t." If RB5X measures the distance to some object, it can calculate the time it would take to get there using "t = d/v." If RB5X knows it has a certain amount of time in which to act, it can calculate how far it could travel during that amount of time using "d = v·t." For example, suppose RB5X needs to return to its base before sunset. If the base is 720 m away, and its speed is 0.1 m/s, it will need to leave for home 2 hours before sunset.

Mission Problem

The following measurements need not be done on the diorama. Any 5 m x 5 m open space with a smooth level surface will do. You also need a measuring tape and stopwatch for data acquisition.

A. Measuring speed is easy. You don't even need to write a program. With the Utility EPROM in place, RB5X can be put into "wander mode" by pressing bumpers 2 and 5. If no obstacles are detected in its path, it will keep moving forward. The actual distance is not important, but should be at least 5 m for accuracy. Time RB5X's motion over this known distance; then use the formula "v = d/t" to get the speed. For a fully charged RB5X on level ground the typical value is around 0.12 m/s. Values lower than 0.10 m/s indicate possible battery or motor problems, and a need to consult your Reference Manual before continuing.

B. This measurement is tricky, since it's difficult to know exactly which way RB5X is pointed before it starts moving. One method for recording the change in direction of motion is to simply scatter some chalk dust in its path and look at its tracks over a 5 meter distance. You can recycle the program GO5M from Mission Problem B for this test. Longer paths yield greater accuracy, but 5 meters should be sufficient. If you extend a straight line tangent to the first meter of motion the veer can be measured as shown in the Fig. 2-1. Directional error can now be calculated as the ratio of veer (X) to distance traveled (5.00 m).



MEASURING DIRECTIONAL ERROR

NOTE: X must be measured in units of meters to yield a proper error ratio

$$\text{error} = \frac{X}{5.00\ \text{m}}$$

Figure 2-1

If your RB5X has a directional error > 0.2, consult your Reference Manual for instructions on adjusting wheel speeds. Errors < 0.2 can be corrected by increasing the diameter of the "slow" wheel with layers of black electrical tape. The following formula was derived empirically, and relates the diameter increase of the wheel to the resulting change in veer.

$$\Delta\text{error} = N \times 0.15$$

Where N is the number of layers of standard ¾ inch x 7 mil (0.007 inch) electrical tape, fractional layers allowed. With a little trial and error you should be able to get RB5X rolling straight as an arrow. Try to get the directional error down to around 0.01.

To apply the tape, set the robot face down on a cushion. Due to the low gearing, you will not be able to rotate the wheel by hand as you wrap the tape. You can, however, use the electric motor to rotate the wheel. Disconnect the motor power lead from where it attaches to the motor controller board. This lead terminates in a female jack that will accept AWG #14 wire stubs. Connect these stubs to a 6 V lantern battery and the wheel will rotate allowing easy application of the tape.

C. Measuring distance accuracy is also easy, but do not attempt it before you have established accurate straight-line motion (see preceding Mission Problem). RB5X must be programmed to move some given distance forward in a straight line.

GO5M: An RCL Program to Test Motion Accuracy

```
1. PREPARE THE ROBOT
2. MOVE DISTANCE FORWARD FOR THIS MANY FEET [5]
```

Note: Remember that line 2 is actually telling RB5X to move 5 meters, not 5 feet. If you don't understand why this is expressed in "meters," refer to the Initial Robot Calibration on page 30.

Measure the actual distance traveled. The error can now be calculated from the formula:

$$\text{error} = \frac{\text{actual distance} - 5.00\text{ m}}{5.00\text{ m}}$$

Positive error means "went too far" and negative error means "didn't go far enough." You need to correct for errors in excess of ±0.01. In this case, correction involves no mechanical operations — it can be done by simply recalibrating the MOVE.SEC/FT parameter in RCL.

The default value for this parameter is 3.00. You may have changed it to 9.84 (metric) during the Initial Robot Calibration. Either way, to adjust for the error measured here use this formula:

MOVE.SEC/FT = (3.0 or 9.84) x (1 ± error)

with the ± applied in the opposite sense of the error. Use this new value to recalibrate RCL now. Procedures for setting this parameter are explained in the Initial Robot Calibration on page 30.

D. Rotational error could be measured directly by attaching a thin rod or pointer to the top of the robots's dome, commanding the robot to rotate, and comparing its rotation to an independently mounted protractor. This method is often suggested by students, but is difficult in practice since the center of rotation is uncertain and the protractor tends to move out of alignment.

An easier, albeit less direct, method is trigonometric. By using a "side-angle-side" analysis, the rotation angle can be calculated from two measured distances. To perform this analysis, RB5X must be programmed to move 5 m, rotate 90° twice in the same direction, and then move another 5 m, producing the pattern of motion shown in Fig. 2-2.

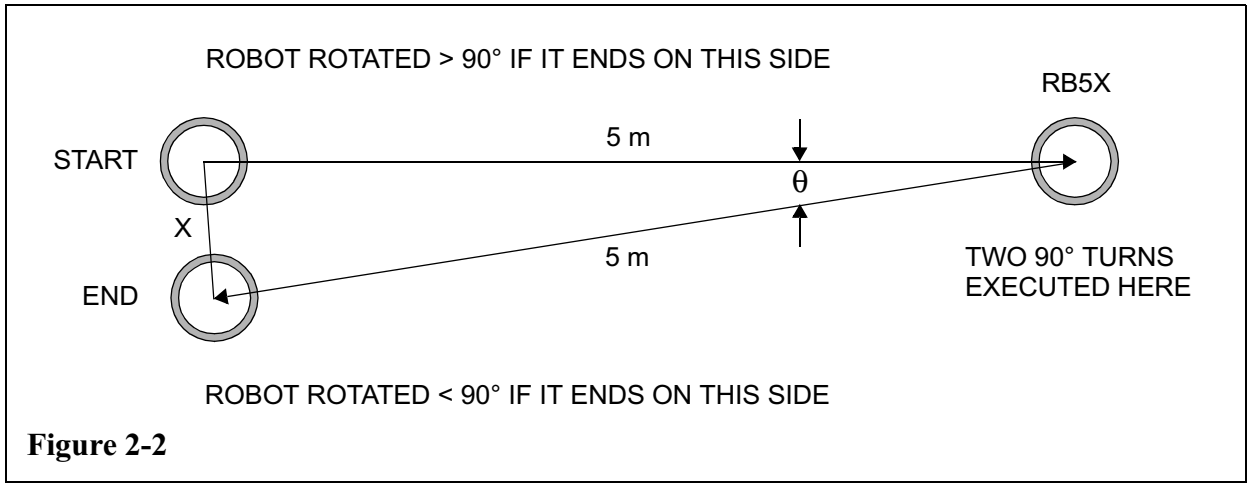


Figure 2-2

ROT90: An RCL Program to Test Rotation Accuracy

1. PREPARE THE ROBOT
2. MOVE DISTANCE FORWARD FOR THIS MANY FEET [5] (Note: distance is in "meters.")
3. SPIN RIGHT 90 DEGREES
4. WAIT FOR [1] SECONDS
5. SPIN RIGHT 90 DEGREES
6. MOVE DISTANCE FORWARD FOR THIS MANY FEET [5] (Note: distance is in "meters.")

If the robot rotates exactly 90° twice it will end where it started (X = 0) and you're done. If not, measure X and calculate the angle θ using:

$$\theta = 2\sin^{-1}\left(\frac{X}{10\text{ m}}\right)$$

Finally, calculate the rotational error as

$$\text{error} = \frac{180° - \theta}{180°}$$

Again, errors in excess of 0.01 should be corrected. You can do this by adjusting the calibration parameter SPIN.SECS/90DEG in RCL. The default value for this parameter is "1.38." To correct for the error measured in this Mission Problem use the formula:

SPIN.SECS/90DEG = 1.38 x (1 ± error)

with the ± applied in the opposite sense of the error. Use this new value to recalibrate RCL now. Procedures for setting this parameter are explained in the Initial Robot Calibration on page 30.

Exploring Further

A. To obtain data for your own error graph you'll need a sonar target. Any flat, hard board about 40 cm square works fine. Mount it securely on a ringstand, perpendicular to and facing the sonar beam, and exactly 1.00 m from RB5X's sonar transmitter.

RB5X must be programmed to measure and report the range to this target for distances from 1.00 m to 5.00 m, incrementing by 0.10 m. This will generate sufficient data to characterize the error.

The program solution given here operates directly on the NSC Tiny Basic variable D, and is independent of whether you have calibrated RB5X for metric or imperial units. We assume you have already run the initial sonar calibration, and thus are testing for residual errors. We also assume the sonar offset to be its default value of "73" (change this parameter in line 12 of the program if you obtained a different value for your robot).

RANGE: An RCL Program to Test Sonar Accuracy

```
1.  PREPARE THE ROBOT
2.  RESERVE THE TINY-BASIC VARIABLE NAMED [A]
3.  RESERVE THE TINY-BASIC VARIABLE NAMED [B]
4.  RESERVE THE TINY-BASIC VARIABLE NAMED [C]
5.  RESERVE THE TINY-BASIC VARIABLE NAMED [E]
6.  BEGIN A LOOP
7.    EXIT IF SONAR DISTANCE IS [MORE] THAN [0] FEET
8.  REPEAT THIS LOOP
9.  CALL THIS ROBOT TASK LINE [START]
10. WAIT FOR ANY BUMPER PRESS AND THEN WAIT [1] SECOND MORE
11. COMPILE THE BASIC STATEMENT [LINK#1100:DELAY20] WHICH MEANS [GET D]
12. COMPILE THE BASIC STATEMENT [E=(30*D - 30*73)/22] WHICH MEANS [CALC CM]
13. COMPILE THE BASIC STATEMENT [A=E/100] WHICH MEANS [CALC M]
```

```
14.  COMPILE THE BASIC STATEMENT [B=(E - 100*A)/10] WHICH MEANS [CALC DM]
15.  COMPILE THE BASIC STATEMENT [C=E - 100*A - 10*B] WHICH MEANS [CALC CM]
16.  SET THE TINY-BASIC VARIABLE [E] EQUAL TO [0]
17.  HONK THE HORN FOR [2] SECONDS
18.  WAIT FOR [1] SECONDS
19.  BEGIN A LOOP
20.    COMPILE THE BASIC STATEMENT [IF E=A GOTO 60] WHICH MEANS [EXIT LOOP]
21.    HONK THE HORN FOR [0.1] SECONDS
22.    WAIT FOR [0.2] SECONDS
23.    COMPILE THE BASIC STATEMENT [IF E=E+1] WHICH MEANS [INCREMENT E]
24.  REPEAT THIS LOOP
25.  SET THE TINY-BASIC VARIABLE [E] EQUAL TO [0]
26.  WAIT FOR [1] SECONDS
27.  HONK THE HORN FOR [2] SECONDS
28.  WAIT FOR [1] SECONDS
29.  BEGIN A LOOP
30.    COMPILE THE BASIC STATEMENT [IF E=A GOTO 75] WHICH MEANS [EXIT LOOP]
31.    HONK THE HORN FOR [0.1] SECONDS
32.    WAIT FOR [0.2] SECONDS
33.    COMPILE THE BASIC STATEMENT [IF E=E+1] WHICH MEANS [INCREMENT E]
34.  REPEAT THIS LOOP
35.  SET THE TINY-BASIC VARIABLE [E] EQUAL TO [0]
36.  WAIT FOR [1] SECONDS
37.  HONK THE HORN FOR [2] SECONDS
38.  WAIT FOR [1] SECONDS
39.  BEGIN A LOOP
40.    COMPILE THE BASIC STATEMENT [IF E=A GOTO 90] WHICH MEANS [EXIT LOOP]
41.    HONK THE HORN FOR [0.1] SECONDS
42.    WAIT FOR [0.2] SECONDS
43.    COMPILE THE BASIC STATEMENT [IF E=E+1] WHICH MEANS [INCREMENT E]
44.  REPEAT THIS LOOP
45.  JUMP TO THE LINE CALLED [START]
```

This program will, after a bumper press, report the sonar distance as a series of beeps in three groups representing “meters,” “decimeters” and “centimeters.” The groups will be separated by long (2 s) beeps. A series of short (0.1 s) beeps equal in number to each of those units of measure will follow each long beep. If no short beeps follow a long beep, that unit’s place is zero.

When you run the program, it waits for a bumper press before reporting the first sonar distance, after which you can move the sonar target at your leisure. When it’s properly position at the next distance, press any bumper and RB5X will report again. Proceed in this manner until you have acquired all the data from 1-5 m.

The error in sonar ranging should be no more than a few percent. If it’s larger, you need to run the sonar calibration routine described in the Initial Robot Calibration section. Even when properly calibrated, there is always some residual error due to the way NSC Tiny Basic truncates the values of all variables to integers. Further, because integers are limited to the range -32767 to +32767, the precision of these sonar measurements is limited. If you attempt to change line 12 to measure millimeters, the values overflow and return zero for the distance.

Line 12 of the program makes use of the relation

$$\text{DISTANCE} = \frac{\text{SONAR COUNT} - \text{SONAR OFFSET}}{\text{COUNTS/FT}}$$

where SONAR COUNT is the variable D returned by line 11. To convert this distance from its native "feet" into units of "cm," the numerical factor of 30 is included. Yes, we know there are actually 30.48 cm/ft, but NSC Tiny Basic does only accepts integers in its expression. You could use a pair of numbers closer to the ratio 30.48/22 (like 61/44) in this expression, but the increase in accuracy is not really required for the problems in this curriculum.

You will also see a slight increase in relative error with distance. This is due to an effect known as dispersion. Sound moves through air at a speed partly dependent on frequency. RB5X's sonar uses a "chirp" burst, consisting of a rapid frequency sweep that sounds like a "click" to your ears at close range. As this burst moves through the air it smears out nonlinearly over distance causing an increase in relative error. Your error graph should be similar to that of Fig. 2-3, but may differ slightly in slope, with a 1-2% increase in relative error over the 5 meters distance.

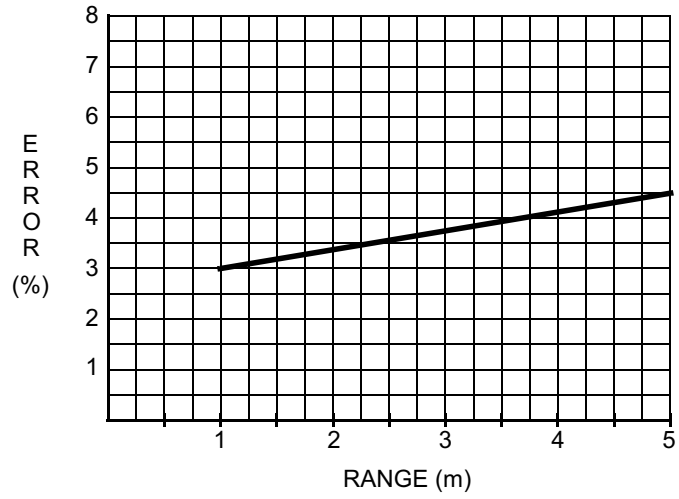


Figure 2-3

One final factor affecting errors in sonar distance measurements is local atmospheric conditions. The calibration parameter SONAR.COUNTS/FT has a default value of 22 based on the sonar unit's original calibration at STP (standard temperature and pressure). A change of ≈ 3% in the speed of sound would alter this value to 21 or 23. If your RB5X is operating under extreme conditions, it may be necessary to adjust this parameter. Here are the correction factors:

SPEED OF SOUND = 344 m/s ± 0.6 m/s/°C (higher temperatures increase the speed)
SPEED OF SOUND = 344 m/s ± 4.4 m/s/1000 m (higher altitudes decrease the speed)

If, for example, you were operating your RB5X in Denver, Colorado, in the winter, in an unheated room, the speed of sound could be as low as 335 m/s. In this case SONAR.COUNTS/FT should be reset to 23. On a hot day in Miami, Florida, mach could hit 355 m/s, requiring an adjustment in the other direction to 21.

B. The essay should mention the following advantages of sonar over laser ranging:

- can penetrate fog or dust
- lower power requirements
- works on transparent targets

It should also mention the following disadvantages:

- ineffective in a vacuum or very low pressure atmosphere
- lower maximum range
- slower response time

Refraction (bending) of the beam is a mutual disadvantage, as both can change direction while passing through layers of air or water with different temperatures and densities.

C. GRC's manual says the arm can lift from 3.3 - 5.0 N (12 - 18 ounces) implying a motor torque of about 1 N·m. When lifting the test mass, take advantage of the more efficient lifting geometry shown in Fig. 2-4. The current drawn by the arm motor is directly proportional to the torque "τ". To maximize F given a maximum torque you need to minimize L. Of course, this means the arm must be programmed to rotate about two axes while lifting, but you will gain around twice the lift capacity using this method.

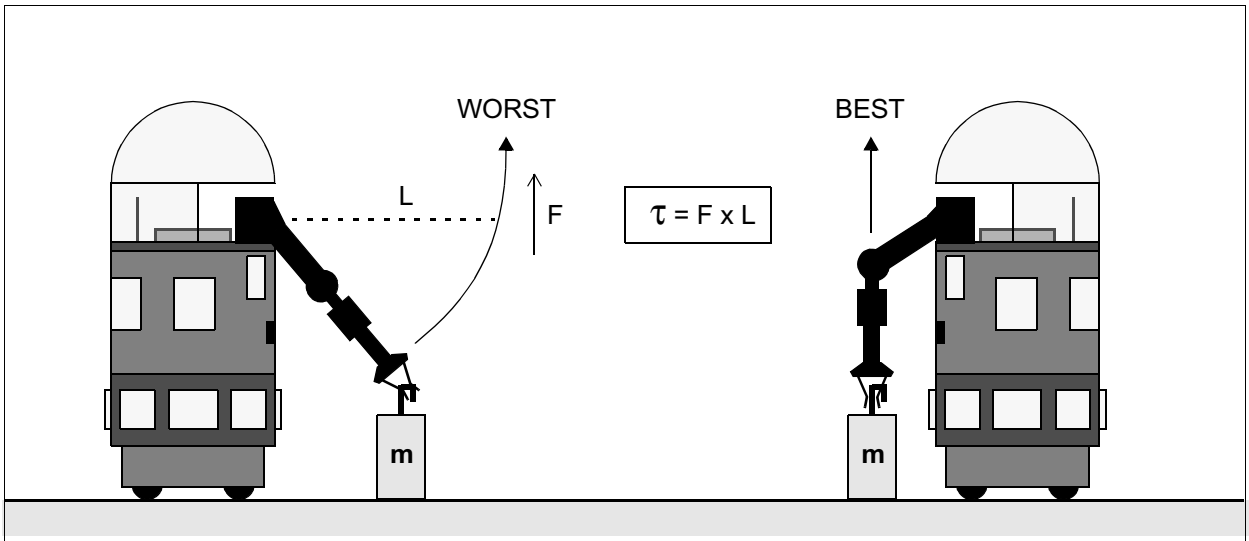


Figure 2-4

Starting with 3.0 N, and incrementing by 0.25 N, lift objects of increasing weight from ground level until the 6 A fuse blows. Alternatively, you could monitor the current using an ammeter in place of the fuse and stop when you hit 5.9 A. The fuse is located on the arm switch bracket. To facilitate these types of measurements in the future, you may want to construct a special "3AG fuse adapter" for your meter. You can solder meter leads to the sides of each end of a blown fuse, and simply insert it where needed for current measurements. Note that this puts your meter in series with the circuit — voltages cannot be measured using this arrangement.

Mission 3: Mapping

Mission Questions

A. RB5X's IR sensors could "see" a crevice by noticing how dark it is compared to solid ground. Of course, since the IR sensor is on the underside of the robot, RB5X may have already fallen into the crevice by the time it is detected — we will deal with this complication in Mission 4. RB5X's sonar and bumpers could detect rocks and mountains by echo and touch; rocks would trigger its bumpers but not its sonar, and mountains would trigger both.

B. If movement commands were given in terms of time (i.e., MOVE TIMED FORWARD), then the robot would need to know

1. how fast it moves (v) and
2. how long it has been moving (t).

With this information it could then find how far it has moved (d) by using the formula:

$d = v \cdot t.$

Alternatively, if movement commands were given in terms of distance (i.e., MOVE DISTANCE FORWARD) a simple running total could be kept in memory. Either way, RB5X would need an "accumulator" program similar to that described in the answer to Mission Question E.

C. RB5X would need to know its current coordinates (x_2,y_2). The straight-line distance can then be calculated using the point-to-point distance formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

which, when starting from RB5X's origin of (6,0), reduces to:

$$d = \sqrt{(x_2 - 6)^2 + (y_2)^2}$$

D. Several possibilities exist. Mars has a weak magnetic field compared to Earth, but a regular compass could be used. It should be located in view of RB5X's camera, e.g., on a rod extending to the front of the robot. The compass would need to compensate for RB5X's own magnetic field, as is done with compasses in recreational vehicles. An interesting exercise here is to let students use a compass and explore how far away it must be to avoid the effect of RB5X's own magnetic field. This is best done outside, away from the magnetic field of your building, wiring and all other large metallic objects. RB5X should be in motion while this test is done.

Of course, this method finds magnetic north as opposed to true north. For local navigation near the landing site this may be sufficient. True north could only be ascertained by reference to an visible astronomical reference point, the most obvious of which is the North Star, Polaris.

Mars' axial tilt (25.2°) and orbital inclination (1.9°) are, by pure coincidence, remarkably close to those of Earth (23.5° and 0°, respectively). Consequently, on both planets, Polaris appears very near the North Celestial Pole. RB5X could use its video camera to capture prolonged scans of the night sky in directions separated by four 90° rotations. One of them would show the characteristic concentric star trail patterns of Fig. 3-1. The direction to north is on the horizon directly below the point of apparent star rotation, a.k.a. the North Celestial Pole. [See also the answer provided for Exploring Further B.]

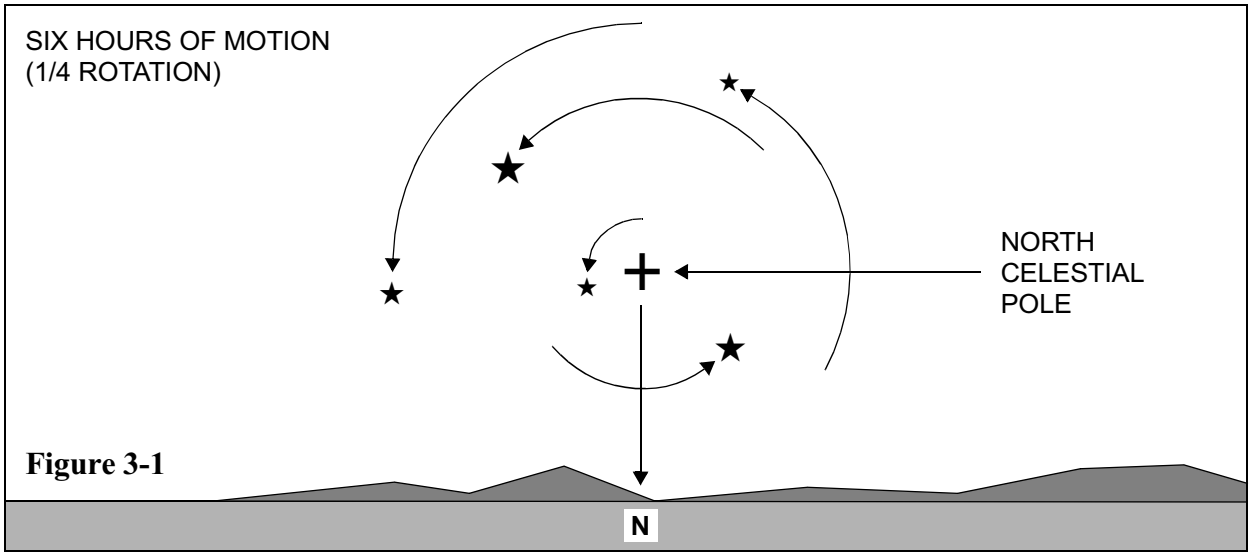


Figure 3-1

E. The linear motions are irrelevant over typical roving distances. At 45° latitude on Mars one would need to travel 42 km straight east (or west) before the direction of north would change by even 1°. At lower latitudes the distance could be larger yet. What RB5X needs to do is account for all the turns it makes while exploring.

Every program line in which a turn is made must be followed by additional lines that update a variable acting as an "accumulator" for rotations. Knowing the current value of this variable will tell RB5X the direction it is facing at any time. Since the original direction is north, or 0°, every time RB5X executes a turn the following numbers must be added to a running total:

- left turn: -90°
- right turn: +90°
- rotate clockwise X degrees: +X°
- rotate counter-clockwise X degrees: -X°

Here's an example of the lines that would have to follow a SPIN RIGHT 90 DEGREES command:

```
1. COMPILE THE BASIC STATEMENT [A=A+90] WHICH MEANS [ACCUMULATE ANGLE]
2. COMPILE THE BASIC STATEMENT [IF=A>360 GOTO 4] WHICH MEANS [OVER 360]
3. COMPILE THE BASIC STATEMENT [IF=A<0 GOTO 6] WHICH MEANS [UNDER 0]
4. COMPILE THE BASIC STATEMENT [A=A-360] WHICH MEANS [RESET ANGLE]
5. JUMP TO THE LINE CALLED [ANGLEOUT]
6. COMPILE THE BASIC STATEMENT [A=360+A] WHICH MEANS [RESET ANGLE]
7. CALL THIS LINE ROBOT TASK LINE [ANGLEOUT]
```

Note that the GOTO statements must refer to compiled program lines, not the RCL lines as shown.

Note also that we must allow for the possibility of A going out of bounds. The conditional tests in lines 2 and 3 ensure A will remain between 0° and 360°.

Of course, since the programmed angles of rotation are never executed exactly, the error in A will grow without limit as the number of turns increases. The only way around this problem is to get a fix on an external frame of reference. Several methods could be adapted to work with RB5X, and were discussed in the answer to Mission Question D.

Mission Problem

RB5X must be programmed to locate the mountain, determine its coordinates, return to the landing site, and report the mountain coordinates via LED signals. We assume the robot starts at its landing site at coordinates (6,0) facing north, and that your diorama dimensions are those of the Sample Diorama provided (10x10). We also assume you have calibrated RCL for metric units.

This is a fairly complex problem, so we'll break it into three separate programs:

MAPPER: locates and records the mountain coordinates
HOME: returns RB5X to its landing site
REPORT: signals mountain coordinates with flashing LEDs

Each, in sequence, will call the next as a task in its final line To save on variables, the coordinates will be stored as the individual digits of variables A and B as follows:

$A = 0\ X_4\ X_3\ X_2\ X_1$

$B = 0\ Y_4\ Y_3\ Y_2\ Y_1$

where the digits of each can take on values from 0 to 9, and a total of up to four coordinates can be stored. The first digit is unavailable since its maximum value is 3 due to the integer size limit of 32,767 imposed by NSC Tiny Basic. For example, if the values are

A = 04902
B = 02233

The coordinates represented would be (3,2), (3,0), (2,9) and (2,4). If less than four coordinates are needed, a value of (0,0) will indicate "no coordinate." We can do this since the origin (0,0) is required to be an empty grid block and thus is not a possible mountain coordinate. The variable E will be used as the exponent (x1, x10, x100 or x1000) that enters coordinates in sequential digit places (ones, tens, hundreds or thousands).

The solution requires the entry of three RCL programs: MAPPER, HOME and REPORT. When all three are entered, running MAPPER will call the other two as needed.

MAPPER: An RCL Program to Map Mountain Coordinates

1. PREPARE THE ROBOT
2. RESERVE THE TINY-BASIC VARIABLE NAMED [A]
3. RESERVE THE TINY-BASIC VARIABLE NAMED [B]
4. RESERVE THE TINY-BASIC VARIABLE NAMED [C]
5. RESERVE THE TINY-BASIC VARIABLE NAMED [E]
6. RESERVE THE TINY-BASIC VARIABLE NAMED [F]
7. RESERVE THE TINY-BASIC VARIABLE NAMED [G]
8. RESERVE THE TINY-BASIC VARIABLE NAMED [H]
9. CLEAR ALL ITEMS
10. BEGIN A LOOP
11. EXIT IF SONAR DISTANCE IS [MORE] THAN [0] FEET (Note: distance is in "meters.")
12. REPEAT THIS LOOP
13. SPIN LEFT 90 DEGREES
14. MOVE DISTANCE FORWARD FOR THIS MANY FEET [3] (Note: distance is in "meters.")
15. SPIN RIGHT 90 DEGREES
16. SET THE TINY-BASIC VARIABLE [E] EQUAL TO [1]
17. BEGIN A LOOP
18. COMPILE THE BASIC STATEMENT [LINK#1100:DELAY20] WHICH MEANS [GET D]
19. COMPILE THE BASIC STATEMENT [C=(((30*D - 30*73)/22) + 42)/50] WHICH MEANS [CALC GB]

Line 19 calculates C as the number of grid blocks to the sonar target. The expression (30*D - 30*73)/22 is the distance to the target in "cm." The addition of 42 cm allows for the offsets created by the robot's position in the center of a grid block and the face of the mountain being near the edge of a grid block (Fig. 3-2). The division by 50 converts this distance into an integral number of grid blocks. If this value exceeds 9, then the object is outside the diorama.

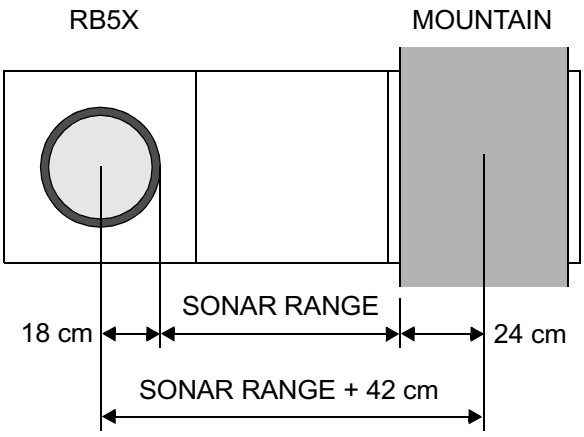


Figure 3-2: (view from above)

20. COMPILE THE BASIC STATEMENT [IF (C>9) AND (A>0) GOTO 72] WHICH MEANS [MT END]
21. COMPILE THE BASIC STATEMENT [IF C>9 GOTO 58] WHICH MEANS [CHECK C]
22. COMPILE THE BASIC STATEMENT [A=A+E*F] WHICH MEANS [RECORD X-SUB-F]
23. COMPILE THE BASIC STATEMENT [B=B+E*C] WHICH MEANS [RECORD Y-SUB-F]
24. COMPILE THE BASIC STATEMENT [E=10*E] WHICH MEANS [INCREMENT E]
25. COMPILE THE BASIC STATEMENT [IF E=10000 GOTO 72] WHICH MEANS [DONE]
26. SPIN RIGHT 90 DEGREES
27. MOVE DISTANCE FORWARD FOR THIS MANY FEET [0.5] (Note: distance is in "meters.")
28. COMPILE THE BASIC STATEMENT [F=F+1] WHICH MEANS [INCREMENT F]
29. SPIN LEFT 90 DEGREES
30. REPEAT THIS LOOP
31. COMPILE THE BASIC STATEMENT [E=E/10] WHICH MEANS [DECREMENT E]
32. HOME

Remember that the program lines referred to in any GOTO statement are the NSC Tiny Basic line numbers, and not the RCL line numbers. This also applies to all programs called as tasks within MAPPER (such as HOME), since NSC Tiny Basic compiles all tasks in a program into a single NSC Tiny Basic program with sequentially numbered lines. See RCL Programming — Setup and Methods for further explanation of how to identify these line numbers.

The program HOME called in the last line of MAPPER must return RB5X to its landing site from wherever MAPPER left it. MAPPER left RB5X at coordinates (F,0) facing north. Note that the variables used in MAPPER are already reserved, and can still be used (or recycled).

HOME: An RCL Program to Return RB5X to the Landing Site

```
1.  PREPARE THE ROBOT
2.  SET THE TINY-BASIC VARIABLE [G] EQUAL TO [1]
3.  COMPILE THE BASIC STATEMENT [IF F>6 GOTO 82] WHICH MEANS [WHICH WAY?]
4.  SPIN RIGHT 90 DEGREES
5.  COMPILE THE BASIC STATEMENT [C=6-F] WHICH MEANS [CALC BLOCKS EAST]
6.  JUMP TO THE LINE CALLED [GO HOME]
7.  SPIN LEFT 90 DEGREES
8.  COMPILE THE BASIC STATEMENT [C=F-6] WHICH MEANS [CALC BLOCKS WEST]
9.  CALL THIS ROBOT TASK LINE [GO HOME]
10. BEGIN A LOOP
11.    MOVE DISTANCE FORWARD FOR THIS MANY FEET [0.5]  (Note:  distance is in "meters.")
12.   COMPILE THE BASIC STATEMENT [IF G=F GOTO 96] WHICH MEANS [DONE MOVING]
13.   COMPILE THE BASIC STATEMENT [G=G+1] WHICH MEANS [INCREMENT G]
14. REPEAT THIS LOOP
15. COMPILE THE BASIC STATEMENT [IF F>6 GOTO 102] WHICH MEANS [FACING WEST]
16. SPIN LEFT 90 DEGREES
17. JUMP TO THE LINE CALLED [AT HOME]
18. SPIN RIGHT 90 DEGREES
19. CALL THIS ROBOT TASK LINE [AT HOME]
20. REPORT
```

The program HOME returns RB5X to its landing site, leaving the robot facing north. The variables A and B, which contain the mountain coordinates, must now be "read out" with a series of signals from the robot's LEDs.

The numbering scheme for the four individually controllable LEDs is shown in Fig. 3-3. Since RB5X is facing north, we'll first use the front and back pairs to signal the latitude, and then use the left and right pairs to signal the longitude.

Breaks between coordinates will be signalled by all lights flashing.

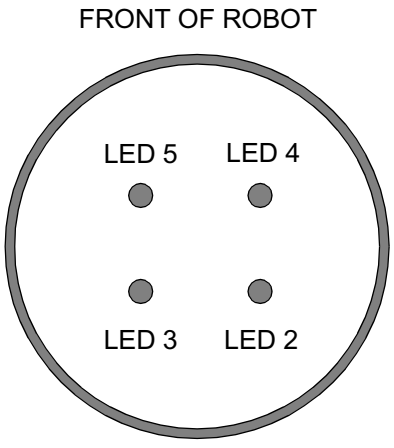


Figure 3-3

REPORT: An RCL Program to Report Mountain Coordinates Using LEDs

```
1.  PREPARE THE ROBOT
2.  CALL THIS ROBOT TASK LINE [EXTRACT]
3.  SET THE TINY-BASIC VARIABLE [C] EQUAL TO [1]
4.  COMPILE THE BASIC STATEMENT [F=A/E] WHICH MEANS [READ LAT]
5.  COMPILE THE BASIC STATEMENT [A=A - F*E] WHICH MEANS [SHIFT PLACE]
6.  COMPILE THE BASIC STATEMENT [G=B/E] WHICH MEANS [READ LON]
7.  COMPILE THE BASIC STATEMENT [B=B - G*E] WHICH MEANS [SHIFT PLACE]
8.  TURN ON ALL THE LIGHTS
9.  WAIT FOR 3 SECONDS
10. TURN OFF ALL THE LIGHTS
11. BEGIN A LOOP
12.   WAIT FOR 1 SECONDS
13.   TURN ON LED NUMBER 2
14.   TURN ON LED NUMBER 3
15.   WAIT FOR 1 SECONDS
16.   COMPILE THE BASIC STATEMENT [C=C+1] WHICH MEANS [INCREMENT C]
17.   COMPILE THE BASIC STATEMENT [IF C>F GOTO 131] WHICH MEANS [EXIT LAT]
18.   TURN ON LED NUMBER 4
19.   TURN ON LED NUMBER 5
20.   WAIT FOR 1 SECONDS
21.   TURN OFF LED NUMBER 4
22.   TURN OFF LED NUMBER 5
23. REPEAT THIS LOOP
24. TURN OFF LED NUMBER 2
25. TURN OFF LED NUMBER 3
26. SET THE TINY-BASIC VARIABLE [C] EQUAL TO [1]
27. BEGIN A LOOP
28.   WAIT FOR 1 SECONDS
29.   TURN ON LED NUMBER 3
30.   TURN ON LED NUMBER 5
31.   WAIT FOR 1 SECONDS
32.   COMPILE THE BASIC STATEMENT [C=C+1] WHICH MEANS [INCREMENT C]
33.   COMPILE THE BASIC STATEMENT [IF C>G GOTO 148] WHICH MEANS [EXIT LON]
34.   TURN ON LED NUMBER 2
35.   TURN ON LED NUMBER 4
36.   WAIT FOR 1 SECONDS
37.   TURN OFF LED NUMBER 2
38.   TURN OFF LED NUMBER 4
39. REPEAT THIS LOOP
40. TURN OFF LED NUMBER 3
41. TURN OFF LED NUMBER 5
42. COMPILE THE BASIC STATEMENT [IF E=1 GOTO 153] WHICH MEANS [DONE]
43. COMPILE THE BASIC STATEMENT [E=E/10] WHICH MEANS [DECREMENT E]
44. JUMP TO THE LINE CALLED [EXTRACT]
45. CALL THIS ROBOT TASK LINE [DONE]
```

RB5X's signals can be read as follows. When the back pair of LEDs go on, count the number of times the front LEDs flash to get the latitude. When the left pair of LEDs go on, count the number of times the right LEDs flash to get the longitude. Between coordinate pairs all lights will flash for three seconds before the next readout begins.

Exploring Further

A. For RB5X to discriminate between sonar targets inside and outside the diorama, the distance to the edge of the diorama must be calculated from its current coordinates. This distance can then be compared numerically to the measured sonar range.

To solve this problem, the following information is required:

- current coordinates of RB5X
- direction RB5X is facing when measurement is made
- dimensions of diorama (grid blocks longitude x grid blocks latitude)

The solution provided assumes RB5X's current coordinates are stored as the ordered pair (A,B), where A and B are the reserved variables for longitude (0-9) and latitude (0-9) respectively. We also assume RB5X is facing in the direction represented by the reserved variable C, whose value can be 0, 90 270 or 360 (see Mission Question E). Finally, we assume your diorama measures 10 x 10 grid blocks — if not, you'll need to adjust the parameters in program lines 12, 15, 18 and 21 to accommodate your diorama's dimensions.

VERIFY: An RCL Program to Determine the Validity of a Sonar Target

```
1.  PREPARE THE ROBOT
2.  RESERVE THE TINY-BASIC VARIABLE NAMED [E]
3.  BEGIN A LOOP
4.    EXIT IF SONAR DISTANCE IS [MORE] THAN [0] FEET  (Note:  distance is in "meters.")
5.  REPEAT THIS LOOP
6.  COMPILE THE BASIC STATEMENT [LINK#1100:DELAY20] WHICH MEANS [GET D]
7.  COMPILE THE BASIC STATEMENT [E=(30*D - 30*73)/22] WHICH MEANS [CALC CM]
8.  COMPILE THE BASIC STATEMENT [IF C=90 GOTO 44] WHICH MEANS [EAST]
9.  COMPILE THE BASIC STATEMENT [IF C=180 GOTO 47] WHICH MEANS [SOUTH]
10. COMPILE THE BASIC STATEMENT [IF C=270 GOTO 50] WHICH MEANS [WEST]
11. CALL THIS ROBOT TASK LINE [NORTH]
12. COMPILE THE BASIC STATEMENT [IF E>50*(9-B)+7 GOTO 56] WHICH MEANS [OUT]
13. JUMP TO THE LINE CALLED [IN]
14. CALL THIS ROBOT TASK LINE [EAST]
15. COMPILE THE BASIC STATEMENT [IF E>50*(9-A)+7 GOTO 56] WHICH MEANS [OUT]
16. JUMP TO THE LINE CALLED [IN]
17. CALL THIS ROBOT TASK LINE [SOUTH]
18. COMPILE THE BASIC STATEMENT [IF E>50*A+7 GOTO 56] WHICH MEANS [OUT]
19. JUMP TO THE LINE CALLED [IN]
20. CALL THIS ROBOT TASK LINE [WEST]
21. COMPILE THE BASIC STATEMENT [IF E>50*B+7 GOTO 56] WHICH MEANS [OUT]
22. CALL THIS ROBOT TASK LINE [IN]
23. HONK THE HORN FOR [3] SECONDS
24. CALL THIS ROBOT TASK LINE [OUT]
```

RB5X will honk its horn when the target is inside the diorama and do nothing when the target is outside the diorama. Of course, linked to other programs, other more useful actions could ensue when lines 23 and 24 are reached.

B. RB5X's video camera allows its teleoperator to observe where the Sun rises and sets. If you connect vectors from RB5X to the sunrise and sunset points, the bisector of the angle formed will point approximately north and south (Fig. 3-4). To construct this geometry RB5X has to follow the Sun for one day, rotating incrementally in place as it watches, and noting those points on the horizon where the Sun rises and sets.

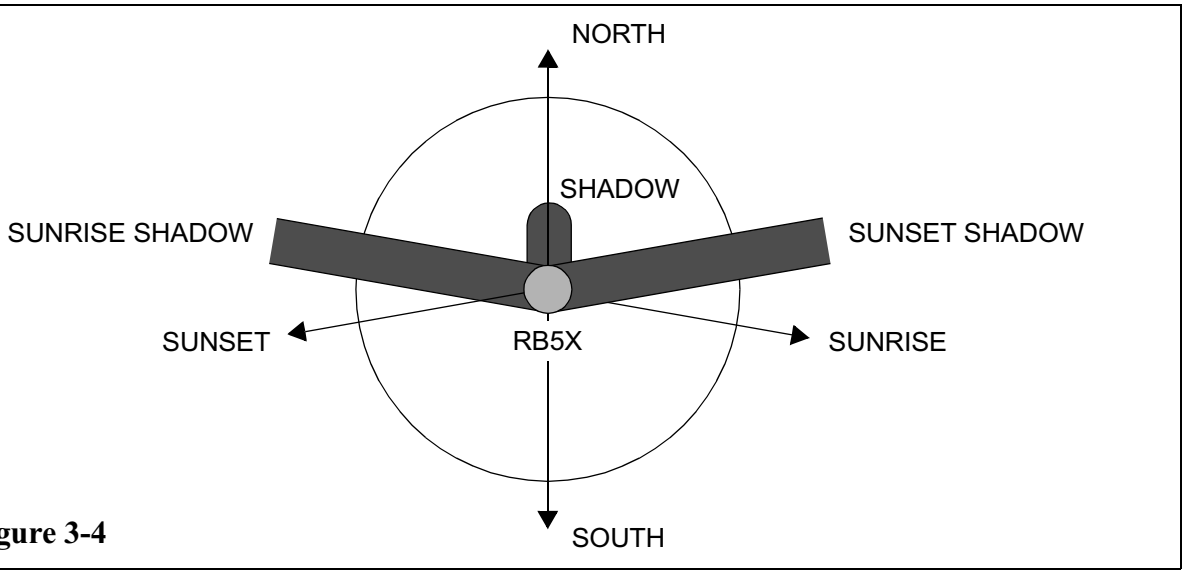


Figure 3-4

It may be possible to obtain the direction of north with a single measurement. If RB5X is in the northern hemisphere, its shadow at noon (halfway between sunrise and sunset) will point north. In the southern hemisphere it will point south. In the tropics, the direction of its shadow at noon depends on what the season is, so additional information would be needed.

The video camera has a vertical field of view of 50° and is mounted at a height of 37 cm above ground level, thus points on the ground closer than 75 cm will not be visible to the teleoperator. RB5X has a height of about 53 cm, so with Sun elevations > 35° it will cast a shadow < 75 cm in length. Depending on season and latitude, RB5X may or may not be able to see its own shadow.

Locations closer to the equator have a higher Sun at noon (and shorter shadows). RB5X may be able to observe the shadow of a nearby rock or, lacking that, could always hold up its robotic arm to create a shadow of greater length. In any case, the shadow cast at noon is the shortest shadow of the day, so with continuous monitoring of the shadow length a time signal is not required.

C. RB5X's sonar is normally aimed forward in a horizontal beam. If the beam could somehow be redirected toward the ground in front of the robot, the presence of a crevice (i.e., the absence of ground) would be signaled by a sudden increase in time and distance for the returning echo, or by no echo at all — in which case the sonar ranging subroutine would time-out.

Certain types of ground cover could produce a false crevice indication. Materials that strongly absorb and/or scatter the sonar beam could produce the same acoustic signature as a crevice, that is, no echo at all. Loose gravel or thick vegetation are possible candidates, as well as large flat rocks tilted such that the sonar beam is deflected off axis.

Fig. 3-5 shows one possible arrangement for such a "look-down" device. Note that in order for RB5X to hear the returning sonar echo, the reflector must be oriented at 45° to horizontal. This follows from a law of wave propagation: the angle of incidence equals the angle of reflection.

The ultrasound is generated at a frequency (f) of around 50 kHz. The speed (v) of sound in air is 340 m/s (1100 ft/s). Wavelength (λ) is related to these quantities by the formula: $\lambda = v/f$. For the ultrasound unit RB5X uses, this works out to about 0.7 cm (1/4 inch).

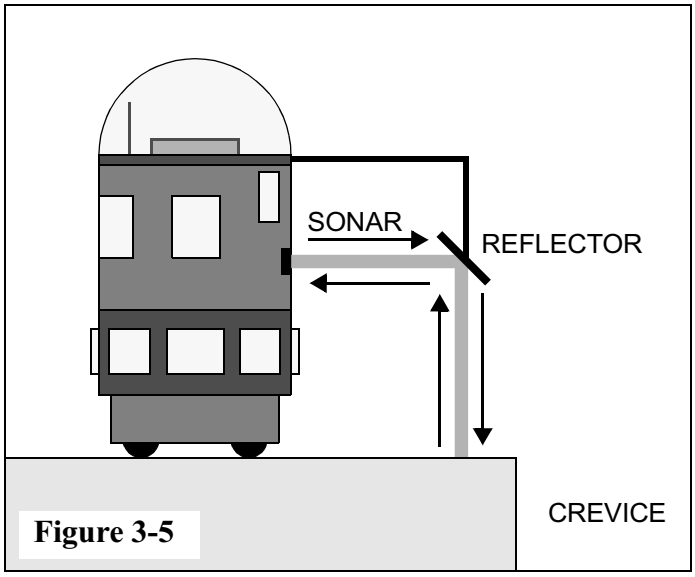


Figure 3-5

Another law of wave propagation requires the reflecting barrier to have dimensions on the order of one wavelength to efficiently reflect a wave. Thus, objects smaller than 0.7 cm are effectively invisible to RB5X's sonar. Theoretically, a 0.7 cm reflector should suffice for this arrangement, but since the beam spreads out at a 20° angle and weakens with distance, the farther the reflector, the larger it must be for reliable detection. To ensure a good echo you want to intercept as much as the beam as possible.

At the robot-to-reflector distance shown in Fig. 3-5 the beam diameter will be around 10 cm, and after rebounding back from the ground it will have spread to around 30 cm, depending on ground roughness and surface scattering. You should use a reflector at least 30 cm square. The material of choice should be flat, smooth, light and rigid. Aluminum or hardboard are good choices since both are inexpensive and easy to work with.

The following lines of code show how a crevice could be detected using the sonar modification of Fig. 3-5, where the sonar beam path length to normal solid ground is ≈ 50 cm.

HEARCREV: An RCL Program to Detect Crevices using Sonar

```
1. PREPARE THE ROBOT
2. BEGIN A LOOP
3.    MOVE DISTANCE FORWARD FOR THIS MANY FEET [0.1]  (Note:  distance is in "meters.")
4.    WAIT FOR [2] SECONDS
5.    EXIT IF SONAR DISTANCE IS [MORE] THAN [0.55] FEET  (Note:  distance is in "meters.")
6. REPEAT THIS LOOP
7. STOP ALL MOTION
```

RB5X will advance incrementally, 10 cm at a time, testing after each advance for an increase in sonar range beyond the safe value of 50 cm. We use 55 cm in line 5 rather than 50 cm to allow for normal variation in the sonar ranging. When the crevice is detected, the robot will stop.

Mission 4: Hazard Avoidance

Mission Questions

A. Fig. 4-1 shows the geometry of the orbits of Earth and Mars. At closest approach, the distance to Mars is:

$d = 2.28 \times 10^8$ km - 1.50×10^8 km
$d = 0.78 \times 10^8$ km = 78,000,000 km

Mars' greatest distance from Earth is:

$d = 2.28 \times 10^8$ km + 1.50×10^8 km
$d = 3.78 \times 10^8$ km = 278,000,000 km

Radio signals travel at the speed of light (300,000 km/s) so the one-way travel time for each of these distances is:

$t = d/v = 0.78 \times 10^8$ km/3.00×10^5 km/s
$t = 260$ s = 4.3 min, and

$t = d/v = 2.78 \times 10^8$ km/3.00×10^5 km/s
$t = 927$ s = 15.4 min

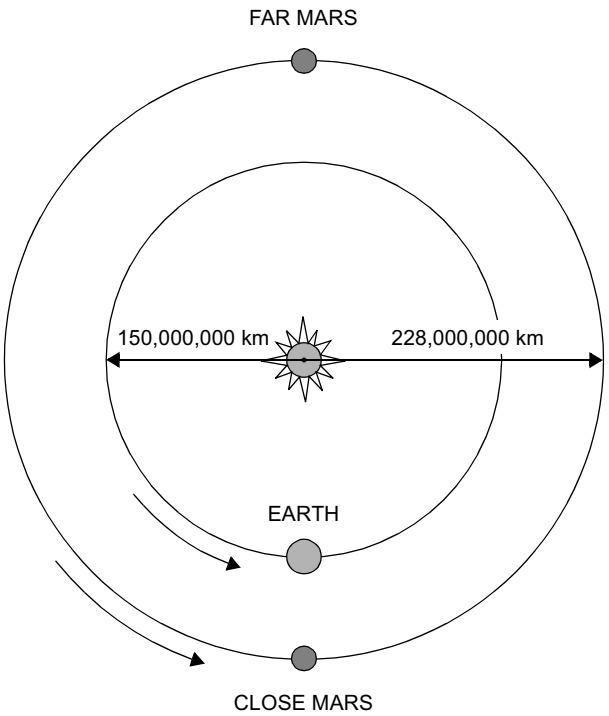


Figure 4-1

B. An electric motor stalled by an outside force can indeed blow a fuse. When an electric motor first starts, it presents no sizeable resistance to the incoming electrons. Huge currents, sometimes 5-6 times the equilibrium current, rush into it. As the motor comes up to speed it develops what is called a "back voltage" from induction between its field coils and magnets. As the back voltage rises with increasing RPM, opposing the supplied voltage, the current decreases and will stabilize at some lower equilibrium value. If the motor is slowed or stopped by an outside force, the back voltage drops and the current will surge. If the surge lasts for too long, a fuse will blow, a breaker will trip, or damage will occur from ohmic heating.

Motors can also be damaged by too high a speed. This could occur when a motor is driven by an input voltage that exceeds its specifications. If this happens, mechanical damage from vibration or centrifugal force is as likely as electrical damage.

For a mobile autonomous machine like RB5X, a stalling condition could occur if the robot

- runs into an object it cannot push
- is made to pull or carry too large a weight
- attempts to move over sticky or soft surfaces (like hot asphalt or sand)

C. The center of gravity (c.o.g.) is the average position of the mass comprising an object. For any simply shaped homogenous object, this point corresponds to the geometric center. It is related to the stability of an object by a very simple scheme: if the c.o.g. is moved beyond the last point of support the object will topple (Fig. 4-2).

RB5X is extremely stable. Its battery pack is the single heaviest component, and is mounted low in the interior of the robot. RB5X's c.o.g. is located approximately 18 cm above the bottom of its wheels, and very near the center of its cross-sectional area. As can be determined by experiment, the RB5X will not topple until its tilt angle exceeds 45° (±5° depending on attached accessories). This is far more stable than typical "off road" vehicles, many of which will roll with < 30° tilt.

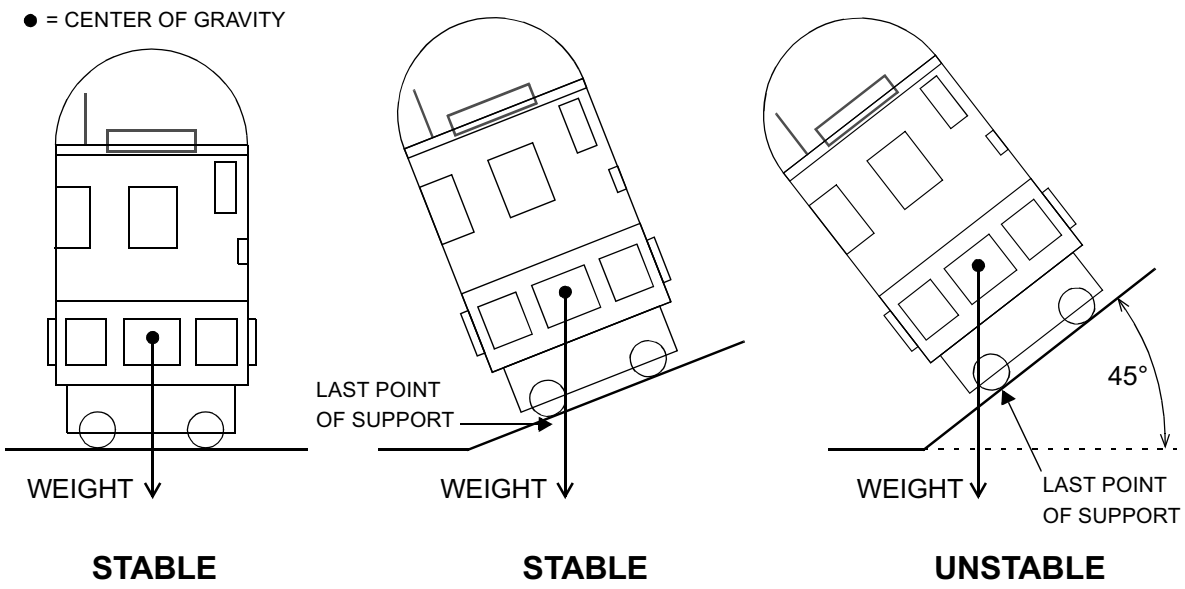


Figure 4-2

To rate the other robotic explorers for stability you need to make visual estimates of the locations of their c.o.g.'s — not an easy task, since you can't look inside them to see where their heaviest components are located. Still, trusting that the engineers who designed them understood the c.o.g. principle, let us assume the c.o.g. was located at "h/3" (a point 1/3 of the way up from the base of the central mass). This is, admittedly, not an exact method. But it at least provides a basis for comparison. Using a protractor, and compensating subjectively for the weight of any projecting components, we estimated the following maximum tilt angles (±5°):

Luna 9: 80°
Surveyor: 75°
Viking: 70°
Sojourner: 60°
Luna 16: 60°
Lunokhod: 50°
Venera 13: 45°
Venera 7: 30°
Ranger: ∞ (not designed for landing)

D. Caution: we do not recommend you actually try this exercise — the arm has not been rated for the stresses it would suffer therein. This Question is intended more as a "thought experiment" to be solved by approximate quantitative methods.

If RB5X has toppled, it needs to roll itself into the position shown in Fig. 4-3 (shoulder up and forearm down). From this position, it should be just barely able to right itself using its arm as a "second class lever." Note how the robot's weight acts as the load (F_o) through its center of mass (see Mission Question C solution). For all levers of any class:

$$M = \frac{F_o}{F_i} = \frac{L_i}{L_o} \Rightarrow F_o = M \cdot F_i \qquad \text{(where M is the mechanical advantage)}$$

Using measured values for the lever arms indicates a value of $M \approx 4.5$. RB5X (plus arm) weighs about 120 N. This means an input force of 27 N is required to <u>initiate</u> the self-righting; as RB5X tilts upward the value of L_o continuously decreases, increasing M and decreasing the required F_i. When the tilt angle reaches 45°, RB5X's own weight will complete the self-righting process as the c.o.g. moves beyond the last point of support (fulcrum).

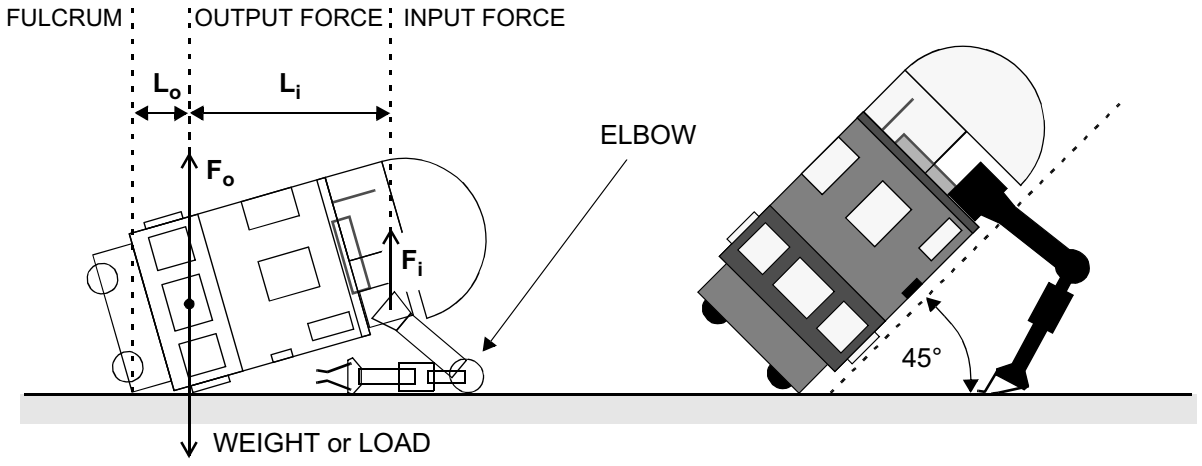


Figure 4-3

Note that the folded arm itself is a lever, the output of which is F_i in the preceding analysis. GRC rates the arm with a maximum lifting force of 5.0 N. This implies an arm motor torque of around 1 N·m (see Mission 2, Exploring Further C solution). Using the elbow motor to rotate the 0.20 m forearm generates an upward force of

$$F = \frac{\tau}{L} = \frac{1 \text{ N·m}}{0.20 \text{ m}} = \underline{25 \text{ N}}$$

which is probably within the uncertainty range for the estimated 27 N F_i. Interestingly, the author performed a direct measurement of F_i and found it to be 20 N instead of the estimated 27 N. This was likely due to an asymmetry of the internal mass distribution.

So can a toppled RB5X right itself? Theory says "probably," but don't risk an arm to find out — unless your budget can support this type of pure research.

Mission Problem

The following code is one possible solution.

SEECREV: An RCL Program to Detect Crevices Using IR

```
1. PREPARE THE ROBOT
2. MOVE FORWARD
3. BEGIN A LOOP
4.    COMPILE THE BASIC STATEMENT [X=#02] WHICH MEANS [TURN ON IR]
5.    COMPILE THE BASIC STATEMENT [@#7801=@#7801 OR X] WHICH MEANS [SET BIT]
6.    COMPILE THE BASIC STATEMENT [IF (@#7802 AND #40)=64 GOTO 37] WHICH MEANS [EXIT]
7. REPEAT THIS LOOP
8. STOP ALL MOTION
9. TURN ON THE FLASHING LIGHTS
```

Note: the loop exit condition "=64" will sense dark colors against a light background. If the stock RCL command EXIT IF TAPE SENSED is used to leave the loop, the default exit condition of "=0" will be applied and light colors will be sensed against a dark background. The crevice is a dark area against a light background, so lines 4-6 must replace the EXIT IF TAPE SENSED command.

In either case, if the IR does not sense the transition between light and dark surfaces, first check for dust or lint accumulated around the photodiode. If that doesn't solve the problem, you may need to adjust the photodiode sensitivity with potentiometer R30 on the CPU board. See your Reference Manual for instructions on this procedure.

If everything functions properly, SEECREV will cause RB5X to stop when it is about halfway over the "edge" of the crevice. As part of a larger program, SEECREV would be followed by a command to MOVE DISTANCE BACKWARD and continue exploration in a safer direction.

Exploring Further

A. Whichever sensor is used (sonar, bumper or IR) the time to stop is limited mainly by RB5X's inertia and not by how long it takes the onboard processor receive input and send a stop signal to the motor relays. Inertia will not be a large factor itself, since the low-geared drive system does not permit much of a "coasting roll." To measure the stopping distance you'll need to modify the program SEECREV provided in the Mission Problem solution. Save the modified program under a new name, e.g., SEECREV2. The diorama crevice can be used for this test.

1. Change SEECREV as follows: delete line 2, and change line 6 to read "... GOTO 36". Now the robot won't move forward, but the IR sensor will still accept input. Run the program and slowly slide the robot forward toward the crevice until the LEDs light. At that point, mark the position of the robot on the floor as precisely as possible.
2. Undo your changes to SEECREV. Move RB5X backwards about a meter and run the program. Mark the final position of the robot on the floor. It won't be very far past the first mark.
3. Measure the distance (d) between the two marks.

Assuming the robot decelerates uniformly from its original speed (v_o) to a stop ($v_f = 0$), we can use kinematics to calculate the effective response time.

$$\Delta t = \frac{d}{v_{ave}} = \frac{d}{\frac{v_o + v_f}{2}} = \frac{2d}{v_o}$$

Using RB5X's cruising speed of (0.12 m/s) and a typical value for stopping distance (0.02 m), you should get a value for Δt of around 0.33 s.

This is a fairly rapid stopping time, involving about 0.4 g's deceleration. On a "proper" diorama surface these values are typical. On ice or wet tile the results would differ significantly.

B. The obvious solution is to construct and attach a periscope as shown in Fig. 4-4. There's only a scant 2.8 cm clearance between chassis and ground, but that's enough room for a custom scope built from black ABS pipe or a cardboard tube. The scope should be flat black, inside and out, to reduce the scattering of ambient light. The two IR diodes are recessed another 2.5 cm, protruding down from the motor controller board. The gap between the sensor and periscope is not relevant, since all we are doing is laterally shifting RB5X's view of the ground.

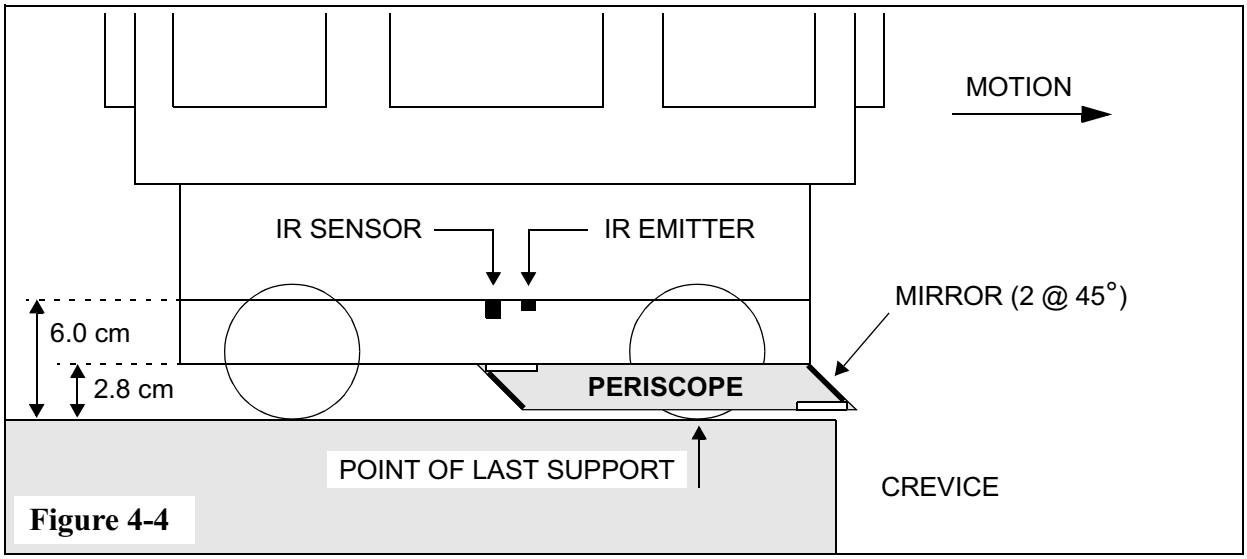


Figure 4-4

A variation on this periscope idea is to use "light pipes" to conduct the transmitted and reflected IR beam to and from the front of the robot. Actual fiber-optic bundles could be bent as needed to establish the beam paths. Less expensive light pipes can be made from ½ inch solid acrylic rods; when heated properly these rods can be deformed to the required shape.

The elegant solution is to physically move the IR diodes to the front of the chassis, but since they are soldered into the circuit board this is not an easy option. If you are skilled in electronics, you could wire a second sensor (950 nm IR type TIL39) and emitter (type TIL413) in parallel with the originals, and a micro-DPDT switch to select between them. You can access the wiring to these diodes (D2 and D3 in the Reference Manual schematic) near where connector J16 attaches to the motor controller card.

Note: If you move the diodes, the charger-finder program as provided in the Reference Manual will no longer function properly. The program could, however, be modified to account for the new diode positions should you choose to make this hardware alteration permanent.

All of these solutions work nicely with the simulated crevice on the diorama, provided the robot has been configured to detect dark against a light background. They would also work with a real crevice, since the ground always reflects more IR than empty air. This assumes, of course, that the "light state" of the sensor has been adjusted to compensate for the average ground reflectivity (typical "dirt" is not very reflective). The photodiode's sensitivity is adjusted via potentiometer R30 on the CPU board. See your Reference Manual for instructions on locating and setting R30.

C. Mission Question C established RB5X's maximum tilt angle as 45°. The problem here is to devise a way for RB5X to sense its angle of tilt and respond accordingly before it tips over. To an engineer, the obvious solution would be a mercury tilt-switch (which may be ruled out by hazardous materials restrictions in some schools) or a strain gauge (which is out of the budget for most schools). Lacking those options, Fig. 4-5 presents a possible solution.

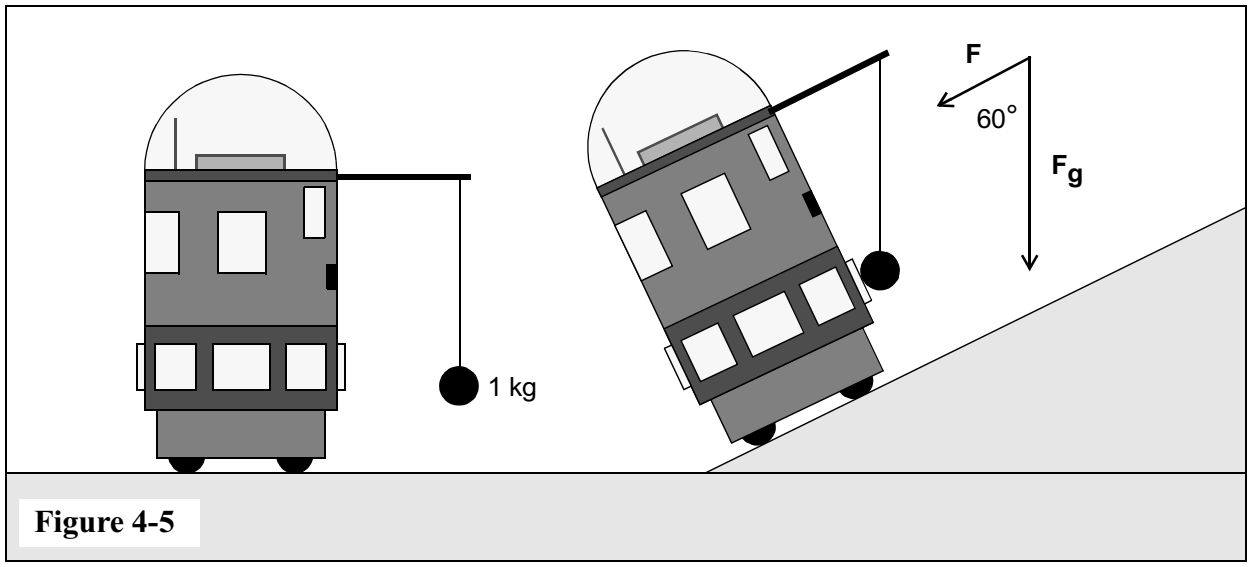


Figure 4-5

A mass suspended in front of each bumper (only one is shown) would swing back and press the bumper if RB5X's tilt exceeded some predefined value. The diagram uses an angle of 30°, which provides a 15° margin of safety. It takes about 5±1 N of force to activate one of RB5X's bumper switches, depending on state of wear. Push-button switches depend on the flexure of an elastic component to operate; over many cycles of use the force required to trigger the switch decreases. Newer switches may take up to 6 N, older ones as little as 4 N. You'll need to experiment to find what exact force is required for your robot.

By using a 1 kg mass, the weight of which is 9.8 N, we will have 5 N applied perpendicular to the bumper when the mass is hanging at the requisite angle. Using vector analysis, at a 60° angle:

$F = \cos(60°)Fg = 0.5Fg.$

This is why a 10 N weight is needed to apply 5 N of force.

Of course, there are a few obvious problems with this arrangement. Even a single such device, much less eight of them, gets a bit obtrusive. The string suspending the mass would not affect the sonar (see Mission 3, exploring further C), but it could get annoying being constantly in the view of the video camera. The boom from which the mass is suspended could be shortened, but vector analysis would then indicate a larger required mass. Alternatively, the boom could be shortened and lowered, preserving the geometry, but then the physical size of the 1 kg mass gets in the way (unless you want to spring for something very dense like, say, osmium or iridium — but we've already talked about budget limitations). One additional problem might be caused by vibration or wind inducing the mass to swing on its own, even on level ground. A simple damping mechanism could easily reduce this effect.

If you'll accept for now the suggested arrangement, the next step would be to interface this tilt-sensing device with a program that would cause RB5X to stop forward motion whenever a slope of 30° is encountered, and then back off to safer ground.

The following code is one possible solution. We assume here that the mass is configured to make contact with bumper 1.

TILT: An RCL Program to Detect and Avoid Hazardous Slopes

```
1.  PREPARE THE ROBOT
2.  CALL THIS ROBOT TASK LINE [MOVE]
3.  BEGIN A LOOP
4.  MOVE DISTANCE FORWARD FOR THIS MANY FEET [0.5]  (Note:  distance is in "meters.")
5.  WAIT FOR [3] SECONDS
5.  EXIT IF BUMPER NUMBER [1] IS TOUCHED
6.  REPEAT THIS LOOP
7.  STOP ALL MOTION
8.  MOVE DISTANCE BACKWARD FOR THIS MANY FEET [0.5]  (Note:  distance is in "meters.")
9.  SPIN LEFT 90 DEGREES
10. MOVE DISTANCE FORWARD FOR THIS MANY FEET [0.5]  (Note:  distance is in "meters.")
11. SPIN RIGHT 90 DEGREES
10. JUMP TO THE LINE CALLED [MOVE]
```

This program mimics the "incremental advance" strategy used by NASA with its robotic rovers, and gives RB5X an additional margin of safety by advancing only 0.5 m at a time. If the slope is changing so rapidly that RB5X goes from level ground to a slope of 30° in less than 0.5 m, the metal skirt at the bottom of the robot would run up against the ground.

The three second wait in line 5 allows the hanging mass time to come to rest against the bumper. You may need to increase this value depending on the type of mass used. Masses that bounce before coming to rest require a longer time to sense.

Mission 5: Navigation and Piloting

Mission Questions

A. This is an open-ended essay question. A good answer could take many forms. A discussion of the instrumentation carried by each is illuminating; this design feature was as much influenced by mission goals as by available technology — the three rovers span over 40+ years of science! The only instruments common to all were cameras and robotic arms. Summarized here in a table are the important characteristics of these three rovers.

	RB5X	**SOJOURNER**	**LUNOKHOD**
mass	10.9 kg	11.5 kg	840 kg
size (HxWxL)	58 cm x 33 cm (D)	28 cm x 48 cm x 63 cm	140 cm x 160 cm x 170 cm
power supply	battery	battery + solar	solar
power output	25 A·hr @ 46 W	∞ @ 10 W	∞ @ 200 W (estimated)
range	2 km	∞	∞
processor	NSC INS 8073	Intel 80C85	unknown
top speed	12 cm/s	0.7 cm/s	50 cm/s
control	tele- or autonomous	tele- or autonomous	teleoperation
locomotion	4 wheels, 2 driven	6 driven wheels	6 driven wheels
exploration	Earth, 2000 and beyond	Mars, 1997	Luna, 1970

B. Having a 37 cm eye-level presents several problems for any explorer. First, and most obvious, RB5X cannot see the top of anything taller than 37 cm; this is a serious handicap in the search for samples of rocks or other matter. Second, on perfectly level ground RB5X's visible horizon will be only 1.6 km away (Fig. 4-1); on uneven terrain it can be even less. Finally, dust raised by wind or RB5X's own drive mechanism will eventually obscure its vision, especially at this low height. Unfortunately, the GRC Windshield Wiper option is not yet available

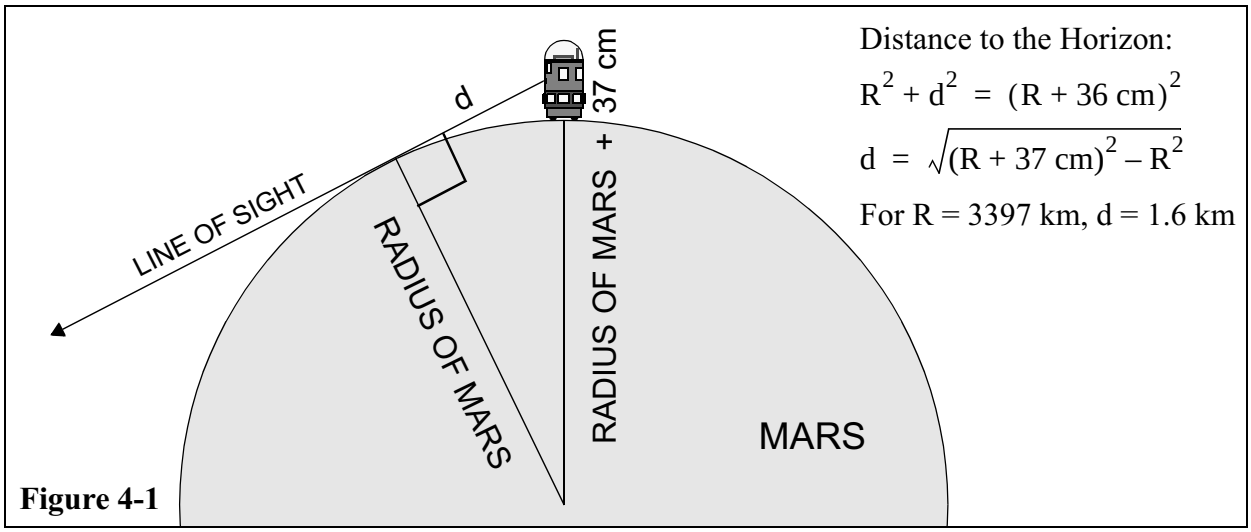


Figure 4-1

Note: It would be especially enlightening for students (and their teachers!) to observe the world through RB5X's eyes. With the RF-linked video camera option this becomes possible. Consider spending one day exploring the classroom and school area while watching RB5X's view through a monitor. Did you ever wonder what the world looks like to your dog or cat?

C. Luna 9's height is given as 1meter. Its camera filmed a reflection from a mirror mounted at the top, so its viewing height was effectively 1 meter. From this alone one would expect a more distant horizon. But Luna 9 was also on the surface of a much smaller sphere (Luna vs. Mars) and this factor will cause the horizon to appear closer. The answer can be determined by applying the same geometric analysis as in Question B. Using R = 1738 km for Luna, and a viewing height of 100 cm, the (lunar) horizon was 1.86 km distant as seen by Luna 9.

D. The easiest way to solve this problem is to plot the information on a sheet of graph paper. Lay out the sight lines with a protractor, and interpolate the intersection of these sight lines as RB5X's location (Fig. 4-2). Some error is unavoidable, but careful plotting should get you within a 1-2° radius of the approved solution: RB5X is located at coordinates 39° W, 28° N.

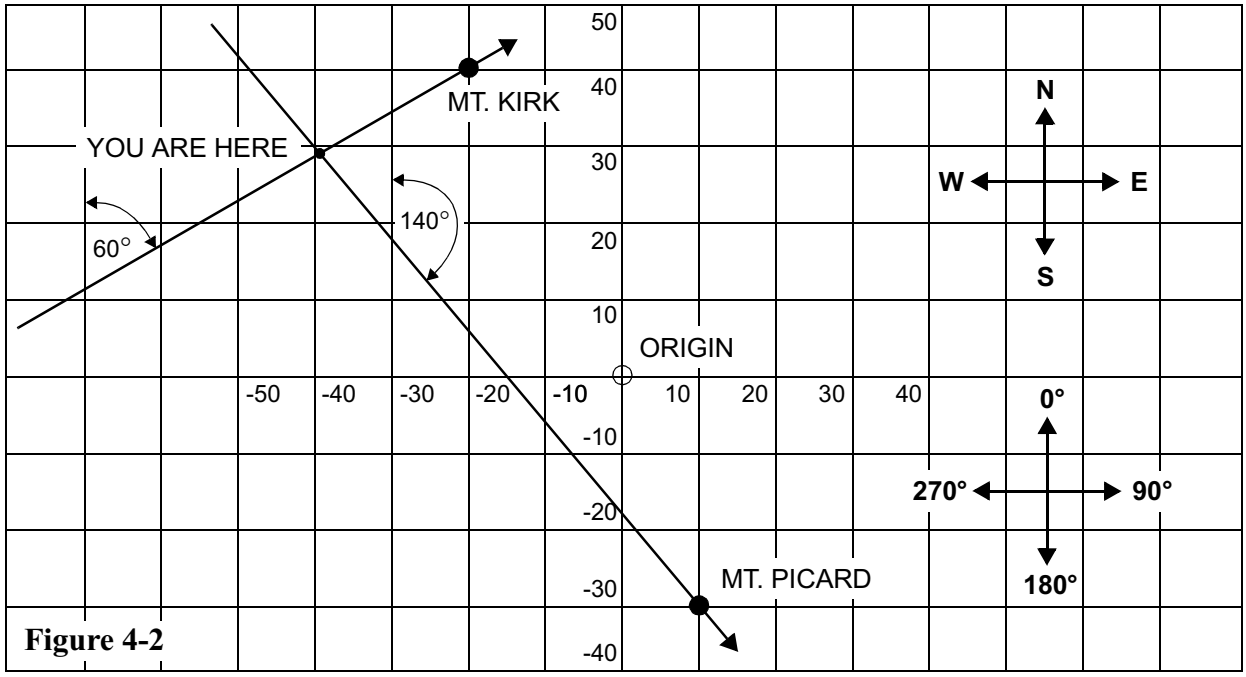


Figure 4-2

Mission Problem

You are asked to write a program that causes RB5X to do the following:

1. accept, as input, the coordinates of an arbitrarily specified grid block adjacent to the long side of the mountain
2. move from its landing site to the specified grid block
3. circumnavigate the mountain's perimeter
4. return to the landing site
5. report the length of the mountain (using horn, LED or voice)

Several methods could be used to provide the coordinate input. The simplest would be to use the coordinates of the grid block adjacent to the mountain as an ordered pair (A,B), and enter this data into the program as variable assignments in NSC Tiny Basic. If you have GRC's RF link option, sending the programs via radio avoids the need for a physical connection, and is the most elegant and realistic approach — that's how it was done with Sojourner. You could also input this data as a sequence of bumper presses representing the latitude and longitude in a specified order, but this increases the size of the program, and is less representative of the way we actually control robotic explorers. Unfortunately, there does not appear to be any method within either RCL or NSC Tiny Basic by which an executing program can write to the computer terminal and call for data as real-time inputs. Our solution uses a built-in variable assignment method.

The input coordinates are an ordered pair (LON,LAT) representing a grid block adjacent to the long side of the mountain — we will use grid block (3,4). These coordinates, assigned in lines 6 and 7, are based on the Sample Diorama; your diorama may require different values. When the robot reaches this grid block, the mountain will be circumnavigated by the program AROUND called in the last line of GOTOMT.

We assume RB5X starts on its landing site facing north, will travel in the most direct orthogonal path to the mountain, will stop on a grid block adjacent to the middle of the mountain, and will not encounter any rocks or crevices enroute (hazards will be addressed in Exploring Further A).

GOTOMT: An RCL Program to Move RB5X from its Landing Site to the Mountain

1. PREPARE THE ROBOT
2. RESERVE THE TINY-BASIC VARIABLE NAMED [A]
3. RESERVE THE TINY-BASIC VARIABLE NAMED [B]
4. RESERVE THE TINY-BASIC VARIABLE NAMED [C]
5. RESERVE THE TINY-BASIC VARIABLE NAMED [E]
6. SET THE TINY-BASIC VARIABLE [A] EQUAL TO [3]
7. SET THE TINY-BASIC VARIABLE [B] EQUAL TO [4]
8. COMPILE THE BASIC STATEMENT [IF A>6 GOTO 39] WHICH MEANS [GO EAST]
9. COMPILE THE BASIC STATEMENT [IF A<6 GOTO 51] WHICH MEANS [GO WEST]
10. CALL THIS ROBOT TASK LINE [GO NORTH]
11. COMPILE THE BASIC STATEMENT [C=B] WHICH MEANS [CALC GB]
12. COMPILE THE BASIC STATEMENT [GOSUB 63] WHICH MEANS [MOVE]
13. JUMP TO THE LINE CALLED [THERE]
14. CALL THIS ROBOT TASK LINE [GO EAST]
15. SPIN RIGHT 90 DEGREES
16. COMPILE THE BASIC STATEMENT [C=A-6] WHICH MEANS [CALC GB]
17. COMPILE THE BASIC STATEMENT [GOSUB 63] WHICH MEANS [MOVE]
18. SPIN LEFT 90 DEGREES
19. JUMP TO THE LINE CALLED [GO NORTH]
20. CALL THIS ROBOT TASK LINE [GO WEST]
21. SPIN LEFT 90 DEGREES
22. COMPILE THE BASIC STATEMENT [C=6-A] WHICH MEANS [CALC GB]
23. COMPILE THE BASIC STATEMENT [GOSUB 63] WHICH MEANS [MOVE]
24. SPIN RIGHT 90 DEGREES
25. JUMP TO THE LINE CALLED [GO NORTH]
26. CALL THIS ROBOT TASK LINE [MOVE]
27. BEGIN A COUNTED LOOP CALLED [E] BEGINNING AT [1] ENDING AT [C]

```
28.    WAIT FOR [1] SECONDS
29.    MOVE DISTANCE FORWARD THIS MANY FEET [0.5]  (Note:  distance is in “meters.”)
30.    WAIT FOR [1] SECONDS
31.  REPEAT THE COUNTED LOOP FOR THE COUNTER [E]
32.  COMPILE THE BASIC STATEMENT [RETURN] WHICH MEANS [RETURN]
33.  CALL THIS ROBOT TASK LINE [THERE]
34.  AROUND
```

At this point RB5X is left adjacent to the mountain, facing north, at coordinates (A,B). The next step is for RB5X to explore the perimeter of the mountain, measuring its length along the way.

AROUND: An RCL Program to Circumnavigate the Mountain

```
1.   PREPARE THE ROBOT
2.   RESERVE THE TINY-BASIC VARIABLE NAMED [F]
3.   RESERVE THE TINY-BASIC VARIABLE NAMED [G]
4.   SET THE TINY-BASIC VARIABLE [F] EQUAL TO [A]
5.   BEGIN A LOOP
6.     EXIT IF SONAR DISTANCE IS [LESS] THAN [0.2] FEET  (Note:  distance is in “meters.”)
7.   REPEAT THIS LOOP
8.   CALL THIS ROBOT TASK LINE [WEST]
9.   BEGIN A LOOP
10.    SPIN LEFT 90 DEGREES
11.    MOVE DISTANCE FORWARD THIS MANY FEET [0.5]  (Note:  distance is in “meters.”)
12.    COMPILE THE BASIC STATEMENT [F=F-1] WHICH MEANS [DECREMENT LON]
13.    COMPILE THE BASIC STATEMENT [IF A=F GOTO 100] WHICH MEANS [DONE]
14.    SPIN RIGHT 90 DEGREES
15.    COMPILE THE BASIC STATEMENT [LINK#1100:DELAY20] WHICH MEANS [GET D]
16.    COMPILE THE BASIC STATEMENT [E=(30*D - 30*73)/22] WHICH MEANS [CALC CM]
17.    COMPILE THE BASIC STATEMENT [IF E>20 GOTO 127] WHICH MEANS [NO MT]
18.  REPEAT THIS LOOP
19.  CALL THIS ROBOT TASK LINE [NORTH]
20.  COMPILE THE BASIC STATEMENT [G=F] WHICH MEANS [SAVE MIN LON]
21.  MOVE DISTANCE FORWARD THIS MANY FEET [1.5]  (Note:  distance is in “meters.”)
22.  SPIN CLOCKWISE FOR [180] DEGREES
23.  CALL THIS ROBOT TASK LINE [EAST]
24.  BEGIN A LOOP
25.    SPIN LEFT 90 DEGREES
26.    MOVE DISTANCE FORWARD THIS MANY FEET [0.5]  (Note:  distance is in “meters.”)
27.    COMPILE THE BASIC STATEMENT [F=F+1] WHICH MEANS [INCREMENT LON]
28.    SPIN RIGHT 90 DEGREES
29.    COMPILE THE BASIC STATEMENT [LINK#1100:DELAY20] WHICH MEANS [GET D]
30.    COMPILE THE BASIC STATEMENT [E=(30*D - 30*73)/22] WHICH MEANS [CALC CM]
31.    COMPILE THE BASIC STATEMENT [IF E>20 GOTO 136] WHICH MEANS [NO MT]
32.  REPEAT THIS LOOP
33.  CALL THIS ROBOT TASK LINE [SOUTH]
34.  COMPILE THE BASIC STATEMENT [G= F-G] WHICH MEANS [CALC MT]
35.  MOVE DISTANCE FORWARD THIS MANY FEET [1.5]  (Note:  distance is in “meters.”)
36.  SPIN CLOCKWISE FOR [180] DEGREES
37.  JUMP TO THE LINE CALLED [WEST]
38.  CALL THIS ROBOT TASK LINE [DONE]
39.  SPIN LEFT 90 DEGREES
40.  HOMER
```

AROUND breaks the robot's motion into four parts, one for each compass direction. When the robot returns to its starting grid block it will rotate to face away from the mountain, at which point the program HOMER (called in the last line of AROUND) takes RB5X back to its landing site.

Note that the length of the mountain (measured in grid blocks) is currently stored as the value of the variable G. The starting coordinates for the return trip (A,B) are also still in memory.

HOMER: An RCL Program to Return RB5X to the Landing Site

```
1,  PREPARE THE ROBOT
2.  CALL THIS ROBOT TASK LINE [HEAD SOUTH]
3.  COMPILE THE BASIC STATEMENT [C=B] WHICH MEANS [CALC GB]
4.  COMPILE THE BASIC STATEMENT [GOSUB 172] WHICH MEANS [MOVER]
5.  COMPILE THE BASIC STATEMENT [IF A<6 GOTO 148] WHICH MEANS [HEAD EAST]
6.  COMPILE THE BASIC STATEMENT [IF A>6 GOTO 160] WHICH MEANS [HEAD WEST]
7.  JUMP TO THE LINE CALLED [HOME]
8.  CALL THIS ROBOT TASK LINE [HEAD EAST]
9.  SPIN LEFT 90 DEGREES
10. COMPILE THE BASIC STATEMENT [C=6-A] WHICH MEANS [CALC GB]
11. COMPILE THE BASIC STATEMENT [GOSUB 172] WHICH MEANS [MOVER]
12. SPIN LEFT 90 DEGREES
13. JUMP TO THE LINE CALLED [HOME]
14. CALL THIS ROBOT TASK LINE [HEAD WEST]
15. SPIN RIGHT 90 DEGREES
16. COMPILE THE BASIC STATEMENT [C=A-6] WHICH MEANS [CALC GB]
17. COMPILE THE BASIC STATEMENT [GOSUB 172] WHICH MEANS [MOVER]
18. SPIN RIGHT 90 DEGREES
19. JUMP TO THE LINE CALLED [HOME]
20. CALL THIS ROBOT TASK LINE [MOVER]
21. BEGIN A COUNTED LOOP CALLED [E] BEGINNING AT [1] ENDING AT [C]
22.   WAIT FOR [1] SECONDS
23.   MOVE DISTANCE FORWARD THIS MANY FEET [0.5]  (Note:  distance is in "meters.")
24.   WAIT FOR [1] SECONDS
25. REPEAT THE COUNTED LOOP FOR THE COUNTER [E]
26. COMPILE THE BASIC STATEMENT [RETURN] WHICH MEANS [RETURN]
27. CALL THIS ROBOT TASK LINE [HOME]
28. REPORT
```

RB5X is now back at its landing site, facing north. The program REPORT called in the last line of HOMER will signal the length of the mountain as a series of honks on its horn. Each honk will represent one grid block of distance.

REPORT: An RCL Program to Report the Length of the Mountain

```
1,  PREPARE THE ROBOT
2.  BEGIN A COUNTED LOOP CALLED [E] BEGINNING AT [1] ENDING AT [G]
3.  WAIT FOR [1] SECONDS
4.  HONK THE HORN FOR [1] SECONDS
5.  REPEAT THE COUNTED LOOP FOR THE COUNTER [E]
```

Other more elegant methods of reporting the mountain's length are possible if your RB5X has the voice/sound synthesis option. We present here a brief summary of methods needed to convert a numerical value in the robot's memory to spoken digits.

To extract the individual digits of an integer we can perform a series of successive divisions and store each digit as a separate variable. For example, assume the value to be reported is "123" and that it is stored as the variable D. Since the results of all calculations are truncated to integers, the individual digits (1, 2 and 3) can be extracted as follows.

A=(D/100) stores hundreds digit in A
B=(D-100A)/10 stores tens digit in B
C=(D-100A)-10B stores ones digit in C

Next, we need a subroutine that speaks each digit, one at a time, by invoking its corresponding phrase. If the variables A, B and C are successively loaded into the variable U, the subroutine could proceed as follows.

```
1.   COMPILE THE BASIC STATEMENT [IF U = 0 GOTO 11] WHICH MEANS [SAY 0]
2.   COMPILE THE BASIC STATEMENT [IF U = 1 GOTO 13] WHICH MEANS [SAY 1]
3.   COMPILE THE BASIC STATEMENT [IF U = 2 GOTO 15] WHICH MEANS [SAY 2]
     ...
11.  SPEAK THE PHRASE CALLED:  [ZERO]
12.  JUMP TO THE LINE CALLED [DONE]
13.  SPEAK THE PHRASE CALLED:  [ONE]
14.  JUMP TO THE LINE CALLED [DONE]
15.  SPEAK THE PHRASE CALLED:  [TWO]
16.  JUMP TO THE LINE CALLED [DONE]
     ...
```

Of course, these phrases must be created at the start of the program using the MAKE A PHRASE command as explained in the Speech/Sound Synthesis Manual. Here are some of the phonemes you'll need to make the phrases needed for reporting numerical values:

0: Z.AY.I1.R.O1.U1
1: W.UH1.UH2.N
2: T.IU.U1.U1
3. TH.R.E1.Y
4: F.O1.O2.R
5: F.AH1.EH3.Y.V
6: S.I1.I3.K.PA0.S
7: S.EH1.EH3.V.I2.N
8: A2.A2.Y.T
9: N.AH1.EH3.Y.N

point: P.O1.UH3.I3.AY.N.T
range: R.A1.AY.Y.N.D.J.PA0
meters: M.E1.Y.T.ER.S
distance: D.I1.S.T.EH3.N.T.S
length: L.EH1.EH3.NG.TH

Exploring Further

A. Placed on a remote grid block (X,Y) of the diorama, given its starting coordinates, and facing south, RB5X is asked to "find its way home" while avoiding obstacles along the way. Since the locations of the mountain, rocks and crevice are unknown, obstacle detection loops (using sonar in line 3, bumper 1 in line 10, and IR in line 13) are required between motion loops. Due to RCL program size limits, the obstacle detection loop will be a separate program (PROBE) called in line 8 of the main program (FINDHOME) and inserted as a subroutine therein.

PROBE: An RCL Subroutine for Detecting Mountain, Rock or Crevice

```
1.  CALL THIS ROBOT TASK LINE [MT PROBE]
2.  SET THE TINY-BASIC VARIABLE [G] EQUAL TO [0]  (flag value 0 = no obstacle, 1 = obstacle)
3.  COMPILE THE BASIC STATEMENT [LINK#1100:DELAY20] WHICH MEANS [GET D]
4.  COMPILE THE BASIC STATEMENT [E=(30*D - 30*73)/22] WHICH MEANS [CALC CM]
5.  COMPILE THE BASIC STATEMENT [IF E<10 GOTO 48] WHICH MEANS [MT]
6.  CALL THIS ROBOT TASK LINE [RK&CR PROBE]
7.  BEGIN A COUNTED LOOP CALLED [C] BEGINNING AT [1] ENDING AT [5]
8.    MOVE DISTANCE FORWARD THIS MANY FEET [0.1]  (Note:  distance is in "meters.")
9.    WAIT FOR [2] SECONDS
10.   COMPILE THE BASIC STATEMENT [Y=@#7800:IF Y=251 GOTO 55] WHICH MEANS [RK]
11.   COMPILE THE BASIC STATEMENT [X=#02] WHICH MEANS [TURN ON IR]
12.   COMPILE THE BASIC STATEMENT [@#7801=@#7801 OR X] WHICH MEANS [SET BIT]
13.   COMPILE THE BASIC STATEMENT [IF (@#7802 AND #40)=64 GOTO 55] WHICH MEANS [CR]
14. REPEAT THE COUNTED LOOP FOR THE COUNTER [C]
15. JUMP TO THE LINE CALLED [RETURN]
16. CALL THIS ROBOT TASK LINE [RETREAT]
17. BEGIN A COUNTED LOOP CALLED [F] BEGINNING AT [1] ENDING AT [C]
18.   MOVE DISTANCE BACKWARD THIS MANY FEET [0.1]  (Note:  distance is in "meters.")
19.   WAIT FOR [2] SECONDS
20. REPEAT THE COUNTED LOOP FOR THE COUNTER [F]
21. SET THE TINY-BASIC VARIABLE [G] EQUAL TO [1]
22. CALL THIS ROBOT TASK LINE [RETURN]
23. COMPILE THE BASIC STATEMENT [RETURN] WHICH MEANS [RETURN]
```

Note that PROBE, when executed, will leave the robot advanced by one grid block if it is safe to do so, and return a flag value of G=0. If mountain, rock or crevice are detected, the robot remains on (or returns to) the grid block from which it started; in this case a value of G=1 is returned.

FINDHOME: An RCL Program Demonstrating AI Roving

```
1.  PREPARE THE ROBOT
2.  RESERVE THE TINY-BASIC VARIABLE NAMED [A]
3.  RESERVE THE TINY-BASIC VARIABLE NAMED [B]
4.  RESERVE THE TINY-BASIC VARIABLE NAMED [C]
5.  RESERVE THE TINY-BASIC VARIABLE NAMED [E]
6.  RESERVE THE TINY-BASIC VARIABLE NAMED [F]
7.  RESERVE THE TINY-BASIC VARIABLE NAMED [G]
8.  PROBE
9.  SET THE TINY-BASIC VARIABLE [A] EQUAL TO [X]  (provide the actual longitude here)
10. SET THE TINY-BASIC VARIABLE [B] EQUAL TO [Y]  (provide the actual latitude here)
```

```
11. BEGIN A LOOP
12.   EXIT IF SONAR DISTANCE IS [MORE] THAN [0] FEET  (Note:  distance is in "meters.")
13. REPEAT THIS LOOP
14. CALL THIS ROBOT TASK LINE [GO LAT]
15. BEGIN A LOOP
16.   COMPILE THE BASIC STATEMENT [GOSUB 31] WHICH MEANS [PROBE]
17.   COMPILE THE BASIC STATEMENT [IF G=1 GOTO 73] WHICH MEANS [GO LON]
18.   SET THE TINY-BASIC VARIABLE [B] EQUAL TO [B-1]
19.   COMPILE THE BASIC STATEMENT [IF B=0 GOTO 73] WHICH MEANS [GO LON]
20. REPEAT THIS LOOP
21. CALL THIS ROBOT TASK LINE [GO LON]
22. COMPILE THE BASIC STATEMENT [IF (A=6) AND (B=0) GOTO 112] WHICH MEANS [HOME]
23. COMPILE THE BASIC STATEMENT [IF A<6 GOTO 77] WHICH MEANS [GO EAST]
24. COMPILE THE BASIC STATEMENT [IF A>6 GOTO 94] WHICH MEANS [GO WEST]
25. CALL THIS ROBOT TASK LINE [GO EAST]
26. SPIN LEFT 90 DEGREES
27. BEGIN A LOOP
28.   COMPILE THE BASIC STATEMENT [GOSUB 31] WHICH MEANS [PROBE]
29.   COMPILE THE BASIC STATEMENT [IF G=0 GOTO 90] WHICH MEANS [ADVANCE]
30.   SPIN RIGHT 90 DEGREES
31.   JUMP TO THE LINE CALLED [GO LAT]
32.   SET THE TINY-BASIC VARIABLE [A] EQUAL TO [A+1]
33.   COMPILE THE BASIC STATEMENT [IF (A=6) AND (B=0) GOTO 117] WHICH MEANS [HOME]
34. REPEAT THIS LOOP
35. CALL THIS ROBOT TASK LINE [GO WEST]
36. SPIN RIGHT 90 DEGREES
37. BEGIN A LOOP
38.   COMPILE THE BASIC STATEMENT [GOSUB 31] WHICH MEANS [PROBE]
39.   COMPILE THE BASIC STATEMENT [IF G=0 GOTO 107] WHICH MEANS [ADVANCE]
40.   SPIN LEFT 90 DEGREES
41.   JUMP TO THE LINE CALLED [GO LAT]
42.   SET THE TINY-BASIC VARIABLE [A] EQUAL TO [A-1]
43.   COMPILE THE BASIC STATEMENT [IF (A=6) AND (B=0) GOTO 122] WHICH MEANS [HOME]
44. REPEAT THIS LOOP
45. CALL THIS ROBOT TASK LINE [HOME]
46. SPIN CLOCKWISE FOR [180] DEGREES
47. JUMP TO THE LINE CALLED [DONE]
48. SPIN LEFT 90 DEGREES
49. JUMP TO THE LINE CALLED [DONE]
50. SPIN RIGHT 90 DEGREES
51. CALL THIS ROBOT TASK LINE [DONE]
```

Having survived all manner of obstacles and traversed a long distance, RB5X is now safely back at the landing site, facing north, and ready for its next challenge.

B. The teleoperator is not significantly burdened by the lack of video. Simple audible signals can be used to convey an accurate "picture" of what RB5X encounters. For our example solution:

1 horn beep ⇒ all clear, forward movement by one grid block was successful
2 horn beeps ⇒ the mountain is on the grid block directly in front of the robot
3 horn beeps ⇒ a rock is on the grid block directly in front of the robot
4 horn beeps ⇒ the crevice is on the grid block directly in front of the robot

By using a map of the diorama, and annotating it according to horn signals, an accurate "image" of the area around the landing site can be ascertained. This will also facilitate RB5X's return to the landing site after the rock is found.

Unfortunately, the RF link does not allow the type of "real time" operation enjoyed by model car and airplane enthusiasts. It does, however, allow for the uploading of programs that self-execute without the need for a bumper press. To enable this feature, use the calibration process to set the BUMPER NEEDED parameter to "NO."

The "incremental advance" strategy used by NASA makes excellent sense under these conditions. RB5X will be programmed to move only a single grid block at a time, report whether the move was successful or aborted, and then await instructions for its next move. The strategy, then, is to upload a new program for each move. These programs will recycle the obstacle detection loops provided in the RCL program PROBE (Exploring Further A), and will be identical to each other except for line 11 as shown below. Of course, the pilot also has the option to upload an entirely new program consisting of movement commands only; this could be done if corrective repositioning or back-tracking is required.

The variable A will hold a value indicating the type of obstacle encountered, in accordance with the horn scheme outlined earlier. It is thus initialized at 1 indicating "all clear."

EXPLORE: An RCL Program for Teleoperation

```
1.  PREPARE THE ROBOT
2.  RESERVE THE TINY-BASIC VARIABLE NAMED [A]
3.  RESERVE THE TINY-BASIC VARIABLE NAMED [B]
4.  RESERVE THE TINY-BASIC VARIABLE NAMED [C]
5.  RESERVE THE TINY-BASIC VARIABLE NAMED [E]
6.  BEGIN A LOOP
7.    EXIT IF SONAR DISTANCE IS [MORE] THAN [0] FEET  (Note:  distance is in "meters.")
8.  REPEAT THIS LOOP
9.  SET THE TINY-BASIC VARIABLE [A] EQUAL TO [1]
10. CALL THIS ROBOT TASK LINE [ROTATE]
11. SPIN CLOCKWISE FOR [X] DEGREES  (Note:  X to be decided by the pilot)
12. CALL THIS ROBOT TASK LINE [MT PROBE]
13. COMPILE THE BASIC STATEMENT [LINK#1100:DELAY20] WHICH MEANS [GET D]
14. COMPILE THE BASIC STATEMENT [B=(30*D - 30*73)/22] WHICH MEANS [CALC CM]
15. COMPILE THE BASIC STATEMENT [IF B<10 GOTO 46] WHICH MEANS [MT]
16. JUMP TO THE LINE CALLED [RK&CR PROBE]
17. SET THE TINY-BASIC VARIABLE [A] EQUAL TO [2]
18. JUMP TO THE LINE CALLED [REPORT]
19. CALL THIS ROBOT TASK LINE [RK&CR PROBE]
20. BEGIN A COUNTED LOOP CALLED [C] BEGINNING AT [1] ENDING AT [5]
21.   MOVE DISTANCE FORWARD THIS MANY FEET [0.1]  (Note:  distance is in "meters.")
22.   WAIT FOR [2] SECONDS
23.   COMPILE THE BASIC STATEMENT [Y=@#7800:IF Y=251 GOTO 63] WHICH MEANS [RK]
24.   COMPILE THE BASIC STATEMENT [X=#02] WHICH MEANS [TURN ON IR]
25.   COMPILE THE BASIC STATEMENT [@#7801=@#7801 OR X] WHICH MEANS [SET BIT]
26.   COMPILE THE BASIC STATEMENT [IF (@#7802 AND #40)=64 GOTO 65] WHICH MEANS [CR]
27. REPEAT THE COUNTED LOOP FOR THE COUNTER [C]
```

```
28.  HONK THE HORN FOR [1] SECONDS
29.  JUMP TO THE LINE CALLED [DONE]
30.  SET THE TINY-BASIC VARIABLE [A] EQUAL TO [3]
31.  JUMP TO THE LINE CALLED [RETREAT]
32.  SET THE TINY-BASIC VARIABLE [A] EQUAL TO [4]
33.  CALL THIS ROBOT TASK LINE [RETREAT]
34.  BEGIN A COUNTED LOOP CALLED [E] BEGINNING AT [1] ENDING AT [C]
35.    MOVE DISTANCE BACKWARD THIS MANY FEET [0.1]  (Note:  distance is in "meters.")
36.    WAIT FOR [2] SECONDS
37.  REPEAT THE COUNTED LOOP FOR THE COUNTER [E]
38.  CALL THIS ROBOT TASK LINE [REPORT]
39.  BEGIN A COUNTED LOOP CALLED E] BEGINNING AT [1] ENDING AT [A]
40.    HONK THE HORN FOR [1] SECONDS
41.    WAIT FOR [1] SECONDS
42.  REPEAT THE COUNTED LOOP FOR THE COUNTER [E]
43.  CALL THIS ROBOT TASK LINE [DONE]
```

The pilot can now plan the next move based on the reported results of EXPLORE. This usually will involve only a change to line 11, as noted above.

If you have the voice/sound synthesis option, the REPORT sequence (lines 38-42) can be replaced by commands that verbally identify the obstacles. Use these phonemes:

all clear: AW.L.PA0.K.L.AY.I3.R
mountain: M.AH1.UH3.W.N.T.I1.I3.N
rock: R.AH1.UH3.K
crevice: K.R.EH1.EH3.V.I1.I3.S

C. The basic process here is similar to Exploring Further B, with the important difference being the teleoperator can see what's going on at all times. Of course, what is seen by the pilot is what is seen by the camera. We would simply turn our head, but RB5X needs to rotate its entire body to survey the area for its next move — the camera's horizontal field of view is only 60°. Further, the vertical field of view is only 50°, and the camera is mounted 37 cm above ground level, so a point on the ground closer than about 0.75 meter is out of sight.

In practice, the first program would be a short sequence of commands that rotates the robot and allows the pilot to survey the area and formulate a movement strategy. The specific strategy will be unique to each diorama and pilot but will need to address the following issues:

- pilot depth perception (distance estimates for programmed motions)
- energy conservation (avoid wasted motions)
- obstacle avoidance (mountain and crevice limit motions)
- diagonal vs. orthogonal motion (program command choices)

Once found, the actual movement to the rock, however orchestrated, will most likely proceed as a series of 2-5 programs, each correcting for the movement errors of its predecessor. Verification of physical contact between bumper 1 and the rock will be signalled by the corresponding red LED located at the front of the LED board.

Mission 6: Energy Management

Mission Questions

A. The 10 amp-hour rating means the battery could supply 10 amps of current for 1 hour, 1 amp of current for 10 hours, or any other combination such that the following equation is met:

$$10 \text{ amp}\cdot\text{hr} = I\cdot t$$

where I = current in units of amperes, and t = time in units of hours. This equation is really just a simplification of the more general

$$E = P\cdot t = I\cdot\Delta V\cdot t$$

where E = energy in units of joules, P = power output in units of watts, V = voltage in units of volts (6.6 volts for RB5X's battery when fully charged), and t = time in units of seconds.

To predict how long and how far RB5X could travel we'd need to know the current (I) drawn by its drive motors. For a fully charged battery, with the robot traveling on smooth level ground, this value measures approximately 2.0 amperes. Using the first equation, we get

time of operation: $t = 10 \text{ amp}\cdot\text{hr}/I = \underline{5.0 \text{ hours}}$
range: $d = v\cdot t$ (with $v \approx 0.42$ km/hr) = $\underline{2.1 \text{ km}}$

This answer is in close agreement with specifications given in the Reference Manual, where the average operational time of a fully charged RB5X is cited as 4-8 hours. Since "operational time" usually includes non-moving activities, and since the motors are the largest load on the batteries, we would expect an answer on the low end of that 4-8 hour range for continuous motion.

Note: the 2.1 km answer makes several simplifying assumptions about the behavior of batteries and motors. The output voltage and motor speed are not constant — they decrease slowly as the battery discharges. For a further discussion of the range issue, see the Mission Problem solution.

B. RB5X uses two 6.6 volt batteries. The drive motor battery is fused at 6.0 amperes; the electronics battery is fused at 1.0 ampere. Assuming the robot is fully charged, and is working at its maximum power output, the total power available would be:

$$P = I\cdot\Delta V = (1.0 \text{ A} + 6.0 \text{ A})\cdot 6.6\text{V} = 46 \text{ A}\cdot\text{V} = 46 \text{ W (watts)}$$

Sojourner was rated at 10 W, so RB5X is 4.6 times as powerful as Sojourner. RB5X rules!

C. The maximum slope RB5X could climb would depend on three external factors:

1. rolling friction between its wheels and the surface
2. the strength of gravity
3. ambient winds (head or tail)

D. The Reference Manual cites the weight of RB5X (on Earth) as 24 lb (107 N). In this Manual's Appendix 4 (Data for the Planets) the gravity on Mars is given as 3.69 N/kg compared to Earth's 9.81 N/kg. A simple ratio can be used, without conversion from pounds to newtons, i.e.,

$$\text{weight on Mars} = \frac{3.69\ \text{N/kg}}{9.81\ \text{N/kg}} \times 24\ \text{lb} = 0.376 \times 24\ \text{lb} = 9.02\ \text{lb}\ (= 40.1\ \text{N})$$

This has a lot to do with energy management, since with only 1/3 its normal weight there will be 1/3 the rolling friction and 1/3 the energy expended when climbing hills.

Mission Problem

The solution to Mission Question A provides a rough estimate of RB5X's range, but assumes the drive motor speed remains constant. In reality, as the battery is discharged, both voltage output and motor speed decrease (nearly) linearly. Also, when the output voltage drops to 5.4 volts, a self-protection circuit shuts the robot down rendering the remaining energy unavailable.

To accurately predict RB5X's range we need to measure the rate of change in voltage and speed ($\Delta V/\Delta t$ and $\Delta v/\Delta t$, respectively). The time of operation (t_f) can then be found by extrapolating V to 5.4 volts. RB5X's range can now be calculated as the integral of speed over time during the interval t = 0 to t_f. Fig. 6-1 illustrates this process graphically. For a linear $\Delta v/\Delta t$, the result of this integration is the same as the area of the shaded triangle, i.e.,

$$d = \tfrac{1}{2}(\text{base})(\text{height}) = \tfrac{1}{2}(t_f)(0.42\ \text{km/hr} - v_f).$$

To obtain $\Delta V/\Delta t$ and $\Delta v/\Delta t$ we need to measure V and v at two levels of battery charge (e.g., full and half charge), and the time it takes to go from full charge to half charge. Full charge values are given (Fig. 6-1). Half charge values must be measured. With this data, $\Delta V/\Delta t$ and $\Delta v/\Delta t$ can be calculated. Finally, plot the graph and calculate the range (d).

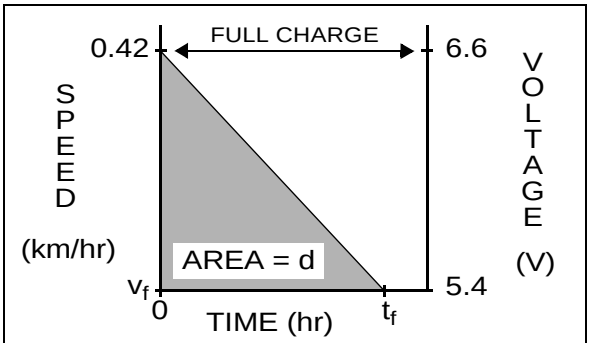


Figure 6-1

Another, more elegant, approach is to use analytic geometry and solve directly for the range:

$$d = \frac{0.72\left(\frac{\Delta v}{\Delta t}\right)}{\left(\frac{\Delta V}{\Delta t}\right)^2} + \frac{0.50}{\frac{\Delta V}{\Delta t}}\ (\text{km})$$

Measured values yield $\Delta v/\Delta t = 0.12$ km/hr/hr and $\Delta V/\Delta t = 0.42$ V/hr. So $t_f = 2.9$ hr and d = 1.8 km.

Note: the LEDs on the front of the robot only give an approximate measure of the battery's charge (Fig. 6-2). Use a voltmeter directly attached to the main battery for more accurate data.

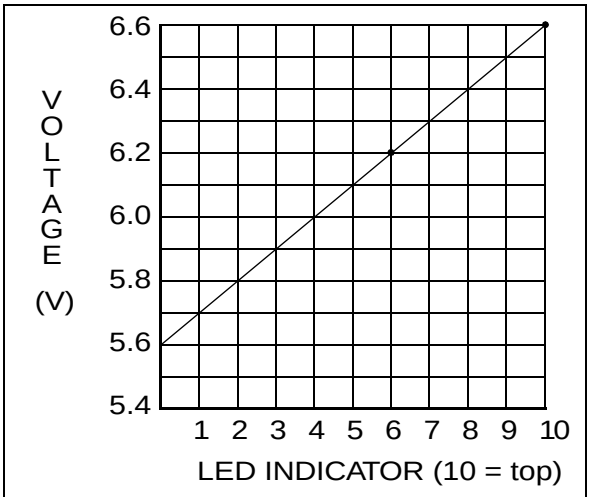


Figure 6-2

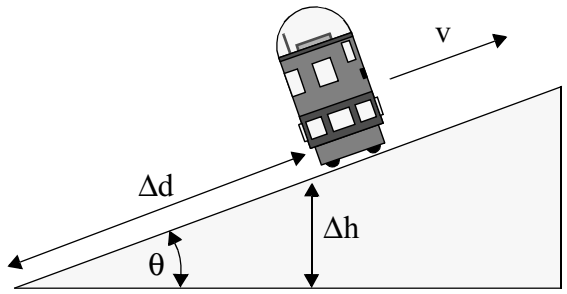
Exploring Further

A. The internal factors determining the maximum slope RB5X can climb are the 6 ampere limit (via fusing) on its drive motor circuit, and the internal and rolling friction on its drive system.

Power expended against friction at constant speed on level ground can be found from

$$P = F_f \cdot v$$

Power is also defined as $\Delta E/\Delta t$. In the case of climbing, it's the gravitational potential energy that is changing. Power expended climbing a slope of angle θ is given by Eq. 5 in Fig. 6-3.



RATE OF CHANGE IN POTENTIAL ENERGY

1. $P = \Delta Ep/\Delta t = \Delta(mgh)/\Delta t = mg(\Delta h/\Delta t)$
2. $\sin(\theta) = \Delta h/\Delta d \Rightarrow \Delta h = \Delta d \cdot \sin(\theta)$
3. $\Delta d = v \cdot \Delta t$

Substituting Eq. 3 into Eq. 2 ...

4. $\Delta h = [v \cdot \Delta t] \cdot [\sin(\theta)] \Rightarrow \Delta h/\Delta t = v \cdot \sin(\theta)$

Substituting Eq. 4 into Eq. 1 ...

5. $P = \Delta Ep/\Delta t = mgv \cdot \sin(\theta)$

Figure 6-3

So the total power output required to overcome friction and climb a slope is

$$P = F_f \cdot v + mgv \cdot \sin(\theta)$$

Solving the above for θ yields an equation for maximum slope:

$$\theta = \sin^{-1}\left(\frac{P - F_f \cdot v}{mgv}\right)$$

Thus, to predict RB5X's maximum climbing slope we need to know the following data:

$P = I \cdot \Delta V = (6.0\text{ A})(6.6\text{ V}) = 40\text{ W}$ [Read on to find out why this value is too large.]
$F_f = 105\text{ N}$ [Exploring Further B explains how this value can be found.]
$v = 0.12\text{ m/s}$
$m = 11\text{ kg}$
$g = 9.8\text{ N/kg}$

Substituting these values into the equation for maximum slope produces: θ = undefined (!?)

This undefined value for θ tells us there is no "maximum slope." RB5X could lift its own weight vertically at the given speed of 0.12 m/s (if configured to do so with a simple pulley) — and have power to spare! This is a result of the extremely low gear ratio used in its drive system. If told to climb an increasingly steep slope, RB5X would topple off balance before it would stall out.

The chart below shows actual measurements of RB5X's performance on a series of four smooth, hard, and increasingly steep slopes. The speed is remarkably constant, and the voltage drop of the battery as current increases is almost negligible, indicating sufficient power for locomotion under all but the most extreme conditions. Current was measured by connecting an amp meter between the drive motor and main battery. Voltage was measured by connecting a volt meter in parallel with the battery. All measurements were made with the battery fully charged.

slope (deg)	I (amps)	V (volts)	v (m/s)
0	1.95	6.2	0.115
5.2	2.15	6.1	0.113
8.7	2.35	6.0	0.110
11.7	2.50	5.9	0.105

The reason for the voltage drop is the battery's internal resistance. Electricity is not "created" by a battery — the battery acts like more a pump, causing electricity to flow around the entire circuit, battery included. As electricity flows through the battery it experiences this internal resistance, giving up some energy and generating heat. The higher the current, the greater the voltage drop. The greater the battery's capacity (as measured in amp-hours), the smaller the voltage drop. <u>All</u> batteries in all applications will exhibit this effect. However, if the battery is properly sized for its application, the voltage drop will be within design parameters and cause no operational problems.

This also explains why the 40W power output cited earlier is too large. Under load, the battery's output is less than 6.6 V. Further, as the current approaches the fuse-limited value of 6.0 A, the voltage drop would trigger a system shutdown at 5.4 V. Extrapolating values in the above chart, this shutdown will occur when the current reaches around 3.5 A. Just before shutdown, the power output will have been reduced to a mere 19W (3.5 A x 5.4 V), and the speed to 0.09 m/s. Using these values to recalculate the maximum climbing slope yields $\theta \approx 80$ degrees! Of course, RB5X would not be stable on that steep a slope (see Mission 4, Mission Question C).

B. If we try to measure the friction directly, say by pulling RB5X along the ground with a rope attached to a spring scale, the drive wheels will slide on most surfaces due to the low gearing. Rather than disengaging the wheels from the drive gears, which is a fairly complicated operation, we can obtain an indirect measurement employing the following analysis.

The friction experienced at a constant speed can be calculated from the power output required to maintain that speed. The equation below shows how this can be done. Using data from the chart in Exploring Further A (1g on level ground) we get:

$$F_f \cdot v = P = I \cdot \Delta V \Rightarrow F_f = \frac{I \cdot \Delta V}{v} = \frac{(1.95\ \text{A})(6.2\ \text{V})}{0.115\ \text{m/s}} = 105\ \text{N}$$

2g conditions can be simulated by loading RB5X with an additional 11 kg of mass. 25 pounds (11.4 kg) of barbell weights strapped evenly around the robot is close enough. Measuring again on level ground, we obtain:

$$F_f = \frac{I \cdot \Delta V}{v} = \frac{(2.10\ A)(6.1\ V)}{0.111\ m/s} = 115\ N$$

This is a difference of only 10 N in friction. Rolling friction is proportional to weight, so under 2g conditions it should have doubled. This tells us that only a small part of the calculated 105 N is due to rolling friction, and that the majority is internal (drive gear friction and electrical resistance losses). Assuming these internal losses are constant, we can model the total friction as:

$$F_{total} = F_{internal} + F_{rolling}$$

Since the rolling friction at 2g is twice that at 1g, we can use the total friction values of 105 N and 115 N to calculate the rolling friction at 1g. Writing the above equation for 1g and 2g, and solving simultaneously by elimination of variables, we obtain:

$F_{rolling} = 10\ N$
$F_{internal} = 95\ N$

In practice, the internal losses will increase slightly at higher gravities due to increased current requirements. But as a first approximation for the distribution of power losses during locomotion, we can use the following mathematical model:

$$F_{total} = 95\ N + 10\ N \cdot \left(\frac{g}{9.8\ N/kg}\right)$$

where "g" is the gravitational field intensity of the planet to be explored.

C. Direct measurement of the energy used for communication requires values for both the power expended and the signal duration ($\Delta E = P \cdot \Delta t$). Measurement of the power requires values for both the voltage drop across the signal device (not the battery voltage), and the current flowing through that device ($P = I \cdot \Delta V$). These measurements are easy for the horn and LED, but difficult for the voice and RF signalling options. The comparison is further complicated by the fact that each of these signalling devices has a different effective range and speed of operation, e.g., an SOS can be sent in microseconds via RF. Here is an approximate comparison of the various signalling modes, in order of increasing energy consumption (measurement notes follow):

Device	**Signalling Method (optimized?)**	**I (amps)**	**ΔV (volts)**	**t (s)**	**ΔE (= I·ΔV·t)**
RF	send characters "SOS" @ 1200 baud	0.160	6.2	0.0025	0.0025 J
HORN	three ½ s beeps, three 1 s beeps, three ½ s beeps	0.003	4.5	6.0	0.08 J
VOICE	say "S-O-S"	0.08 AC	0.7 AC	2.6	0.1 J
LED	using a single LED: three ½ s blinks, three 1 s blinks, three ½ s blinks	0.244	1.8	6.0	2.6 J
ARM	raised arm, flexing elbow ±30°: 3 right waves, 3 left waves, 3 right waves	5.3	6.2	400	1.3×10^4 J

Data for the preceding table were obtained using various methods and assumptions:

RF: current and voltage values provided by manufacturer, assumed 1200 baud transfer rate through port to transmitter, assumed carrier wave power predominates over modulation power
HORN: measured current increase from electronics battery and voltage drop across horn
VOICE: measured AC current to, and AC voltage drop across, the speaker itself, adjusted pitch of sound card to approximate 60 Hz tone, assumed 0.7 power factor, used single phoneme (OO)
LED: used published values for this diode (IN41418)
ARM: used approximate values quoted by GRC

D. The charger nest provided with RB5X applies an overvoltage of 7.5 V at a current of 1.5 A during the charge cycle. That works out to 11.25 W peak charging power. As the battery approaches full charge the current decreases in response to increasing back-voltage from the battery. Current is limited to prevent overheating and outgassing of severely depleted (or damaged) batteries. This is a well-understood process, used routinely in most of battery chargers.

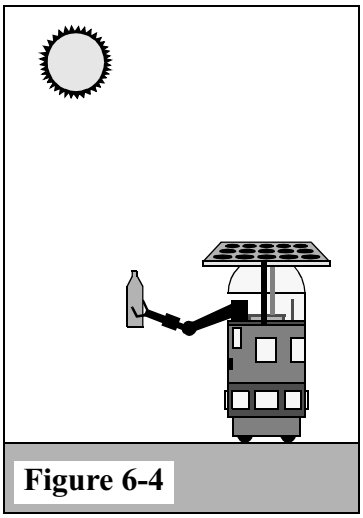


Figure 6-4

Replacing the charger with an onboard solar panel would be a simple exercise. Many commercial sources of panels in the 10-15 W range are available, though most are rated at 12 V. The output would need to be reduced to 7.5 V using either a voltage divider or Zener diode. Current limiting is built into solar panels by simply choosing one rated at 1.5 A peak output.

Alternatively, you could build a custom-sized panel from individual cells, providing the requisite 7.5 V and 1.5 A exactly (in full sunlight). At lower light levels the charge rate would be less, but charge would still accumulate — as the light level on a solar cell decreases, the current will also decrease, but the voltage remains nearly constant.

So how large a solar panel would be needed? Could RB5X even support it? Easily. Currently available crystalline silicon solar cells have an efficiency of around 15%. On Earth, the solar flux peaks at about $1000 W/m^2$. Simple arithmetic shows that a solar panel only 30 cm x 30 cm would suffice. With framing and spacing between cells, maybe 40 cm x 40 cm. This is just the right size for mounting over the dome without seriously compromising the robot's balance. The panel would add only 1-2 kg of mass, and also provide some shade for RB5X's brain. On Mars, of course, the solar flux is less by a factor of $(1 AU/1.52 AU)^2$, requiring a solar panel around 65 cm square to make up for the 57% lower solar flux.

Mission 7: Sample Acquisition

Mission Questions

A. There is, of course, no standard answer here. But it is an interesting exercise to actually shoot the photos described. Reproduce them at 8½" x 11" size for distribution to students; this can be done easily with a digital camera, and only grayscale reproductions are necessary. Students enjoy the challenge of looking for the differences. If you have a particularly static classroom, you may want to "engineer" a few subtle changes yourself, just to give them something to look for. There's a popular syndicated puzzle carried in many newspapers based on this idea, though it uses cartoon panels rather than actual photos.

Some of the obvious changes that would be visible in an average classroom include:

- direction of light from windows
- position of wastebasket
- surface of the teacher's desk
- condition of chalkboard
- time shown on clock
- position of student desks
- litter on floor

B. The easiest method is to figure out ahead of time the size of a 5.0 N rock on Mars. This could then be compared to the size of the grippers at the end of RB5X's robotic arm (about 7 cm when fully opened). You could find by experiment that the average 5.0 N spherical rock on Earth has a diameter of (coincidentally) about 7 cm. Since the gravity on Mars in 0.38 that of Earth, RB5X could lift (on Mars) a rock that weighs (5 N)/(0.38) or 13.2 N on Earth. Again, by experiment, you can find the average 13.2 N rock has a diameter of about 10 cm. Fig. 7-1 shows how this rock might appear in comparison to its grippers as seen through RB5X's own video camera.

Note: the robotic arm is mounted above the camera; when it's fully extended, grippers down and just above the rock, it will still be in the field of view.

If the rock of interest is not within reach, a somewhat more difficult method is required. Lithospheric rock on terrestrial planets has a typical range of densities well known to geologists. Visual examination, even through a remote camera, can provide further clues about the rock type (sedimentary, metamorphic or igneous) further narrowing the density range.

The average density of "generic rock" is around

$\rho = 3000 \text{ kg/m}^3 \pm 1000 \text{ kg/m}^3$ (depending on type)

or approximately 40 lb/ft^3 in Earth gravity.

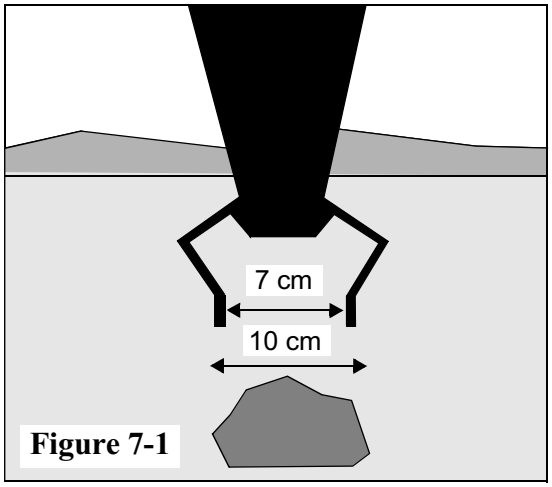


Figure 7-1

If the volume of the rock is known, its weight can then be calculated from the equation

$$F_g = m \cdot g = (\rho \cdot V) \cdot g$$

where "g" is the local gravitational field intensity (3.69 N/kg for Mars) and "V" is rock's volume.

How the volume is calculated depends on the shape of the specimen. Many rocks can be closely enough approximated as spheres, cylinders or parallelepipeds without introducing huge errors.

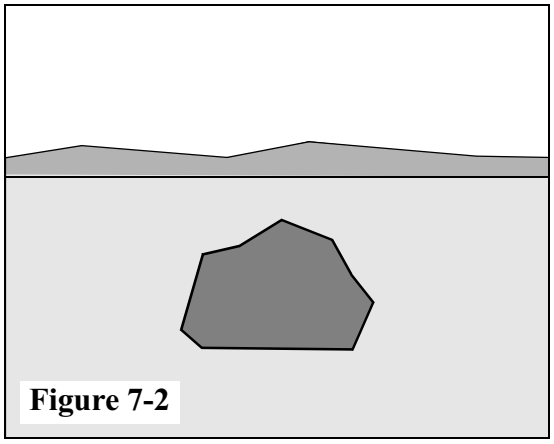


Figure 7-2

Also needed is a scale for the image. Consider Fig. 7-2 without the robotic arm in view; is the rock in the image small and close, or large and distant? There's no way to tell without additional information. See Exploring Further A for an explanation of how this can be done.

Suppose the rock has a measured volume of $V = 0.026\ m^3$ (to continue the Exploring Further A calculation). If the rock is on Mars, and has a typical density, then its weight would be:

$$F_g = m \cdot g = (\rho \cdot V) \cdot g = (3000\ kg/m^3)(0.026\ m^3)(3.96\ N/kg) = 315\ N$$

This rock would exceed the lifting capacity of RB5X's robotic arm.

C. In order to prevent biological contamination, all parts of the delivery spacecraft and lander that come in contact with the rover/sampler would need to be sterilized before launch. Most of the assembly is done in a "clean room" anyway, so this presents little additional difficulty.

NASA, and its counterparts in other countries, recognize the need for this type of precaution, not only for validating the possible discovery of life on other worlds, but also for the ecological protection of the worlds they visit. Introducing a geogenic organism into an ecosystem with none of its usual predators could have disastrous consequences. Ask any Australian about rabbits.

The surface of Mars is highly oxidized by ultraviolet light that penetrates its thin atmosphere, so indigenous life forms, if they exist, are likely subsurface. Robotic missions of the past and future will continue to use "coring" as an effective method of sample acquisition. Coring involves the drilling and extraction of a vertical cylinder of soil. These cores can often be read like tree rings, revealing historical information as age increases with depth. It is in such cores that life on another world may first be discovered. If the cores are to be returned to Earth, and if they contain forms of alien life, there is the possible threat of reverse contamination. Sample return missions to Mars are in the planning stages now — all include strict quarantine procedures with triple-redundant safety mechanisms.

Few people remember that when US astronauts first returned from the Moon, they were locked in a sealed metal container for over 24 hours. Only after microscopic examination had revealed they were free of any "Moon microbes" were they released from their "captivity."

Mission Problem

NASA's current sample search strategy is to move the rover in ever-increasing circles away from the lander. Frequent trips back to the lander (which contains the sample return rocket) guarantee that, if a rover failure occurs, at least some samples will get back. This strategy balances the risk of "no sample return" vs. "area explored." Multiple trips back to the lander are time consuming, since rovers move slowly, but without a sample return the mission fails in its primary goal.

We will use a random search pattern. RB5X will choose movement directions with a randomly generated number. This may not be the most efficient search method, since a given grid block could be visited more than once, but it is an optimal solution given program size limits. If you try to write a program that searches geometrically (ever-increasing circles, consecutive columns, etc.) and negotiates obstacles within that geometric constraint, the program size increases to the point where you run out of either available variables or memory.

This is a tough mission to program, but some of the needed routines have already been written in previous missions — PROBE and FINDHOME can be recycled from Mission 5 with only a few minor changes to each. The main program, FINDROCK, will call these other programs as tasks.

Here is the slightly modified version of PROBE to be used as a subroutine in FINDROCK:

PROBE: An RCL Subroutine for Detecting Mountain, Rock or Crevice

```
1.  CALL THIS ROBOT TASK LINE [MT PROBE]
2.  SET THE TINY-BASIC VARIABLE [G] EQUAL TO [0]
3.  COMPILE THE BASIC STATEMENT [LINK#1100:DELAY20] WHICH MEANS [GET D]
4.  COMPILE THE BASIC STATEMENT [E=(30*D - 30*73)/22] WHICH MEANS [CALC CM]
5.  COMPILE THE BASIC STATEMENT [IF E<10 GOTO 56] WHICH MEANS [MT]
6.  CALL THIS ROBOT TASK LINE [RK&CR PROBE]
7.  BEGIN A COUNTED LOOP CALLED [C] BEGINNING AT [1] ENDING AT [5]
8.    MOVE DISTANCE FORWARD THIS MANY FEET [0.1]  (Note:  distance is in "meters.")
9.    WAIT FOR [2] SECONDS
10.   COMPILE THE BASIC STATEMENT [Y=@#7800:IF Y=251 GOTO 57] WHICH MEANS [RK]
11.   COMPILE THE BASIC STATEMENT [X=#02] WHICH MEANS [TURN ON IR]
12.   COMPILE THE BASIC STATEMENT [@#7801=@#7801 OR X] WHICH MEANS [SET BIT]
13.   COMPILE THE BASIC STATEMENT [IF (@#7802 AND #40)=64 GOTO 58] WHICH MEANS [CR]
14. REPEAT THE COUNTED LOOP FOR THE COUNTER [C]
15. JUMP TO THE LINE CALLED [RETURN1]
16. CALL THIS ROBOT TASK LINE [RETREAT]
17. BEGIN A COUNTED LOOP CALLED [F] BEGINNING AT [1] ENDING AT [C]
18.   MOVE DISTANCE BACKWARD THIS MANY FEET [0.1]  (Note:  distance is in "meters.")
19.   WAIT FOR [2] SECONDS
20. REPEAT THE COUNTED LOOP FOR THE COUNTER [F]
21. JUMP TO THE LINE CALLED [RETURN1]
22. COMPILE THE BASIC STATEMENT [G=1:GOTO 59] WHICH MEANS [MT]
23. COMPILE THE BASIC STATEMENT [G=2:GOTO 48] WHICH MEANS [RK]
24. COMPILE THE BASIC STATEMENT [G=3:GOTO 48] WHICH MEANS [CR]
25. CALL THIS ROBOT TASK LINE [RETURN1]
26. COMPILE THE BASIC STATEMENT [RETURN] WHICH MEANS [RETURN]
```

PROBE will attempt to move the robot forward one grid block. If a mountain, rock or crevice is detected, the robot will back off to its previous coordinates. The type of obstacle encountered is identified by the variable G (see lines 22-24). Note that since PROBE is called at the start of the program, it receives the same line number as in FINDHOME for subroutine calls, i.e., GOSUB 31.

Also needed is a subroutine for rotation. Based on the direction RB5X is currently facing, and on the direction it wants to turn to, we must program a specific rotation angle. The current direction will be held by the variable H (1=N, 2=E, 3=S, 4=W). The desired direction is specified by the variable U employing the same numerical code.

ROTATE: An RCL Program to Determine Rotation Angles

```
1.  CALL THIS ROBOT TASK LINE [ROTATE]
2.  COMPILE THE BASIC STATEMENT [IF U=H GOTO 78] WHICH MEANS [R=0]
3.  COMPILE THE BASIC STATEMENT [IF (U-H=1) OR (H-U=3) GOTO 70] WHICH MEANS [R=90]
4.  COMPILE THE BASIC STATEMENT [IF (U-H=2) OR (H-U=2) GOTO 75] WHICH MEANS [R=180]
5.  SPIN LEFT 90 DEGREES
6.  JUMP TO THE LINE CALLED [RETURN2]
7.  SPIN RIGHT 90 DEGREES
8.  JUMP TO THE LINE CALLED [RETURN2]
9.  SPIN CLOCKWISE FOR [180] DEGREES
10. CALL THIS ROBOT TASK LINE [RETURN2]
11. SET THE TINY-BASIC VARIABLE [H] EQUAL TO [U]
12. COMPILE THE BASIC STATEMENT [RETURN] WHICH MEANS [RETURN]
```

Note that we needed to call line 10 RETURN2 and not just RETURN. This is to distinguish it from line 25 of PROBE. Since both PROBE and ROTATE are called by the program FINDROCK, we must avoid duplicate labels when identifying lines by name.

We need one more short subroutine to avoid the 1000 byte program size limit. LOGIC performs an analysis of the obstacle detection loop in PROBE.

LOGIC: An RCL Program to Respond to Obstacles

```
1.  CALL THIS ROBOT TASK LINE [LOGIC]
2.  COMPILE THE BASIC STATEMENT [IF (G=1) OR (G=3) GOTO 92] WHICH MEANS [OBSTACLE]
3.  COMPILE THE BASIC STATEMENT [IF G=2 GOTO 123] WHICH MEANS [ROCK]
4.  COMPILE THE BASIC STATEMENT [RETURN] WHICH MEANS [RETURN]
```

Note that it is acceptable to leave a subroutine using a GOTO or JUMP statement without actually reaching the RETURN statement at the end.

We can now proceed to the main program solution for this Mission Problem. RB5X needs to keep track of its position at all times. The robot's current coordinates will be stored as an ordered pair of variables (A,B), initialized at (6,0). RB5X begins the mission facing north as usual.

FINDROCK: An RCL Program for Sample Acquisition

```
1.  PREPARE THE ROBOT
2.  RESERVE THE TINY-BASIC VARIABLE NAMED [A]
3.  RESERVE THE TINY-BASIC VARIABLE NAMED [B]
4.  RESERVE THE TINY-BASIC VARIABLE NAMED [C]
5.  RESERVE THE TINY-BASIC VARIABLE NAMED [E]
6.  RESERVE THE TINY-BASIC VARIABLE NAMED [F]
7.  RESERVE THE TINY-BASIC VARIABLE NAMED [G]
8.  RESERVE THE TINY-BASIC VARIABLE NAMED [H]
9.  RESERVE THE TINY-BASIC VARIABLE NAMED [U]
10. PROBE
11. ROTATE
12. LOGIC
13. SET THE TINY-BASIC VARIABLE [A] EQUAL TO [6]
14. SET THE TINY-BASIC VARIABLE [B] EQUAL TO [0]
15. BEGIN A LOOP
16.   EXIT IF SONAR DISTANCE IS [MORE] THAN [0] FEET  (Note:  distance is in "meters.")
17. REPEAT THIS LOOP
18. CALL THIS ROBOT TASK LINE [HUNT]
19. COMPILE THE BASIC STATEMENT [U=RND(1,4)] WHICH MEANS [RND DIRECTION]
20. COMPILE THE BASIC STATEMENT [IF U=H GOTO 92] WHICH MEANS [BEEN THERE]
21. COMPILE THE BASIC STATEMENT [GOSUB 61] WHICH MEANS [ROTATE]
22. COMPILE THE BASIC STATEMENT [IF U=4 GOTO 117] WHICH MEANS [GO WEST
23. COMPILE THE BASIC STATEMENT [IF U=3 GOTO 111] WHICH MEANS [GO SOUTH
24. COMPILE THE BASIC STATEMENT [IF U=2 GOTO 105] WHICH MEANS [GO EAST
25. CALL THIS ROBOT TASK LINE [GO NORTH]
26. COMPILE THE BASIC STATEMENT [IF B+1>9 GOTO 92] WHICH MEANS [LIMITS]
27. COMPILE THE BASIC STATEMENT [GOSUB 31] WHICH MEANS [PROBE]
28. COMPILE THE BASIC STATEMENT [GOSUB 81] WHICH MEANS [LOGIC]
29. COMPILE THE BASIC STATEMENT [B=B+1:H=0] WHICH MEANS [ADVANCED]
30. JUMP TO THE LINE CALLED [HUNT]
31. CALL THIS ROBOT TASK LINE [GO EAST]
32. COMPILE THE BASIC STATEMENT [IF A+1>9 GOTO 92] WHICH MEANS [LIMITS]
33. COMPILE THE BASIC STATEMENT [GOSUB 31] WHICH MEANS [PROBE]
34. COMPILE THE BASIC STATEMENT [GOSUB 81] WHICH MEANS [LOGIC]
35. COMPILE THE BASIC STATEMENT [A=A+1:H=0] WHICH MEANS [ADVANCED]
36. JUMP TO THE LINE CALLED [HUNT]
37. CALL THIS ROBOT TASK LINE [GO SOUTH]
38. COMPILE THE BASIC STATEMENT [IF B-1<0 GOTO 92] WHICH MEANS [LIMITS]
39. COMPILE THE BASIC STATEMENT [GOSUB 31] WHICH MEANS [PROBE]
40. COMPILE THE BASIC STATEMENT [GOSUB 81] WHICH MEANS [LOGIC]
41. COMPILE THE BASIC STATEMENT B=B-1:H=0] WHICH MEANS [ADVANCED]
42. JUMP TO THE LINE CALLED [HUNT]
43. CALL THIS ROBOT TASK LINE [GO WEST]
44. COMPILE THE BASIC STATEMENT [IF A-1<0 GOTO 92] WHICH MEANS [LIMITS]
45. COMPILE THE BASIC STATEMENT [GOSUB 31] WHICH MEANS [PROBE]
46. COMPILE THE BASIC STATEMENT [GOSUB 81] WHICH MEANS [LOGIC]
47. COMPILE THE BASIC STATEMENT [A=A-1:H=0] WHICH MEANS [ADVANCED]
48. JUMP TO THE LINE CALLED [HUNT]
49. CALL THIS ROBOT TASK LINE [ROCK]
50. MOVE DISTANCE FORWARD THIS MANY FEET [0.5]  (Note:  distance is in "meters.")
51. SET THE TINY-BASIC VARIABLE [U] EQUAL TO [3]
```

```
52. COMPILE THE BASIC STATEMENT [GOSUB 61] WHICH MEANS [ROTATE]
53. FINDHOME
```

RB5X now has the rock nestled securely in its RAD, is facing south at coordinates (A,B), and is ready to return to the landing site. FINDHOME, called in line 53, is a modified version of the AI homing program used back in Mission 5.

FINDHOME: An RCL Program to Bring the Rock Home

```
1.  CALL THIS ROBOT TASK LINE [GO LAT]
2.  BEGIN A LOOP
3.    COMPILE THE BASIC STATEMENT [GOSUB 31] WHICH MEANS [PROBE]
4.    COMPILE THE BASIC STATEMENT [IF G>0 GOTO 137] WHICH MEANS [GO LON]
5.    SET THE TINY-BASIC VARIABLE [B] EQUAL TO [B-1]
6.    COMPILE THE BASIC STATEMENT [IF B=0 GOTO 137] WHICH MEANS [GO LON]
7.  REPEAT THIS LOOP
8.  CALL THIS ROBOT TASK LINE [GO LON]
9.  COMPILE THE BASIC STATEMENT [IF (A=6) AND (B=0) GOTO 175] WHICH MEANS [HOME]
10. COMPILE THE BASIC STATEMENT [IF A<6 GOTO 141] WHICH MEANS [HEAD EAST]
11. COMPILE THE BASIC STATEMENT [IF A>6 GOTO 158] WHICH MEANS [HEAD WEST]
12. CALL THIS ROBOT TASK LINE [HEAD EAST]
13. SPIN LEFT 90 DEGREES
14. BEGIN A LOOP
15.   COMPILE THE BASIC STATEMENT [GOSUB 31] WHICH MEANS [PROBE]
16.   COMPILE THE BASIC STATEMENT [IF G=0 GOTO 154] WHICH MEANS [ADVANCE]
17.   SPIN RIGHT 90 DEGREES
18.   JUMP TO THE LINE CALLED [GO LAT]
19.   SET THE TINY-BASIC VARIABLE [A] EQUAL TO [A+1]
20.   COMPILE THE BASIC STATEMENT [IF (A=6) AND (B=0) GOTO 180] WHICH MEANS [HOME]
21. REPEAT THIS LOOP
22. CALL THIS ROBOT TASK LINE [HEAD WEST]
23. SPIN RIGHT 90 DEGREES
24. BEGIN A LOOP
25.   COMPILE THE BASIC STATEMENT [GOSUB 31] WHICH MEANS [PROBE]
26.   COMPILE THE BASIC STATEMENT [IF G=0 GOTO 171] WHICH MEANS [ADVANCE]
27.   SPIN LEFT 90 DEGREES
28.   JUMP TO THE LINE CALLED [GO LAT]
29.   SET THE TINY-BASIC VARIABLE [A] EQUAL TO [A-1]
30.   COMPILE THE BASIC STATEMENT [IF (A=6) AND (B=0) GOTO 185] WHICH MEANS [HOME]
31. REPEAT THIS LOOP
32. CALL THIS ROBOT TASK LINE [HOME]
33. SPIN CLOCKWISE FOR [180] DEGREES
34. JUMP TO THE LINE CALLED [DONE]
35. SPIN LEFT 90 DEGREES
36. JUMP TO THE LINE CALLED [DONE]
37. SPIN RIGHT 90 DEGREES
38. CALL THIS ROBOT TASK LINE [DONE]
```

RB5X has returned the rock to its landing site. Mission accomplished.

Exploring Further

A. An object of known size (like a ruler) could be placed next to the rock, as if often done during documentary photography of archeological relics. This is not an option for all robotic explorers. Rather, a ranging device (sonar or radar) can be used to find the distance from the camera to the rock. This information, combined with the known f.o.v. (field of view) of the camera, allows the rock's size to be determined using simple plane trigonometry as follows.

Assume the image in Fig. 7-2 was taken by RB5X's camera, which has a 60° horizontal f.o.v. The image is 7.0 cm wide, and the rock is 2.5 cm wide (in the image). The rock's angular size θ is proportional to its linear size, so we can write these ratios:

$$\frac{2.5\text{ cm}}{7.0\text{ cm}} = \frac{\theta}{60°}$$

and solve for the angular size of the rock (20° in this example). Assume also that RB5X has measured a sonar range for the rock of d = 1.5 m. The actual size of the rock can now be found using the trigonometric relationship shown in Fig. 7-3:

$$D = \frac{2d\tan\left(\frac{\theta}{2}\right)}{1 + \tan\left(\frac{\theta}{2}\right)}$$

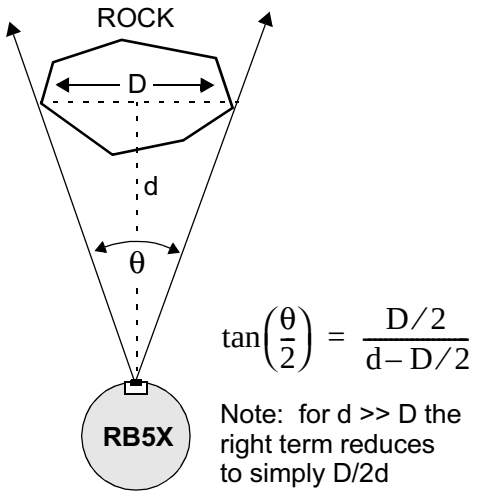


Figure 7-3: Top View (not to scale)

Using the given values for d and θ results in a calculated rock diameter of D = 0.50 m (or a radius of R = 0.25 m). If we approximate the shape of the rock as a sphere, its volume would be

$$V = \frac{4}{3}\pi R^3 = 0.065\text{ m}^3$$

If you need to know the mass or weight of the rock, you could now use the methods described in the solution to Mission Question B.

B. Fig. 7-4 shows the gripper end of the robotic arm in its nearly-closed position. The three opposing wire "fingers" open to a maximum width of 7 cm. The use of three points of contact improves RB5X's grip on round and irregularly-shaped objects. The opposition distance is controllable from 0 to 100%.

You could increase the maximum width by simply bending the fingers farther apart, but this would also prevent full closure on smaller objects. Attaching finger extenders that overlap as shown increases the maximum grip while still allowing full closure.

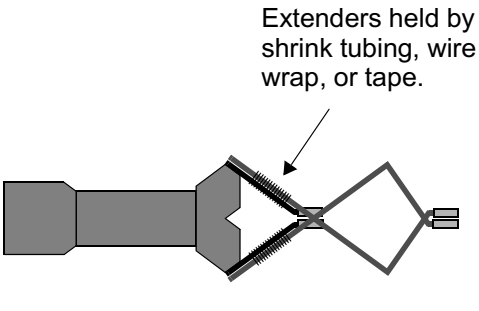


Figure 7-4: Forearm w/ Grip Doubler

C. The least obtrusive location for this "backpack" is at the top of the extension panel used to raise the dome when the arm is attached, and centered over the access panel (bumper 5). Centering is required since the arm cannot move on the Z-axis.

Use a single 0.125" x 1" bolt through holes drilled near the top of the coffee can and the top of the extender panel (Fig. 7-5). This allows the can to swing out of the way for easy access to switches and cables. With the can's center of mass below the point of support it will hold itself upright.

The exact degrees of rotation required will depend on the exact location of the block, and must be determined experimentally. Given the starting position (1) with wrist rotated to grip along the Z-axis, and the grip fully open, try the following:

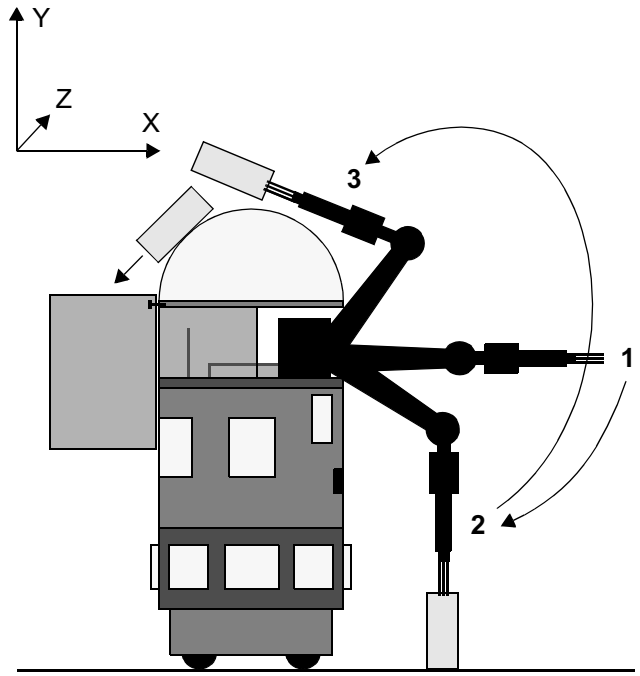


Figure 7-5

1. MOVE ARM AT THE SHOULDER [DOWN] FOR [30] DEGREES
2. MOVE THE FOREARM [IN] FOR [30] DEGREES
3. MOVE THE HAND [CLOSE] FOR [20] %
4. MOVE ARM AT THE SHOULDER [UP] FOR [90] DEGREES
5. MOVE THE FOREARM [OUT] FOR [120] DEGREES
6. MOVE THE HAND [OPEN] FOR [100] %

At this point the block should slide down the back of RB5X's dome and into the "backpack." If not, adjust the rotation degrees until it works consistently. Be sure to start the block of wood at the same location every time. This movement sequence may be useful, once learned, in other applications involving sample acquisition. Be aware that samples harder than wood will scratch the dome while sliding on it.

D. This final exercise is open-ended for good reason. Program solutions from several Missions could be incorporated here. We trust that you have acquired sufficient skills in the programming, operation and modification of your RB5X to successfully define and complete it.

Mission to Mars was designed to increase your expertise in heuristics. Congratulations to those who completed all the Missions; your problem-solving ability has been significantly advanced. At this point, if you can visualize it, you can probably program it. There are no limits to what you, and your RB5X, can do — beyond the nine variables and 8000 byte memory.

Prepare to go where no robot has gone before.