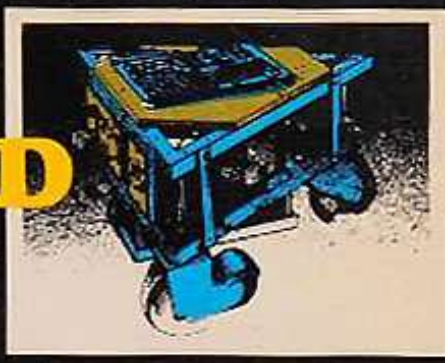
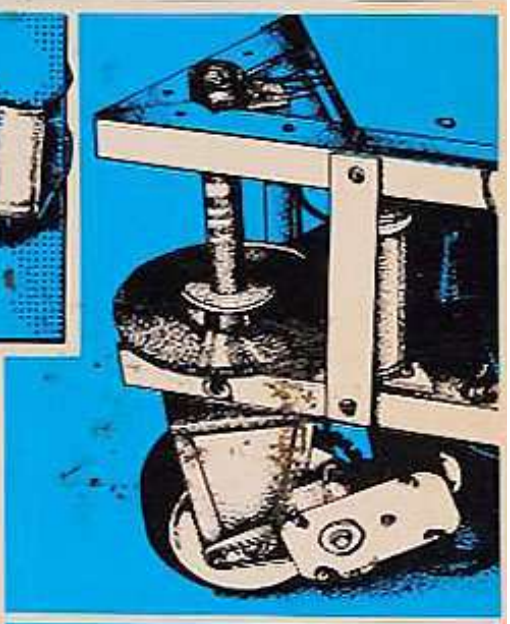
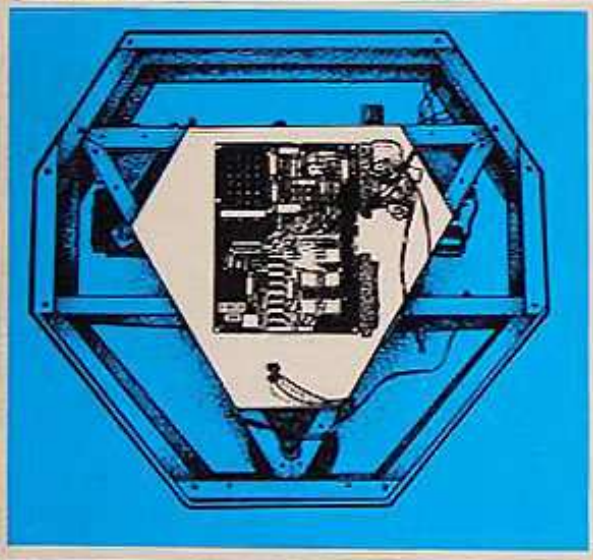
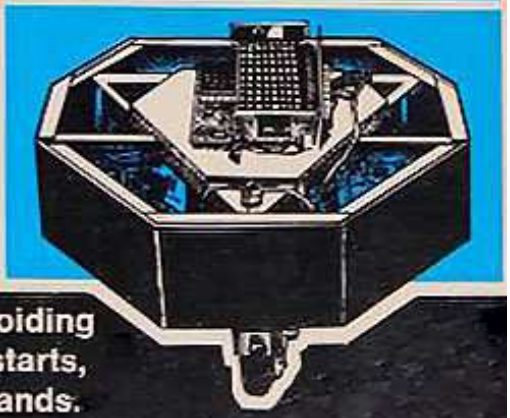
HOW TO BUILD A COMPUTER-CONTROLLED ROBOT



Tod Loofbourrow







"Mike" moves under his own control avoiding all objects placed in his way . . . stops, starts, and changes direction on voice commands.

HAYDEN

HOW TO BUILD A COMPUTER-CONTROLLED ROBOT

HOW TO BUILD A COMPUTER-CONTROLLED ROBOT

TOD LOOFBOURROW

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

All photographs by The Art Works Photography, Parsippany, N.J.

Library of Congress Cataloging in Publication Data

Loofbourrow, Tod.
How to build a computer-controlled robot.

Includes index.
1. Automata--Amateurs' manuals. 2. Microcomputers --Amateurs' manuals. I. Title.
TJ211.L66 629.8'92 78-5879
ISBN 0-8104-5681-8

Printed in the United States of America

6 7 8 9 PRINTING

80 81 82 83 84 85 86 YEAR

preface

Every person has a certain fascination with robots. Some of us are intrigued enough to try to build one. Robotics is one of the most interesting results of today's advanced technology. It combines electronics and mechanics with biology and psychology, raising questions about the definition of "intelligence."

The invention of the integrated circuit (IC) chip and the consequent development of microcomputers have revolutionized the field of robotics. With microcomputers, the great amount of logic circuitry previously required for robots can be discarded in favor of software. Therefore, the cost of robot building drops considerably.

"Microtron" ("Mike"), the robot detailed in this book, uses the ultimate in current hobby computer technology. Despite his complexity, Mike is well within the budget of the average hobbyist. Mike is not difficult to build. The only technical requirements for building him are the ability to read and understand an electronic circuit diagram, and a reasonable mechanical ability.

Mike operates in either of two ways. He can be totally independent, exploring an area with his various "senses," or he can be operated manually from a control box.

Mike is built in three main stages. As you complete each stage of your Mike's construction, he becomes more complex, more independent, and more "human." You'll have fun building and working with him, and you will experience the thrill of creating a type of intelligence observed only by robot builders.

I want to thank the members of the Amateur Computer Group of New Jersey, whose suggestions have helped make Mike what he is today. I also want to give special thanks to my family without whose support and encouragement Mike would not have been possible.

TOD LOOFBOURROW

contents

Preface v

1. Introduction to "Mike" 1

Stage I — Mobility 2
Stage II — Independence 3
Stage III — Advanced Sensory Systems 5
Checklist 7

2. Constructing the Basic Framework 8

Basic Framework Parts List 8
Lower Triangle 9
Upper Triangle 12
Battery Cage 14
Directional Control Assembly 16

3. Power Supply, Speed Control, and Directional Control Circuits 19

Power Supply Circuit Parts List 19
Speed Control Circuit Parts List 20
Directional Control Circuit Parts List 20
Power Supply Circuit 21
Speed Control Circuit 23
Directional Control Circuit 26

4. Secondary Circuits, the Microcomputer, and Software 29

Parts List 29
A/D Circuit Parts List 29
Inverter Circuit Parts List 30
Analog to Digital (A/D) and Inverter Circuits 30
Joystick and Directional Control Pot 32
The Microcomputer 36
Software 37
Joystick Control Program 43
The Self-Direction Program 49
Self-Direction Program 53

5. Impact Sensors 56

Parts List 56
Impactor Sensor Detector Parts List 56
Impact Sensor Selector Parts List 56
Outer Frame 57
Impact Sensors 62
Impact Sensor Circuitry 65
Software 68
Impact Sensor Control Routine 72

6. Ultrasonics 75

Parts List 75
Ultrasonic Detection Circuitry 76
Software 79
Ultrasonic Detection Program 80
Tuning the Ultrasonic Detection Circuit 83

7. Voice Recognition 84

Voice Recognition Parts List 84
Voice Recognition Circuit Parts List 84
Voice Recognition Circuitry 85
Software 88
Voice Recognition Program 109

8. Mike's Future 123

Appendix 129

Index 131

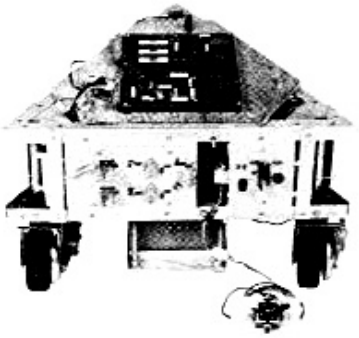
how to build
a computer-controlled robot

chapter one introduction to "mike"

The question most commonly asked about robotics is, "Why build a robot?" If the idea of having a robot to act as your companion, entertainer, or slave does not intrigue you, perhaps one of the following reasons will. One reason for building a robot is that it is an exciting way to learn about electronics and microcomputers. You can also learn the actual limits of our present technology and perhaps make improvements in it. Other reasons are the fun of building a robot and the excitement of watching your creation "come to life." However, the most important factor comes from an inner curiosity about nonhuman forms of intelligence. Motivation to create this intelligence is the driving force behind most roboticists.

Robots can be thought of as representing a step in evolution. As man evolved, he became more and more intelligent. Now man is attempting to create intelligence in the form of robots. Robots are almost an extension of man's intelligence.

As you proceed with Mike's construction, he evolves in his own way in a sense. He is built in three stages. In Stage I, he is operated by means of a joystick, and he is totally under your control. Unfortunately, if you should happen to drive him into a barrier, he has no provision for resisting your command. In Stage II, Mike becomes independent. He can "see" and "feel" his environment and can react accordingly. In Stage III, Mike gains the ability to hear and respond to whistles and certain voice commands. Stage III also contains various ideas, as yet not implemented, for Mike's further development.



stage I—mobility

In the first stage, the mechanics of Mike are constructed. His basic triangular framework is constructed of angle aluminum. It houses his power supply, which is a 12 V car battery. His mobility is provided by three motorized wheels. Mike is controlled with a joystick that is connected to him by a thin cable. You can make him move at five speeds forward and five speeds reverse, his top speed being about walking speed. He can also be made to turn at angles from 0° to 60° in either direction by rotating his front wheel with a motor.

At this stage, Mike is about 14 in. high and is shaped somewhat like a spearhead. Each side of his triangular frame is 23 in. long. On top of the frame rests Mike's brain, a Kim–1 microcomputer ("micro" for short).* The Kim directs Mike's operation and allows him to execute your commands. The joystick sends out two voltages to an analog-to-digital (A/D) circuit. The A/D circuit converts these voltages to digital values, which are stored in the microcomputer. Through these values, the Kim can determine the turning angle of the front wheel and the speed at which Mike is moving and compare these with your commands. If your commands differ from the actual position of the front wheel or from the speed of the motorized wheels, the position or speed is changed until it meets your command.

I have found that with Mike's three wheels he is powerful enough to push or pull 150 lb or carry over 600 lb with ease. You can test him over a variety of terrains to determine the limits of his capabilities. His three main circuits—the power supply circuit, the speed control circuit, and the directional control circuit—are mounted on the sides of the triangular frame. The Kim–1 rests on a sheet of ½-in. plywood, which is bolted to the top of the aluminum frame. The Kim is connected to three circuits—the power supply circuit, the A/D circuit, and the inverter circuit. The power supply circuit supplies the Kim as well as the logic circuitry with power. The A/D circuit allows the micro to compare

*The terms *Kim–1*, *Kim*, *Micro*, *Microprocessor*, *Microcomputer*, *Computer*, *Processor*, and *Mike's brain* will be used interchangeably throughout this book.

commands from the joystick with the actual position of the front wheel. The inverter circuit changes the output of the micro from logic 1 to logic 0. This circuit leads to the speed control and the directional control circuits, which in turn lead to the motorized wheels and directional control motor, respectively.

During most of Stage I, Mike remains totally under your control. The only actions of which he is capable are those dictated by the joystick. At the end of the Stage I section of this book, there is a program that provides Mike with a certain amount of independence prior to the second stage of development. This program is called the self-direction program. The self-direction program causes Mike to move in a predetermined pattern. The pattern can be changed by changing the values in two tables that are stored in the computer. I have included a program for a pattern that forms an asterisk and one that forms a cloverleaf pattern. It should not be hard for you to write programs for your own patterns once you understand the basic concepts of the self-direction program.

At the completion of Stage I, Mike will have become a robot more on the order of a machine than a cyborg (humanoid robot).

stage II—independence

In the second stage of construction, Mike becomes independent. Now a true robot, he makes his way about an area totally independent of a controller. Mike looks different from the way he did in Stage I. Around his original triangular framework is an eight-sided frame. Although Mike is still 14 in. high, he has expanded to 27 in. in width and has begun to look like the base of a full-sized robot.

In Stage II, Mike is equipped with ultrasonic "sight." His "eye" is a small, ultrasonic transducer which sends out every quarter of a second a wave of sound inaudible to the human ear. By noting the length of time required for the sound wave to be reflected back to the transducer, Mike can determine the distance of an object in his path. When something blocks his path, he "sees" it with his ultrasonic sensor. He backs up, turns to the right, and proceeds on his way, avoiding the obstacle. The

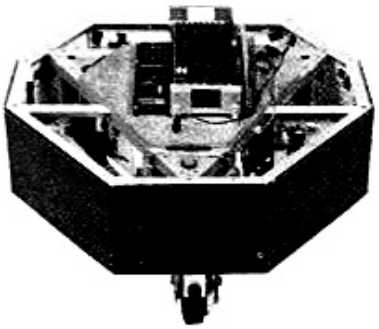
range of the ultrasonics can be varied from 1 in. to more than 10 ft by changing one number in the computer.

If Mike's ultrasonics don't see an object in his path (perhaps because it is too far to one side), contact with the object is made by the main sensory system—the impact sensors. The impact sensors feel that an object is contacting them, and Mike moves away from the object. The impact sensors absorb much of the impact of contacting an object. Therefore, Mike touches objects so lightly that you can actually let him bump into you. Each one of Mike's eight impact sensors contains five ribbon switches sandwiched between two sheets of aluminum. Each sensor is attached to one side of Mike's eight-sided frame.

The reaction that Mike has when one of his impact sensors is hit depends on which sensor receives the impact. For example, if the front impact sensor is contacted, Mike quickly backs away from the object he touched. Then he turns right and proceeds forward. Contact with a side sensor causes him to back up and turn away from the side that was hit. Mike's movements in reaction to each sensor impact are determined by a table of values stored in Mike's "brain." This table can be changed easily to give Mike sensor responses as complex as you wish. The impact sensor responses I include for Mike enable him to seek out and pass through doorways. The system allows Mike to find his way into, around, and out of a room.

On occasion, a low-level object such as a curb may not come into contact with the impact sensors or be in the range of the ultrasonic sensors. In that case, Mike touches the object lightly with a soft rubber feeler. This contact triggers a switch that makes Mike react as he does to an impact sensor hit. The feelers ride about 2 in. above the ground and will detect nearly any obstacle Mike may encounter. Each feeler contains an SPDT-center-off-momentary-contact toggle switch. Two feelers are located near Mike's front, and two are located in back.

At the end of Stage II Mike is an independent robot. Of course, you can take control of him at any time by plugging in the joystick and loading the joystick control program. It is fascinating to watch Mike move around, "seeing" objects before he comes near them and following the behavior patterns that you have programmed.



stage III—advanced sensory systems

In Stage III, Mike is enabled to recognize and respond to whistles at certain frequencies and to voice commands. This stage is presently being worked on by the author. At all times, Mike is listening. He is comparing any sounds that come into his "ear," (a microphone) with word templates that are stored in the computer. When the sounds match one of the templates to a reasonable degree, Mike determines that he has heard one of the words he knows. Immediately, Mike executes the action or actions specified by the word. For example, if Mike recognizes a "left" command, he turns left. As the voice circuit and program have been tested and improved, Mike has progressed from whistles to words. Whistling at a constant frequency is relatively easy for Mike to discern. Spoken words are much more difficult. A large amount of testing and modification has been necessary in order for Mike to recognize spoken words.

As I have already said, words are matched by comparing them with templates. You form these templates through a special program called the template control program. You can place any reasonably short words or sounds in the templates. The templates are stored in the computer to wait for a matching word or sound to be said. The nature of voice recognition is such that only *your* commands will be heeded. No one else's voice will match your templates.

The last half of Stage III is devoted to the future plans I have for my Mike. One such idea is for using an image sensor camera. An image sensor camera produces a black and white picture. It could be used by Mike actually to "see" his environment. Through the use of this camera, Mike could also be able to pick out certain objects that he recognizes in his surroundings.

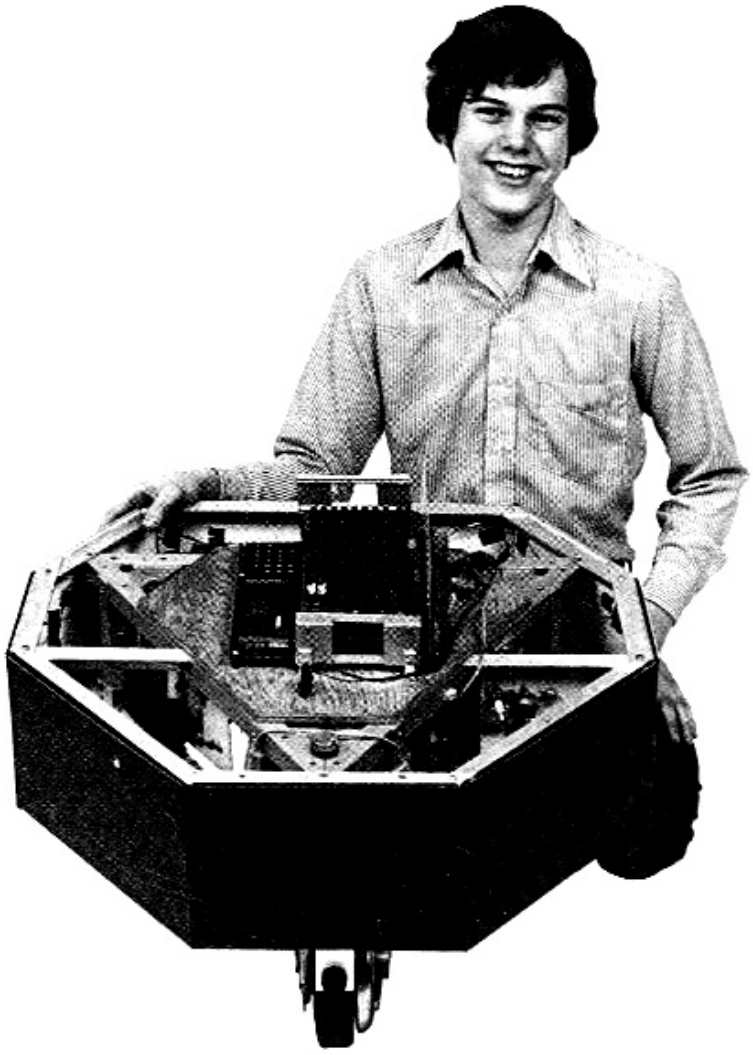
Another system that I plan to add to my Mike is a voice. Mike would be able to respond verbally to voice commands or to talk if it were appropriate. In addition, I am going to build an upper body on Mike's eight-sided frame. I have not yet determined the exact shape that I want his body to assume, although I believe that his height will be about 5 ft.

I am planning to add one or two arms to Mike so that he can manipulate as well as explore his environment. The claws on the arm(s) will contain a type of force sensor so that Mike can apply the ideal amount of pressure to pick up various objects.

Two final additions that I would like to make to Mike are a video terminal and a keyboard. The video terminal and keyboard would be used for testing the image sensor camera and for loading in software. I expect that they will also enable me to create a control language for Mike, with instructions that apply directly to Mike's operation.

As you proceed with your Mike's construction, keep track of your progress with the checklist on the following page. As you complete each step, check the space to its left and date the space to its right.

Now you're ready to begin!



CHECKLIST

Stage I

Date Completed

- ☐ Lower Triangle _____
- ☐ Upper Triangle _____
- ☐ Battery Cage _____
- ☐ Directional Control Assembly _____
- ☐ Power Supply Circuit _____
- ☐ Speed Control Circuit _____
- ☐ Directional Control Circuit _____
- ☐ A/D & Inverter Circuits _____
- ☐ Joystick _____
- ☐ Directional Control Pot _____
- ☐ Kim-1, Mounted _____
- ☐ Joystick Control Program _____
- ☐ Self-direction Program _____

Stage II

- ☐ Outer Frame _____
- ☐ Impact Sensors _____
- ☐ Impact Sensor Circuitry _____
- ☐ Impact Sensor Control Routine _____
- ☐ Ultrasonic Detection Circuitry _____
- ☐ Ultrasonic Detection Program _____

Stage III

- ☐ Voice Recognition Circuitry _____
- ☐ Interrupt Software _____
- ☐ Body _____
- ☐ Speech Synthesis _____
- ☐ Video Terminal & Keyboard _____
- ☐ Arm(s) _____
- ☐ Image Sensor Camera _____

chapter two constructing the basic framework

Basic Framework Parts List:

3 motorized wheels (see text)
1 directional control motor (see text)
20 feet of 1- by 1- by ⅛-in. angle aluminum
1 12-V 84-A-hr car battery and charger (see text)
28 3/16- by 1/2-in. stove bolts
28 3/16-in. nuts
12 3/16-in. flat washers
28 3/16-in. lock washers
9 inches of 1 1/2- by 1/8-in. flat aluminum
A 2-ft by 2-ft square of ½-in. plywood
12 No. 6 1/2-in. wood screws
38 inches of screw rod 1/4-in. to 1/2-in. in diameter
20 nuts (to fit screw rod)
12 lock washers (to fit screw rod)
2 gears, 10:1 ratio, the larger one approximately 5 in. in diameter and the smaller one approximately 1/2 in. in diameter.
1 tube of instant bonding glue

This chapter deals with the construction of Mike's innermost frame. Although building the basic triangular frame is one of the most time-consuming steps of the project, the strength and stability of the end

product will pay off in the long run. This chapter also details construction of the direction control assembly and the mounting of the motorized wheels that provide Mike with his mobility. Once you have completed the frame, you will build a housing for the 12-V battery that powers Mike. The bulk of Mike's Stage I construction will then have been completed.

lower triangle

Mike's inner frame is triangular in shape. I selected a three-sided construction for stability and practicality. The triangle is one of the most stable geometric shapes. A triangle requires only one steerable wheel. Finally, a triangle needs only three wheels, one less than required by a square or a rectangle.

To construct Mike's lower frame you will need three motorized wheels (see Appendix). Together these wheels can carry up to 600 lbs at walking speed. This capacity is more than enough for Mike, who will never weigh more than 200 lbs. When you order the motorized wheels, I suggest that you also order Mike's directional control motor, which will be used later.

While waiting for these items to arrive, you should begin preparing a frame for the wheels. Cut three 23-in. pieces of angle aluminum. Then cut off sections of each piece of aluminum as illustrated in Fig. 2–1. You will be cutting off two $3\frac{1}{2}$- by $\frac{7}{8}$-in. rectangles from each piece of aluminum.

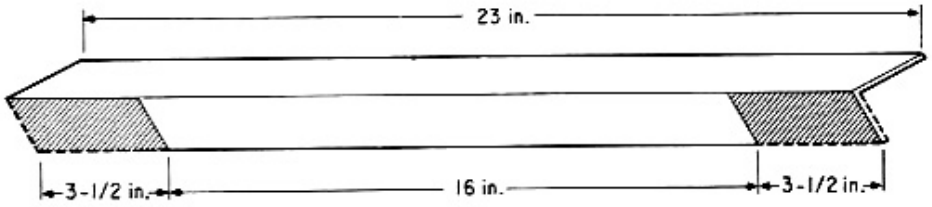


Fig. 2–1. Aluminum side of lower triangle. Cut out shaded pieces.

When your wheels arrive, notice that each wheel is mounted on a shaft (Fig. 2–2). Notice further that each shaft is set in a circular disk shaped like a petrie dish. The shaft can rotate freely in this disk. You will notice that located on the disk of each wheel is a small metal stop protruding below the disk. Observe also that two other stops attached to the wheel frame but not to the disk, extend underneath the disk. This arrangement prevents the shaft from rotating much more than 360°. Turn the disk-mounted stop so that it faces 180° from the motor. Now you are ready to construct the frame.

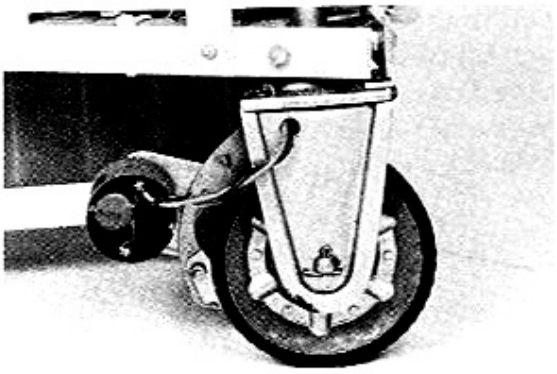


Fig. 2–2. Motorized wheel.

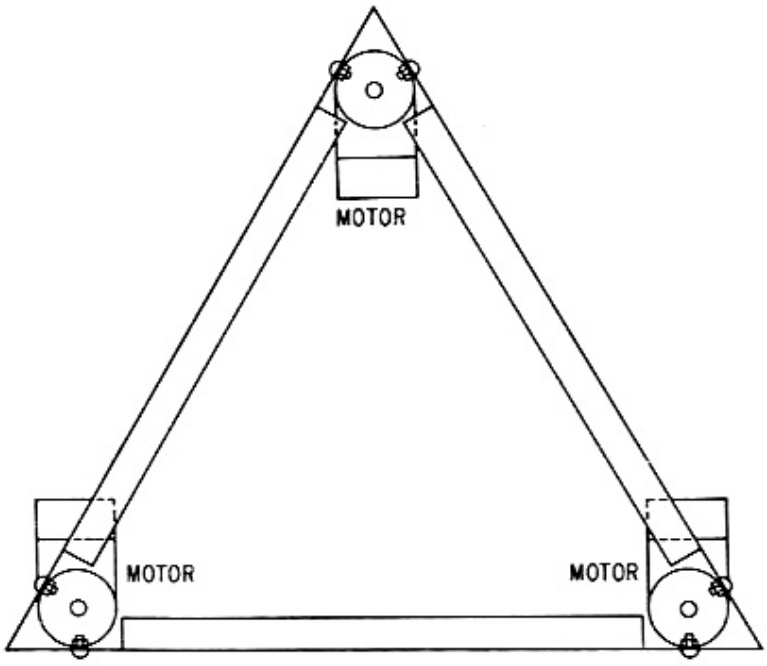


Fig. 2–3. Lower triangle attached to wheels.

Place the front wheel so that its motor faces backward, and place the back two wheels so that their motors face forward. Next, line up the three 23-in. pieces of aluminum with the disks as shown in Fig. 2–3. Make sure that the 16-in. face of each piece of aluminum is turned upward. Drill a $^3/_{16}$-in. hole through the aluminum and the disk in all six places, as shown in Fig. 2–3. Bolt the aluminum to the disks. The order of the items that make up each bolt should be as follows: head of $^3/_{16}$- by ½-stove bolt, flat washer, aluminum, disk, flat washer, lock washer, nut.

The nut should be held with a wrench as the bolt is tightened. The aluminum is bolted to the disks, and the lower triangle is now completed.

Because of his triangular shape, Mike requires only one steerable wheel. Therefore, the back two wheels must be locked in place. Turn the back wheels so that the tires are facing straight forward. (The motors should still be facing front.) Then take a small flat piece of aluminum, approximately $2\frac{1}{8}$ in. long and $\frac{7}{8}$ in. wide. (This can be cut from your angle aluminum by cutting at the vertex of the angle.) Put slightly less than $1\frac{1}{2}$ in. of it in a vise and pound the remaining $\frac{5}{8}$ in. over to one side until you have formed a 90° angle. Next locate the large square plate that is directly below the disk on the wheel. Drill a $\frac{3}{16}$-in. hole in this plate about 1 in. to the right of the disk-mounted stop. Take the piece of aluminum you have bent and put the $1\frac{1}{2}$-in. side flush against the back piece of aluminum. The $\frac{5}{8}$-in. side should extend over the hole in the plate of the wheel. Mark the place where the hole comes into contact with the aluminum and drill another $\frac{3}{16}$-in. hole in the aluminum. Now bolt the aluminum to the plate. (When I use the term "bolt," I mean use a $\frac{3}{16}$- by $\frac{1}{2}$-in. stove bolt on one side of the bolted material and a lock washer and nut on the other side.) Drill one more $\frac{3}{16}$-in. hole through both pieces of aluminum (the bent piece and the back piece of the triangle) and bolt them together. Do the same thing to the other back wheel.

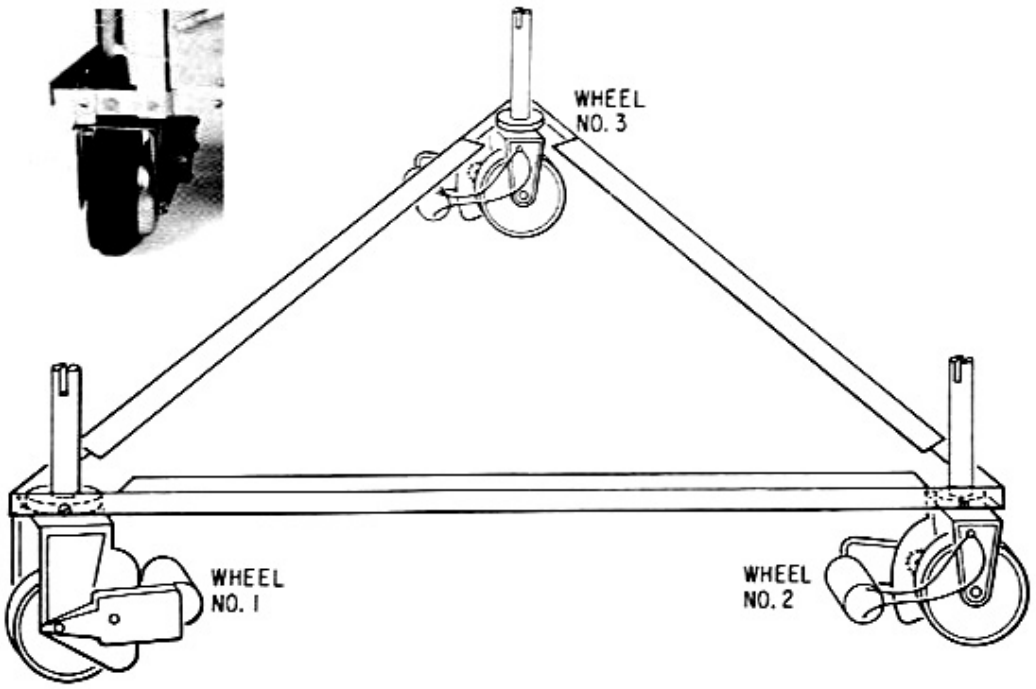


Fig. 2–4. The assembled lower triangle.

upper triangle

The next stage in Mike's construction is the completion of the triangular framework. Because the shafts of the wheels are set loosely in the disks, the wheels tend to wobble. To eliminate this problem, a second triangular frame is built above the first. This frame not only adds support to the wheels but it also strengthens Mike's skeleton.

The first step in making this frame is to cut six 7-in. pieces of angle aluminum. Then cut a rectangle out of each piece as you did in constructing the lower triangle (Fig. 2–1) but cut only 1 in. down the side of the angle instead of 3½ in. Now bolt two of these pieces to each side of the lower triangle as shown in Fig. 2–5. The 5-in. side of each piece of angle aluminum should be facing to the right.

Fig. 2–5. Aluminum bolted to each piece of lower triangle.

The next step in constructing the upper triangle is to build the triangle itself. Cut three 23-in. pieces of angle aluminum. Take each piece and hold it so that only one face of the angle is visible to you. Now, starting at the vertex of the angle, draw a line 30° to the face of the aluminum that you cannot see. Do the same on the other end of the aluminum. Cut along both lines (Fig. 2–6). Repeat the process on your

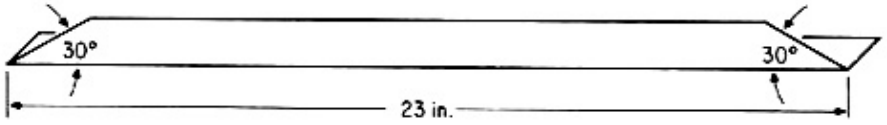


Fig. 2–6. Aluminum side of upper triangle.

other two 23-in. pieces of aluminum. Now cut an equilateral triangle, 23 in. on a side, out of ½-in. plywood. Cut off the corners into three equilateral triangles, 6 in. on a side. Save the remaining piece of wood

because it will be used later. In the meantime, take the 6-in. triangles and cut them down to 5¼ in. on a side. Form a triangle using your three 23-in. pieces of aluminum. Then shove the three plywood triangles into the corners. Obtain 12 wood screws and find a drill the same diameter as the shaft of each screw, less the threads. Using this drill, make 12 holes through the aluminum and the wood as shown in Fig. 2–7. Now locate a

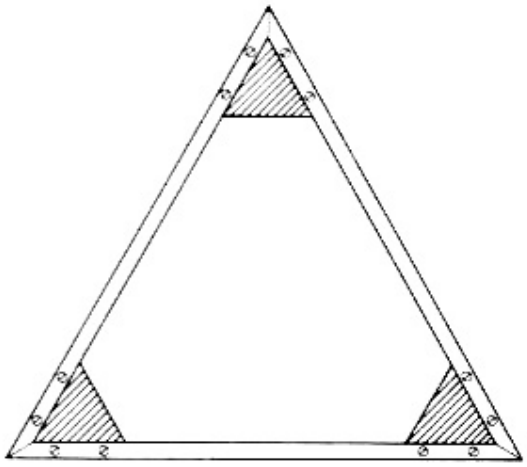


Fig. 2–7. Upper triangle mounted to wood with wood screws.

drill with a diameter equal to the entire shaft of each screw, including the threads. Enlarge the holes in the aluminum with this new drill. Do not, however, drill into the wood. When this is done, place the screws through the aluminum and screw them into the wood. The upper triangle is now completed.

The last step in constructing the triangular frame is to bolt the upper triangle to the rest of the frame. To do this, lay the upper triangle on the six 7-in. braces that you bolted to the lower frame. The wooden triangles should rest on the shafts of the wheels. Straighten the shafts of the wheels in their disks and mark the spots where they hit the wooden triangles. Then remove the upper triangle. Using a ¾-in. drill, make holes through the wood where it was touching the shafts of the wheels. You may have to remove the wood from the upper triangle in order to drill the holes. Place the upper triangle back on the frame, sliding the shafts of the wheels through the holes in the wood. If the shafts do not fit in the holes, sand the holes a little until they do. The triangle should be on the inside of the braces, resting on their inward-pointing faces. Next, drill a 3/16-in. hole through each brace and through the point where it hits the upper triangle, and bolt the upper triangle to the braces. Now Mike's basic inner skeleton is finished, as shown in Fig. 2–8.

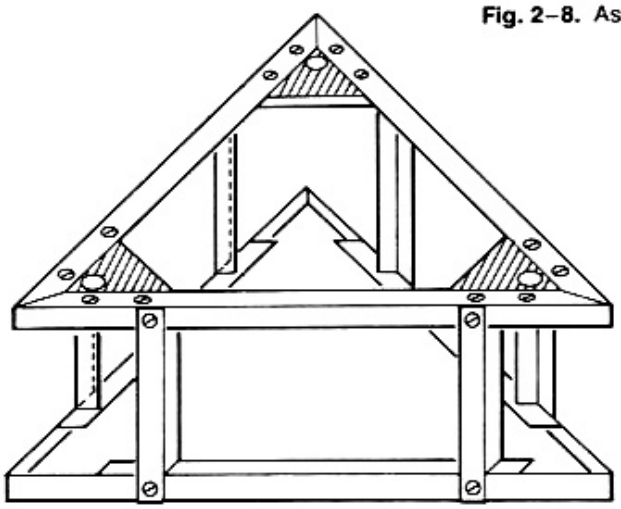


Fig. 2–8. Assembled frame.

battery cage

The third step in Mike's construction is the building of a housing for his power supply, a 12-V 84-A-hr car battery. It is necessary that you get such a battery and a charger for it. The battery that I used has a base measuring 6¾ in. by 10 in. If your battery has a different base size, you will have to modify the instructions to accommodate it. I suggest that you read this section carefully so that you understand the basic concepts of the battery cage's construction.

In addition to a battery and a charger, you will need 38 in. of screw rod. The screw rod can be anywhere from ¼ in. to ½ in. in diameter. Cut the rod into four pieces, each 9½ in. long. Using 1- by 1- by ⅛-in. angle aluminum, cut four pieces, two 6¾ in. long and two 12 in. long. Arrange these pieces in a rectangle, as in Fig. 2–9. Your battery should be able to

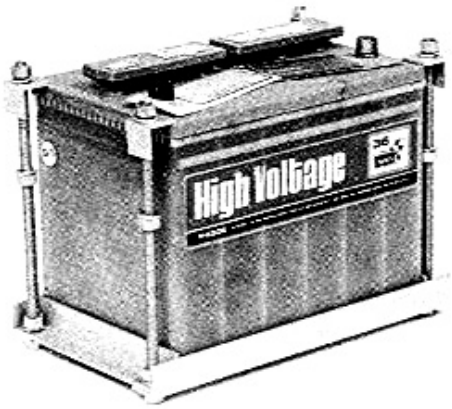


Fig. 2–9. Arrangement of aluminum to form battery cage.

fit into the rectangle. Next, drill a hole large enough to accommodate your screw rod in each of the four places where the aluminum overlaps. Push the four pieces of screw rod through these holes so that they extend about 1/2 in. below each hole. Screw a nut onto the screw rod below the aluminum and wind a lock washer and a nut above it, tightening these until the rod is locked rigidly in place.

Take the rectangular assembly that you have just constructed and align one of the 6 3/4-in. sides directly under the back of the triangular frame. (You will have to raise the frame above the ground in some manner, possibly by placing it on blocks.) Center the battery cage so that the two posterior pieces of screw rod (the ones under the lower triangle) are equidistant from the back wheels. Then mark the spots where they touch the horizontal face of the angle aluminum. Drill a hole big enough to accommodate the screw rod at each of these two spots.

To mount both ends of the battery cage, you need a piece of angle aluminum 10 in. long. Cut one face of the piece in the same way you cut the braces of the triangular frame. One face should now be 8 in. long, the other 10 in. With the 8-in. side facing you and pointing down, cut the 10-in. side as shown in Fig. 2–10. The 2-in. section removed from the piece of aluminum allows room for the directional control motor.

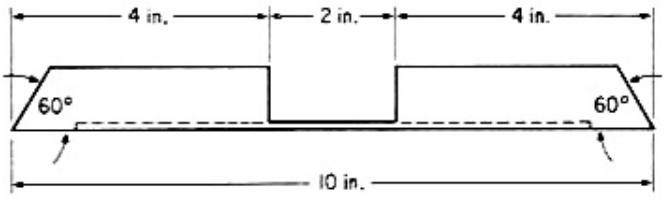


Fig. 2–10. Ten-inch face of aluminum for mounting battery cage. Dotted line shows position of eight-inch face.

Locate the piece of aluminum that forms the back of the lower triangle. Now, holding your newly made 10-in. aluminum piece as seen in Fig. 2–10, lay it on the lower triangle so that it is parallel to this back side. The piece should be resting on all sides of the lower triangle except the back one. Make sure that an object 10 in. long can just barely fit between this piece and the back side of the lower side of the triangle.

To mount the battery cage in the frame, raise the frame above the ground. Then place the battery cage so that two pieces of screw rod are in the holes that you drilled for them in the back of the lower triangle. The two other lengths of screw rod should touch the bottom of the 10-in. piece of aluminum. Mark the spots where they touch and drill holes large enough to accommodate your screw rod at each of these two places. Now place the battery in the cage and twist a nut and a lock washer onto each

piece of screw rod, stopping about 3½ in. from the top of the rod. Push the screw rod through its four designated holes and twist a nut on each piece, turning it until the battery cage is locked solidly in place. The bottom of the battery cage should ride about 2 in. above the ground, as shown in Fig. 2–11.

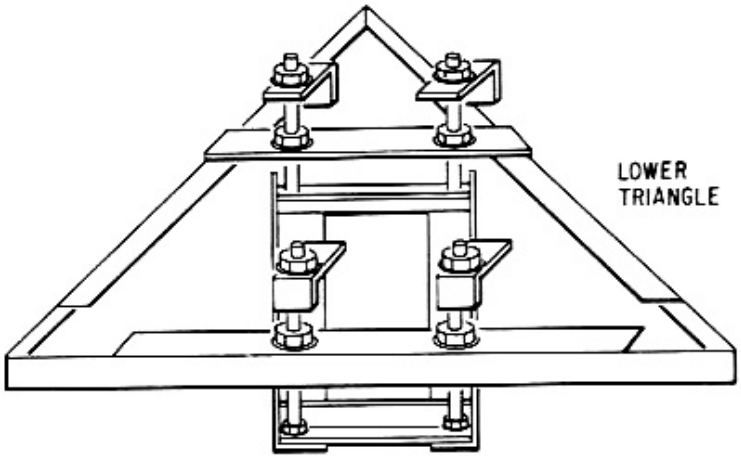


Fig. 2–11. Battery cage positioned within the frame.

The only operation remaining in the construction of the battery cage is the anchoring of the battery itself. This anchorage is accomplished by cutting four pieces of angle aluminum, each 1 in. in length. Drill a hole large enough to accommodate your screw rod in one face of each piece. To determine the position of the hole, place one of the 1-in. pieces flat on top of each piece of screw rod. Push each piece up against the rod so that both faces of the angle touch the rod. Mark the spot where the top of each rod touches the aluminum and drill your hole at that point.

After drilling the holes in the 1-in. pieces, slide them onto the four screw rods, as shown in Fig. 2–12. Then put a lock washer and a nut on top of each piece and screw the nuts down until tight. The battery should now be locked rigidly in place within the frame.

directional control assembly

The directional control assembly consists of a motor for rotating the front wheel and a gearing arrangement to slow down the speed of rotation to a reasonable level. To construct the assembly, you must obtain two gears. (These should be available at a local gear or machine shop.) The gear ratio between the two gears should be approximately

Fig. 2–12. Anchoring battery in cage.

10:1. In other words, the larger of the two gears should have 10 times as many teeth as the smaller gear. If possible, the larger gear should be about 5 in. in diameter, although it can vary from 3½ in. to 6½ in. The larger gear should have a hole not larger than ¾ in. in its center.

The larger gear has to fit on the shaft of the front wheel. Therefore, it is necessary that you drill a ¾-in. hole in the center of the gear. If you do not have the proper equipment to drill a ¾-in. hole in your gear, you may be able to have it drilled at a local machine shop. Remove the upper triangle from the triangular frame and slide your large gear onto the shaft of the front wheel. Slide the gear as far as it can go, until it hits the spot on the shaft where two wires protrude. Once the gear is in place, the upper frame can be bolted back as before.

On the output shaft of the directional control motor is a pulley. Remove the pulley by opening the gearbox and dislodging the output shaft. Then the output shaft can be pulled out and the pulley removed. After replacing the output shaft, drill a 3/16-in. hole in the center of the smaller gear. Slide it onto the output shaft until the top of the shaft is flush with the top of the gear. The gear should be difficult to slide on the shaft and may have to be pounded on with a hammer. Secure the gear on the output shaft with a drop of instant bonding glue.

The last step in constructing the directional control assembly is the mounting of the geared-down motor. The motor comes mounted in a black plastic gearbox. You will notice that out of one end of the black plastic gearbox extends a small cylindrical piece of plastic. This piece of

plastic interferes with the mounting of the motor and should be cut off. Cut a piece of 1 1/2- by 1/8-in. flat aluminum 7 in. long, and cut each end at a 60° angle. At the same time, cut two parallelograms, each 1 in. wide (Fig. 2–13). Locate the center of your 7 in. piece and cut out a U-shaped section 1 in. deep and 3/4 in. wide around the center. Now turn your directional control motor so that the small gear is facing away from you and the motor is facing toward you. Notice that the two screw holes protrude to the sides. Hold the 7-in. piece of aluminum facing as shown in Fig. 2–10 and place it on the black plastic gearbox. The small gear should be within the U-shaped opening, and the piece of aluminum should cover the two screw holes. Mark the places where the aluminum covers the screw holes and drill a 3/16-in. hole at each spot. You may also have to enlarge the screw holes in the gearbox with the drill. Now bolt the aluminum to the directional control motor.

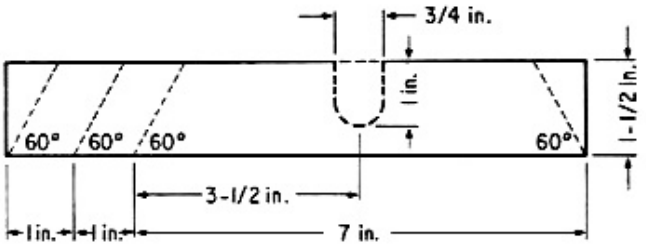


Fig. 2–13. Flat aluminum to be used for mounting directional control motor. Cut on dotted lines.

Glue one of the parallelograms, which you have cut previously, under each end of your 7-in. piece. Now place the 7-in. piece toward the front end of the lower triangle, making sure that it is parallel to the back of the triangle. Slide the aluminum (and the directional control motor) forward until the small gear meshes with the large gear. Then drill a 3/16-in. hole through each end of your 7-in. piece and through the side of the lower triangle. Now that the directional control motor is mounted firmly in place, Mike's inner structure is complete.

chapter three power supply, speed control, and directional control circuits

Power Supply Circuit Parts List:

Perf board, 4 1/2 in. by 8 1/2 in.
1 10 A 100 V diode
2 3 Ω 11 W resistors
1 two-pin connector
6 fuses
- 1 7 A slow blow
- 1 20 A slow blow
- 1 8 A slow blow
- 1 1 A
- 2 1.5 A

7 capacitors
- 2 13,000 μF 15 V electrolytic
- 1 1000 μF 15 V electrolytic
- 4 0.1 μF 50 V disc

2 MC 7805 regulators (TO–3 cases)
2 heatsinks for 7805s (see text)
1 SPDT switch
3 1/2 inches of 1- by 1- by 1/8-in. angle aluminum
4 cable ties
6 3/16- by 1/2-in stove bolts
6 3/16-in. nuts
6 3/16-in. lock washers

Speed Control Circuit Parts List:

Perf board, 4 1/2 in. by 8 1/2 in.
1 SPDT switch
13 resistors
- 1 3.9 kΩ 1/2 W
- 2 4.7 kΩ 1/2 W
- 1 27 kΩ 1/2 W
- 1 33 kΩ 1/2 W
- 1 3.6 kΩ 1/2 W
- 1 330 Ω 1/2 W
- 1 390 Ω 1/2 W
- 1 470 Ω 1/2 W
- 1 560 Ω 1/2 W
- 1 47 Ω 1 W
- 1 10 Ω 10 W
- 1 1.5 Ω 1/2 W

8 transistors
- 4 RS 2012 (Radio Shack)
- 2 HEP S0019 (Motorola)
- 1 RS 2006 (TO-3 case) (Radio Shack)
- 1 SK 3036 (TO-3 case) (RCA)

2 heatsinks
- 1 for RS 2006 (see text)
- 1 for SK 3036 (see text)

2 diodes
- 1 1 A 100 V
- 1 10 A 100 V

1 relay (Radio Shack No. 275-208)
2 capacitors
- 1 0.001 μF 50 V disc
- 1 0.1 μF 50 V disc

6 3/16- by 1/2-in. stove bolts
6 3/16-in. nuts
6 3/16-in. lock washers
4 1/2 inches of 1- by 1- by 1/8-in. angle aluminum

Directional Control Circuit Parts List:

Perf board, 4 1/2 in. by 6 1/2 in.
13 inches of 1 1/2- by 1/8-in. flat aluminum
8 3/16- by 1/2-in. stove bolts
8 3/16-in. nuts
8 3/16-in. lock washers
12 resistors
- 2 150 Ω 1/2 W
- 2 330 Ω 1/2 W
- 2 3.3kΩ 1/2 W
- 2 4.7kΩ 1/2 W
- 2 27kΩ 1/2 W
- 2 33kΩ 1/2 W

6 transistors
- 4 RS 2012 (Radio Shack)
- 2 RS 2006 (Radio Shack)

2 1 A 100 V diodes
2 relays (Radio Shack No. 275-206)

To provide Mike with his mobility, three main circuits must be built. The first of the three, the power supply circuit, provides 5 V for Mike's other circuitry and his microprocessor or "brain." The circuit also allows for connecting a battery charger when Mike's battery runs low on power. The speed control circuit permits the microprocessor to control the operation of the motorized wheels. It contains a relay for reversing the motors to provide both forward and backward motion. The directional control circuit provides the microprocessor with the ability to rotate the front wheel. By activating either the "turn left" or the "turn right" portion of the circuit, the microprocessor can make Mike move in almost any direction.

power supply circuit

The power supply circuit is designed to output 5 V for use by the microprocessor and by the logic circuitry (Fig. 3–1).* The circuit uses two 7805 regulators, which are designed to produce +5 V from an input of +7 V to +35 V. Make sure that the 7805s that you buy are in TO–3

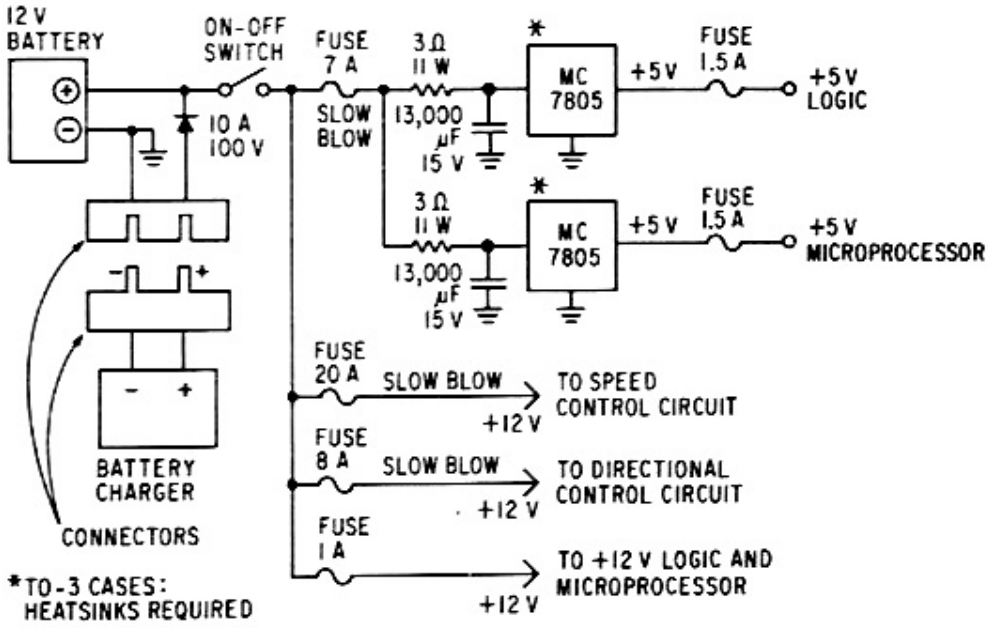


Fig. 3–1. Schematic of power supply circuit.

*An additional 1,000 μF @ 15 V is needed at +5 input at microprocessor. Also each motorized wheel and the directional control motor should have 0.1μF @ 50 V bypass at terminals of motor.

cases. Also try to get heatsinks that are roughly cubical in shape. The 3 Ω 11 W resistors shown in the circuit diagram reduce the +12 V from the battery to +9 V in order not to overburden the regulators.

The power supply circuitry should be built on a piece of perforated (perf, for short) board approximately 4½ by 8½ in. Three parts of the circuit are not mounted on this piece of perf board. The two 13,000 μF capacitors are tied by cable to the 7-in. braces of the triangular frame on either side of the back left wheel. The capacitors are placed in the angle of each brace and tied securely. Another part of the circuit that is not mounted on the perf board is the "on–off" switch. This switch is positioned on the leg of the lower triangle that connects the back left wheel with the front wheel, about 8 in. from the front of this piece of aluminum. It is mounted on the horizontal face of the aluminum. Wires are run from the switch to the power supply circuit. The third and last part of the circuit that is not mounted on the perf board is the connector for attaching the battery charger. The connector is mounted on the upper triangle, directly above the place where the "on–off" switch is positioned on the lower triangle. Unlike the switch, the connector is mounted on the vertical face of the aluminum.

A few points are essential in the layout of the power supply circuit; otherwise, its design is up to you. Place the piece of perf board flat on a table with the 8½-in. side facing you. The two 7805 regulators should be mounted approximately in the center of the perf board, one heatsink ½ in. above the other (Fig. 3–2). On both sides of the regulators go the fuses in fuse holders, the diode, and the two resistors.

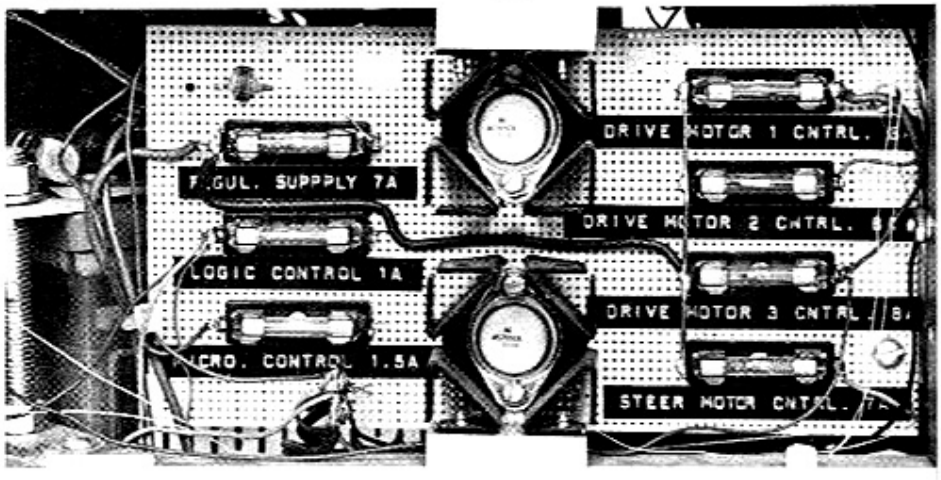


Fig. 3–2. Power supply circuit mounted on frame.

The heatsinks on the power supply circuit facilitate its attachment to Mike's triangular frame. Cut two pieces of 1- by 1- by $\frac{1}{8}$-in. angle aluminum, each $1\frac{3}{4}$ in. long. Now bolt one face of one of the pieces to the top edge of the upper heatsink and one face of the other piece to the bottom edge of the lower heatsink. Use two bolts to anchor each heatsink. The other face of each piece should be flush with the front of the heatsinks. Place the circuit in between the upper triangle and the lower triangle on the left leg of the frame (Fig. 3–3). Keep the circuit as close as possible to the back left wheel. Now bolt the angle aluminum attached to the heatsinks to both levels of the triangular frame.

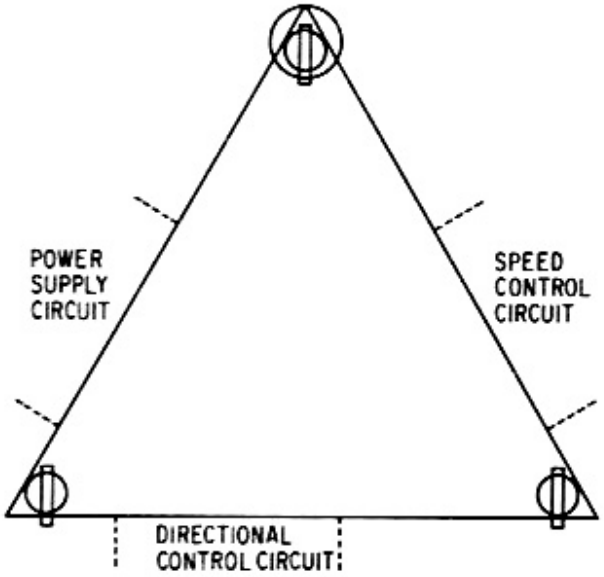


Fig. 3–3. Position of three main circuit boards.

speed control circuit

The speed control circuit consists of two main parts (see Fig. 3–4). The first part, the forward–reverse control, amplifies the input from the microprocessor so that it can control the direction of the motorized wheels through the relay. If the relay is in the released condition, Mike moves backward. However, if the relay is operated by the microprocessor, Mike moves forward. The second main part of the circuit, the speed control, allows Mike to control the speed of his motorized wheels. This portion of the circuit amplifies the logic level of the micro to drive the main power transistor, the SK 3036. The microprocessor can change the speed of Mike's motorized wheels by varying the number of "motor on" and "motor off" cycles. Every millisecond or ms (1/1000 of a second) the

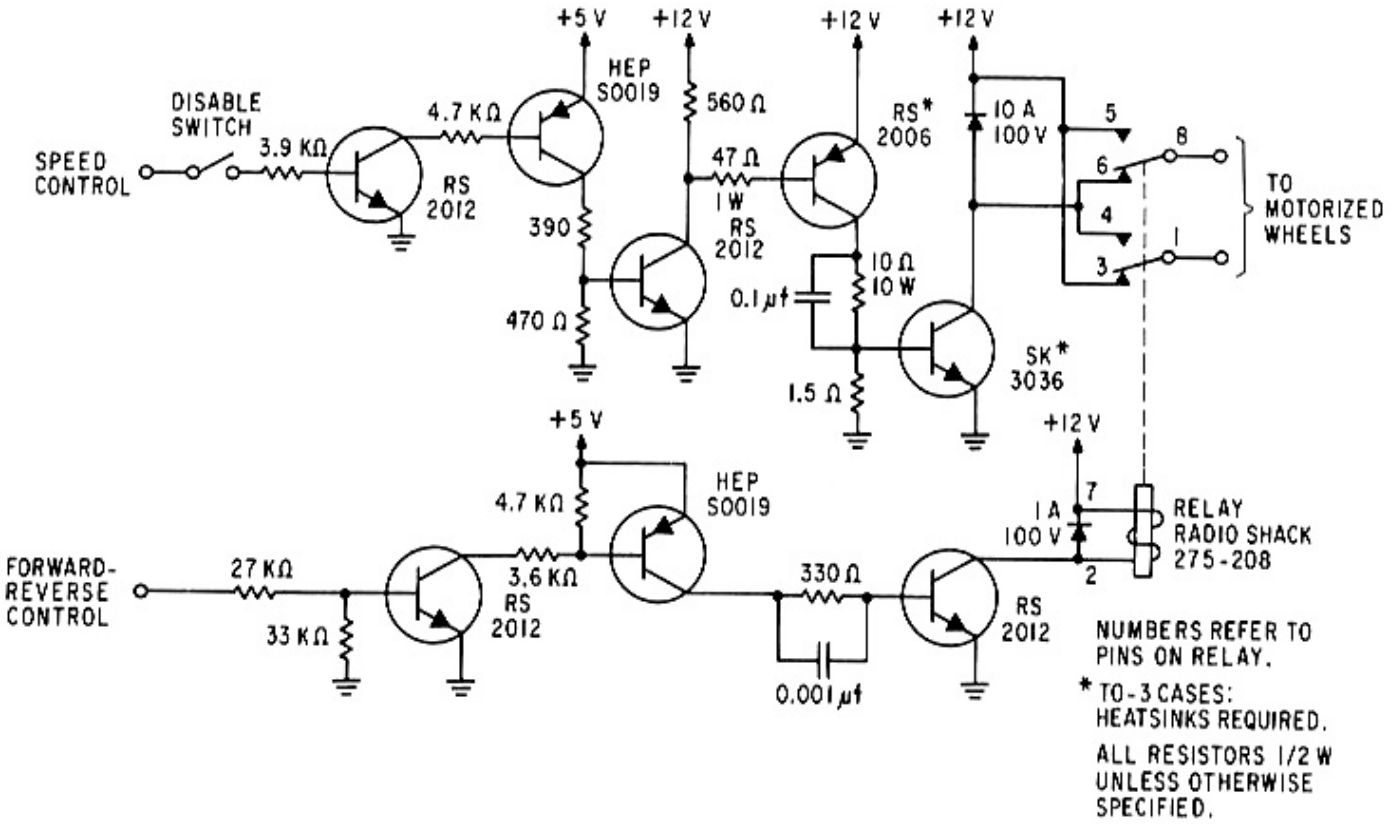


Fig. 3–4. Schematic of speed control circuit.

computer checks to see whether the speed control portion of the circuit (and consequently the motorized wheels) should be "on" or "off." By changing the proportion of "motor on" to "motor off" cycles, Mike can be made to move at different rates of speed. For example, slow speed calls for the speed control to be activated one out of every 10 ms. For medium slow speed, the motors are on for two out of every 10 ms. Top speed is achieved by having the motor on all the time. The role of the microprocessor in controlling the speed control circuit will be explained in greater detail in the chapter on software.

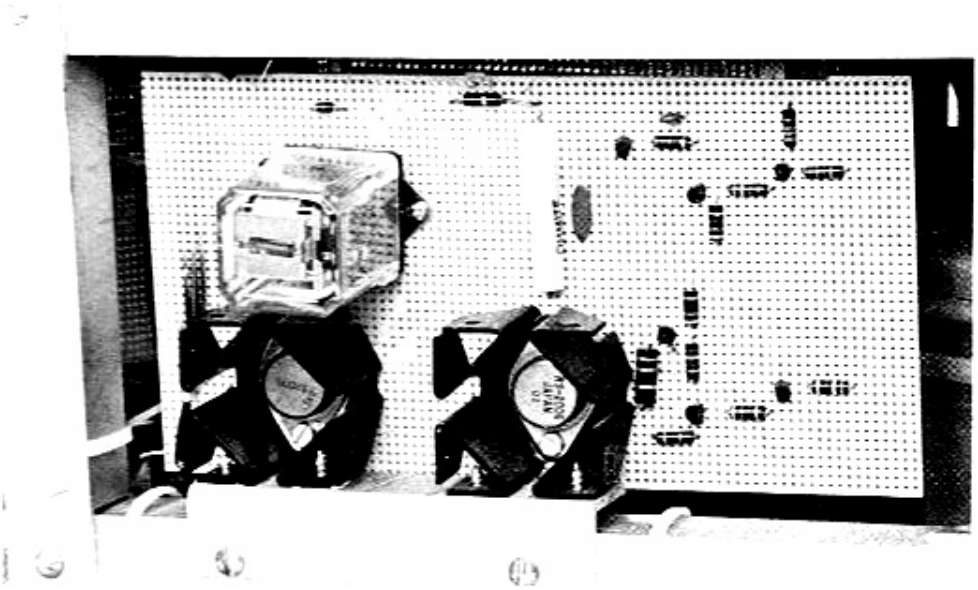


Fig. 3–5. Speed control circuit mounted on frame.

A few points are essential in the construction of the speed control circuit (see Fig. 3–5). Only two transistors in the circuit require heatsinks, the SK 3036 and the RS 2006. Both transistors should use the same kind of heatsink used for the 7805s in the power supply circuit, the type that facilitated its mounting to the triangular frame. The speed control circuit is built on a 4½- by 8½-in. piece of perf board. The SK 3036 should be placed about 1 in. from the right side of the perforated board. The circuit is mounted on the triangular frame by cutting a piece of angle aluminum 4½ in. long and bolting one face of it to both heatsinks and the other face to the vertical face of the lower triangle. As with the power supply circuit, use two bolts to anchor each heatsink. Use two bolts also to anchor the aluminum to the frame. The speed control circuit should be mounted on the leg of the lower triangle connecting the

back right wheel with the front wheel, as shown in Fig. 3–3. Locate it between the upper and lower triangles.

The 10 A 100 V diode in the speed control circuit prevents transients generated by the motorized wheels from destroying the SK 3036. Originally, when I constructed the circuit, I used a 1-A diode in place of the 10-A diode. A sizzling noise and a large cloud of smoke soon informed me that a larger diode was required. I still can't believe the amount of smoke one tiny diode can produce. I replaced the old diode with the present 10-A diode, but my perf board still bears the mark of my disaster.

The only other essential point in constructing the circuit is the mounting of the disable switch. This switch disables the speed control circuit. It should be positioned in the center of the back leg of the upper triangle. Make sure that you mount it on the vertical face of the aluminum.

I have found the disable switch to be invaluable for testing Mike. It has also been very useful as a fail-safe device when debugging software.

directional control circuit

The last of the three main circuits, the directional control circuit, is simple in concept (see Fig. 3–6). Essentially, the circuit contains two separate circuits, one for turning right and the other for turning left. A signal from the microprocessor on either the "turn left" or the "turn right" line is amplified and sent to the proper relay. The front wheel is rotated for the duration of the signal. If somehow both the "turn left" and the "turn right" lines are activated, the directional control motor is disabled temporarily to prevent a short.

The entire directional control circuit is built on a 4½- by 6½-in. sheet of perf board (see Fig. 3–7). When mounting components on the board, leave a ½-in. border on all sides. The 4.7 Ω 10 W resistor, shown in the circuit, limits the current going to the directional control motor in order to slow down the speed of the motor. The resistor dissipates a large amount of heat and therefore should be mounted so that it is not touching the perf board. To prevent scorching the board and to allow for proper heat dissipation, leave about ¼ in. between the resistor and the board.

None of the transistors requires heatsinks because of the circuit's low power consumption. Therefore, the circuit must be mounted by means other than heatsinks. Cut two pieces of 1½- by ⅛-in. flat aluminum 6½ in. long and lay them 3½ in. apart on a table. Now place the directional control circuit, component side up, on top of both pieces. The circuit should overlap ½ in. on each piece. Drill two holes through each

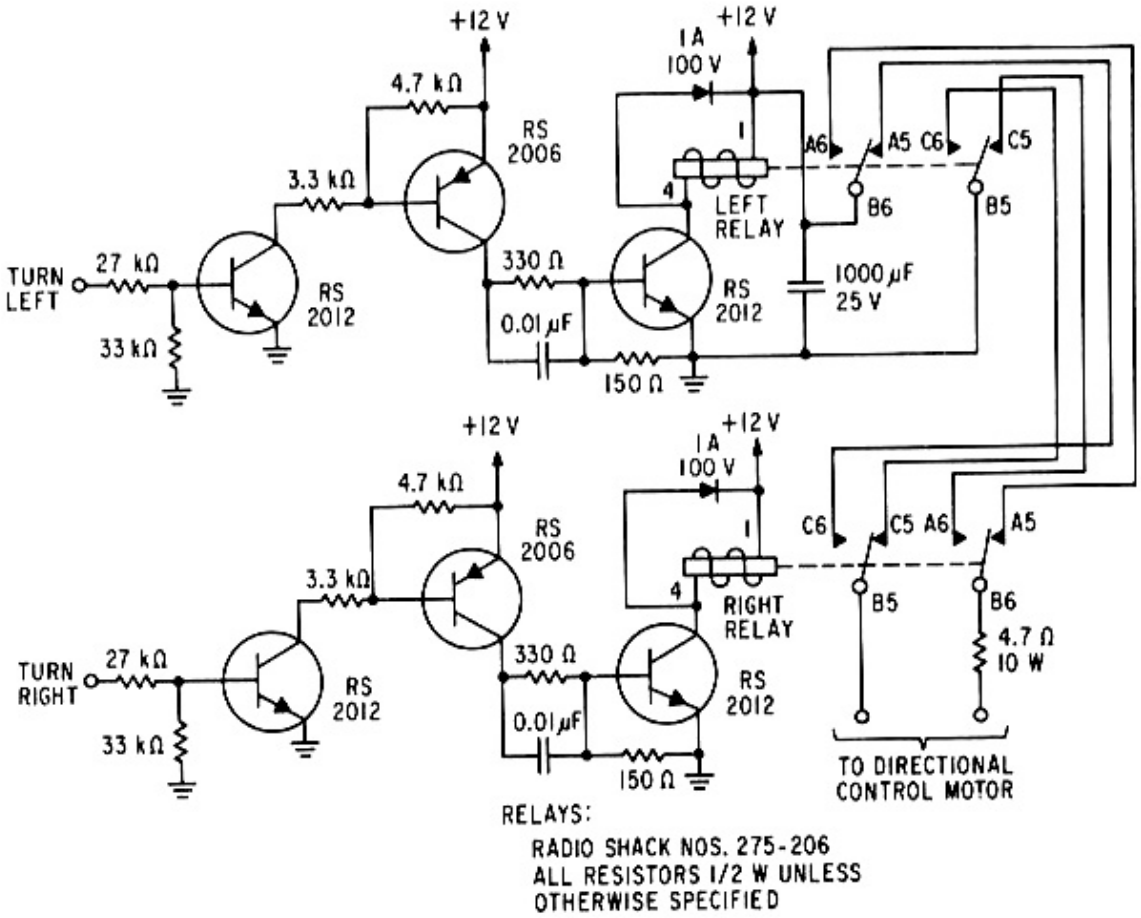


Fig. 3–6. Schematic of directional control circuit.

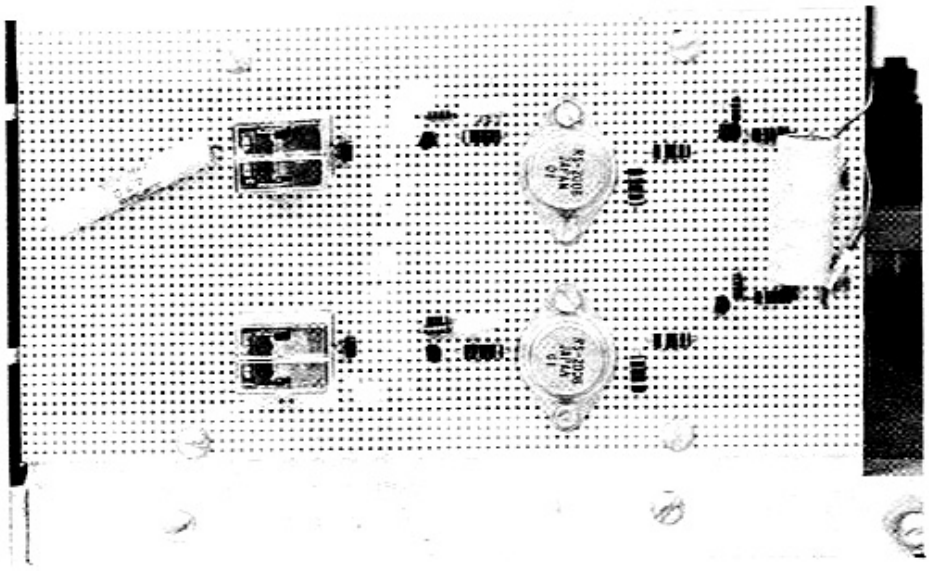


Fig. 3–7. Directional control circuit mounted on frame.

end of the perf board and the place where it overlaps the aluminum. Bolt the circuit to the aluminum. Next place the circuit with the aluminum attached up against the back of the triangular frame (Fig. 3–3). The $6\frac{1}{2}$-in. pieces of flat aluminum should make contact with the back legs of the upper and lower triangles. Drill two holes in each piece of flat aluminum at the points where it contacts the upper or lower triangular frame. Now bolt the flat aluminum to the frame. Mike's three main circuits are now complete.

chapter four secondary circuits, the microcomputer, and software

Parts List:

1 Kim–1 microcomputer (see text)
6 3/16- by 3/4-in. stove bolts
6 3/16-in. nuts
6 3/16-in. lock washers
7 1/2-in. spacers
7 No. 6 1-in. wood screws
1 joystick with 5 kΩ potentiometers
1 10 kΩ potentiometer
1 inch of wood doweling 5/8-in. in diameter
3 inches of black electrical tape
2 No. 4 1/4-in. wood screws
1 No. 6 1/2-in. wood screw
1 3/8-in. nut
2 inches of 1 1/2- by 1/8-in. flat aluminum
Perf board, 3 1/2 in. by 3 1/2 in.

A/D Circuit Parts List:

Perf board, 4 1/2 in. by 3 1/2 in.
1 LM 339 quad comparator IC
1 14-pin low-profile IC socket
8 resistors (all 1/2 watt)
  1 1kΩ
  3 3kΩ
  2 200 Ω
  2 10kΩ
5 capacitors
  4 .1 μF 50 V electrolytic
  1 10 μF 15 V electrolytic
3 1/2 inches of 1 1/2- by 1/8-in. flat aluminum
8 3/16- by 1/2-in. stove bolts
8 3/16-in. nuts
8 3/16-in. lock washers

Inverter Circuit Parts List:

1 0.1 μF 15 V capacitor disc
1 7404 hex inverter IC
1 14-pin low-profile IC socket

All Mike's basic functions are controlled by a microprocessor. This computer makes it possible to eliminate a vast amount of hardware that would otherwise be required for Mike's construction. The microprocessor serves as his "brain," controlling and overseeing all his functions and reflexes. To provide the computer with the ability to control Mike's movement, several secondary circuits are required. Mike is controlled with a joystick. Commands from the joystick are sensed by the A/D circuit, which allows the micro to compare the commands with the actual position and speed of the wheels. An inverter circuit is included to put Mike's circuitry in an inactive mode whenever the "reset" button on the computer is depressed. The last step in the construction of Stage I is loading the software into the microprocessor. The software causes the computer to follow certain steps for controlling Mike's circuitry. Software allows the micro to interpret commands and convert them into actions.

analog to digital (a/d) and inverter circuits

The A/D circuit allows the microprocessor to determine the position of the joystick and of the front wheel in order to carry out a command (see Fig. 4–1). The circuit uses an LM 339 quad comparator IC. The comparator IC should be mounted in a 14-pin IC socket, low profile if possible. The micro starts its internal timer and sets the output line (PA 0) to "1." The "1" sends out 5 V on the output line, which begins charging the 0.1 μ F capacitor connected from PA 0 to ground. The capacitor charges exponentially toward 5 V (Fig. 4–2). As the capacitor charges, the voltage on pins 7, 9, and 5 of the LM 339 increases. When the voltage across the capacitor (and on IC pin 9) equals the voltage coming from the directional control potentiometer (pot), PA 7 goes to "1." While the capacitor has been charging, the microprocessor has been waiting for PA 7 to go to "1." A "1" on PA 5 or PA 6 is ignored by the micro. When PA 7 goes to "1," the timer is read by the computer. The count on the timer is proportional to the setting of the directional control pot. The microprocessor returns the output of PA 0 to "0" and waits for the capacitor to discharge. After the capacitor discharges, the micro repeats the entire process with the directional control command pot and then with the speed command pot, reading PA 6 and PA 5 respectively.

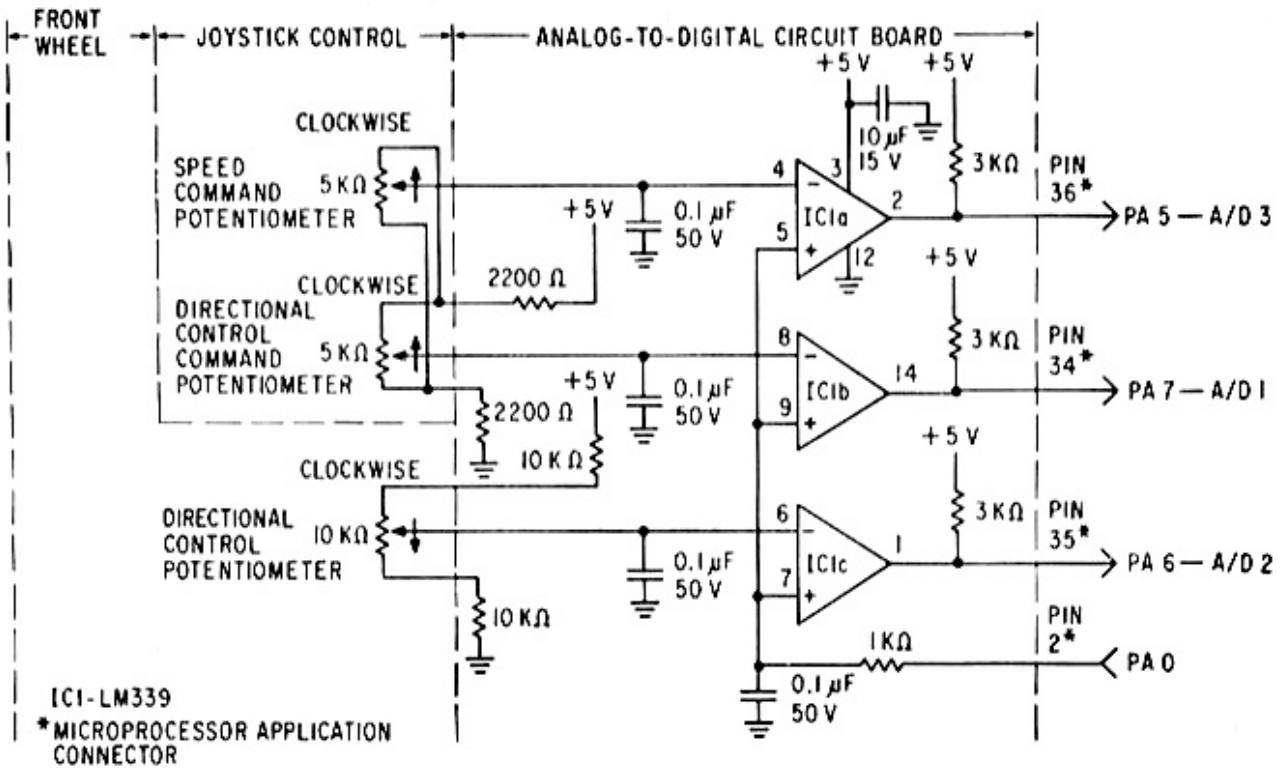


Fig. 4–1. Schematic of analog-to-digital circuit.

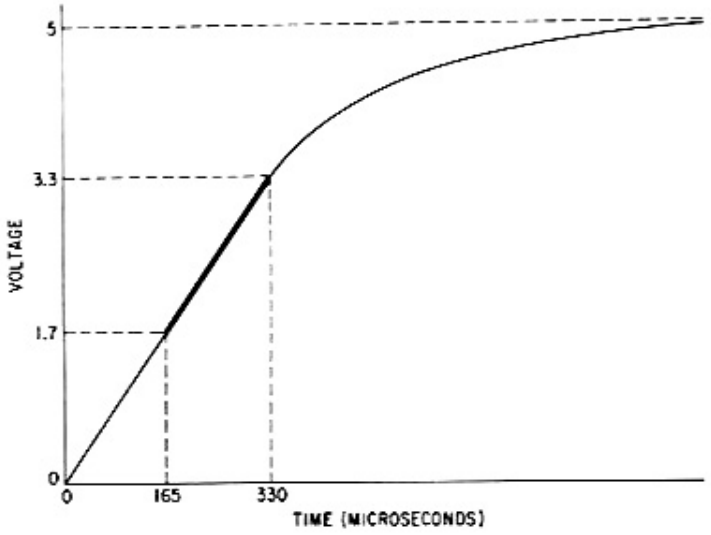


Fig. 4–2. Exponential charge of capacitor.

The A/D circuit should be built on a piece of perf board that is 4½ in. by 3½ in. It should be mounted in the same way as the directional control circuit, on the back leg of the triangular frame. The A/D circuit should be placed to the right of the disable switch.

The Kim–1 microcomputer has one small peculiarity. When the "reset" button is depressed, the Kim sends out logic 1 on the output lines. All Mike's circuits are activated immediately, and he races headlong across the floor. To prevent unsightly holes in walls, add an inverter circuit (Fig. 4–3). The inverter IC causes Mike's circuits to be turned off when "reset" is depressed. The inverter circuit should be built on the same piece of perf board as is the A/D circuit. The 7404 hex inverter IC shown in the circuit is mounted in a 14-pin IC socket, preferably low profile.

joystick and directional control pot

The joystick provides the operator with the ability to control Mike. Try to obtain a joystick with 5 kΩ pots (see Appendix). Mount the joystick on a 3½-in.-square piece of perf board. You may have to cut a hole in the center of the board to make room for the bottom of the joystick. Now select two adjacent potentiometers on the joystick. Use the potentiometer that is farthest away from you and the one that is to your right. One will serve as the speed command pot, the other as the directional control command pot. Hook up the two potentiometers as shown in Fig. 4–1. The small arrow followed by the word "clockwise," shown next to the speed command pot, indicates that as the pot is rotated clockwise

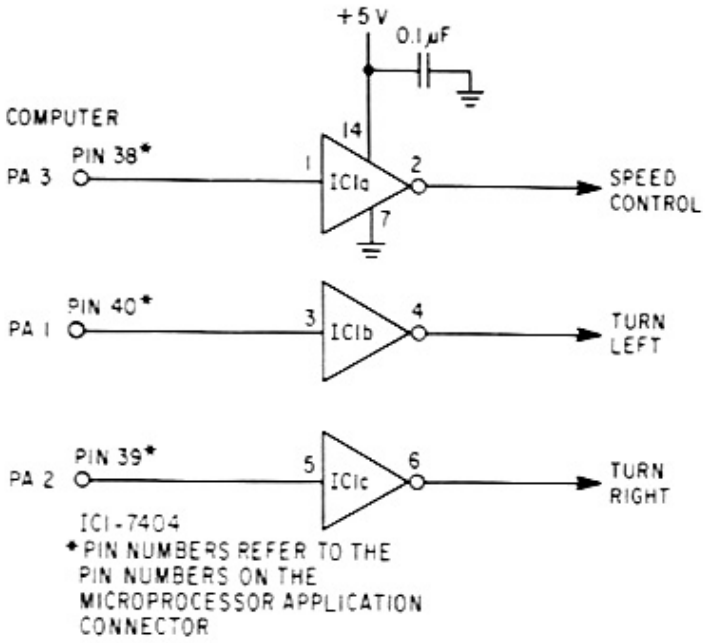
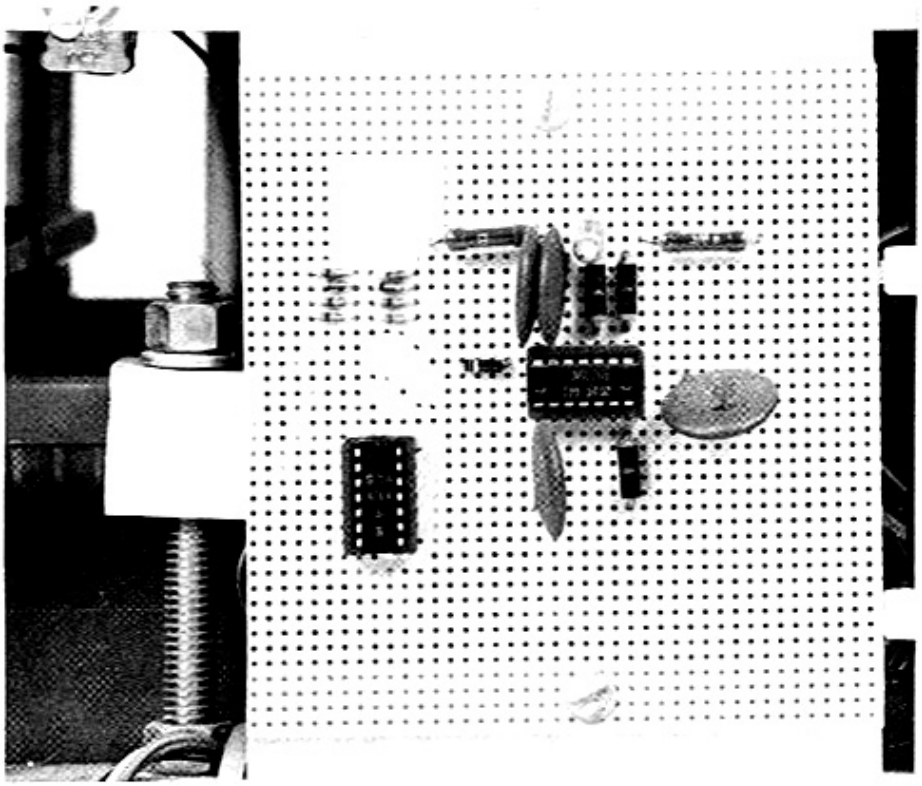


Fig. 4–3. Schematic of inverter circuit.



its voltage approaches +5 V. In other words, the lead of the potentiometer closest to the directional control command pot is connected to the lead of the directional control command pot closest to it and to the 2200Ω resistor on the A/D board that goes to ground. The small arrow followed by the word "clockwise," shown next to the directional control pot, indicates exactly the same as it did for the speed control pot. As the pot is rotated clockwise (view facing the pot's shaft), its voltage approaches +5 V, that is, the lead of the potentiometer farthest from the speed command pot is connected to the lead of the speed command pot farthest from it and to the 5 kΩ resistor on the A/D board that goes to +5 V. Four wires should come from the joystick control—the ground wire, the +5 V wire, and the two wires that connect to the center leads of both potentiometers. Eight feet of four-conductor cable should lead these four wires from the joystick to the A/D circuit. If you wish, you may mount your joystick in a control box.

The directional control potentiometer is a 10 kΩ pot that informs the microprocessor of the position of the front wheel. I used an Ohmite type AB. The pot is mounted on top of the shaft of the front wheel. Drill a hole lengthwise in a 1-in. piece of wood doweling ⅝ in. in diameter, such that the hole can accommodate the shaft of your pot. (Mine was ¼ in. in diameter.)

Place the piece of doweling in the shaft of the front wheel, about ¼ in. from the top. If the doweling does not fit snugly in the shaft, wrap black tape around it until it does. Notice that in the top of the wheel's shaft are two 1-in. slots. Screw a No. 4 ¼-in. wood screw into the wood doweling at the bottom of each of the slots. The screws should be well below the wooden triangle that holds the shaft in place. Cut a 2- by 1-in. piece of aluminum out of your 1½- by ⅛-in. flat aluminum. Now locate the center of one of the 1-in. sides. Draw two angles from this point 60° to the 1-in. side (Fig. 4–4). Cut off the triangles formed by these 60° angles. Locate the point on the aluminum shown by two intersecting lines in the figure. At this point, drill a hole large enough to accommodate the threaded collar of your potentiometer. Slide the 2-in piece of aluminum

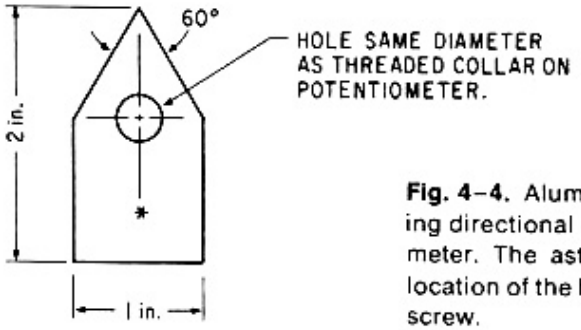


Fig. 4–4. Aluminum for mounting directional control potentiometer. The asterisk shows the location of the hole for the wood screw.

onto the threaded collar of your directional control pot. Make sure that the point formed by the 60° angle is facing in the opposite direction from the leads of the pot. Screw a nut (3/8 in.) onto the pot's threaded collar until it locks the aluminum in place. Turn the pot upside down. Place the shaft of the pot in the hole of the 5/8-in. diameter wood doweling. If the shaft does not fit very snugly, wrap black tape around it until it does.

The 60° angle of the aluminum should fit in the place where the left and right legs of the upper triangle meet. If the angle does not fit, file the aluminum until it does. At the spot marked by an asterisk in Fig. 4–4 drill a hole large enough to start a No. 6 ½-in. wood screw. This hole should be drilled through both the aluminum and the wood below it. Enlarge the hole in the aluminum only, so that the threaded portion of the screw can pass freely through it. Anchor the aluminum with the screw. The directional control pot is now mounted (Fig. 4–5).

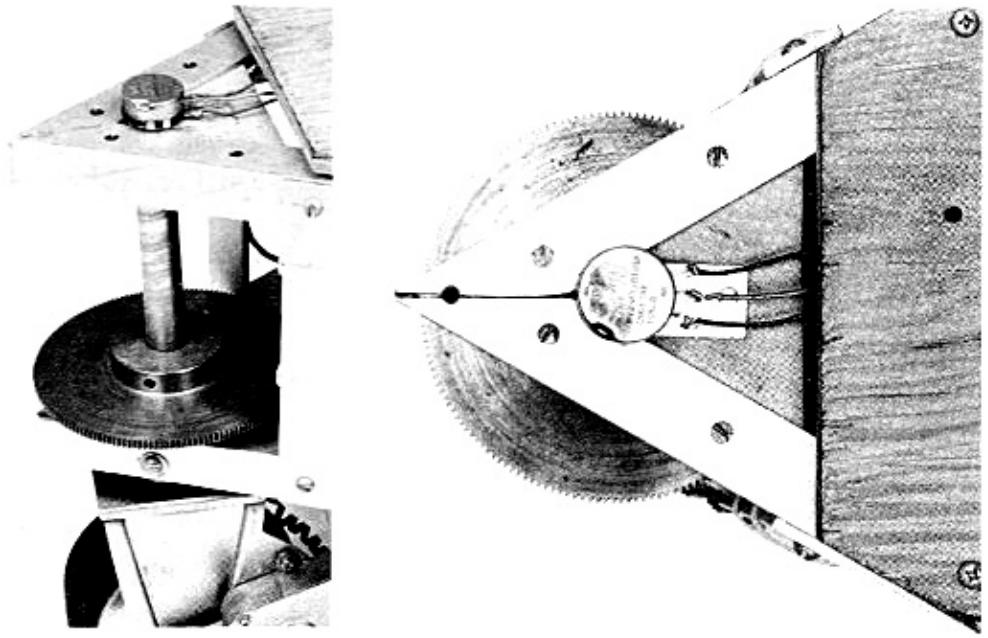


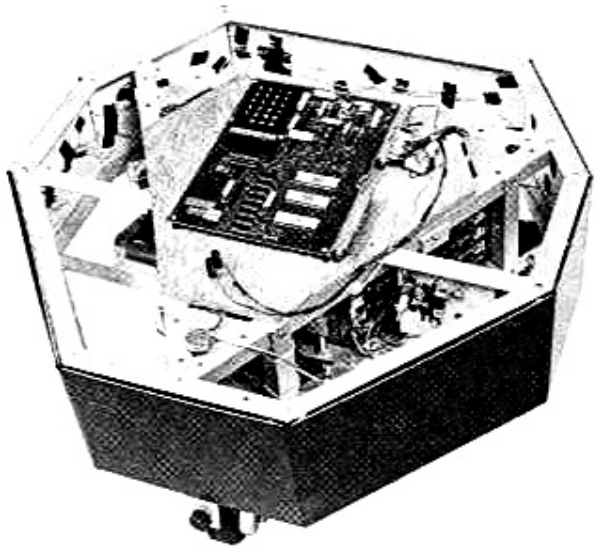
Fig. 4–5. Directional control potentiometer.

The small arrow followed by the word "clockwise" (shown next to the directional control potentiometer) indicates that as the pot is rotated clockwise its voltage approaches ground. In other words, the lead of the pot closest to the right leg of the triangular frame is connected through a 10 kΩ resistor on the A/D circuit to +5 V. The lead closest to the left leg is connected through a 10 kΩ resistor on the A/D board to ground. The center lead is connected directly to the A/D circuit as in Fig. 4–1.

the microcomputer

As previously stated, the microcomputer used to control Mike is a Kim–1 (see Appendix). There are a number of factors that favor the selection of a Kim. One advantage of the Kim is that it is assembled and tested, rather than being in kit form. The Kim–1 is small and lightweight, two crucial advantages for mounting it on the robot. The Kim is also one of the least expensive microcomputers on the market (I purchased it at $245). Another plus is that the Kim comes with a hex keypad and a display built in. Programs can be loaded directly into the Kim without the need of a terminal. In addition to the advantages listed above, the fact that the software in this book is written in 6502 (the language used by the Kim) suggests that you should use a Kim–1 as Mike's brain. The Kim–1 comes with excellent documentation. Carefully read the *Programming and User Manuals* before attempting to use and understand Mike's software.

The Kim is mounted on top of the triangular frame. Take the piece of ½-in. plywood that was left over when you cut out Mike's three wooden triangles. Place the piece of plywood on top of the triangular frame so that its three longest sides are aligned with the sides of the upper triangle. Bolt the plywood to the upper triangle, using two bolts to anchor each of its long sides to the frame. Now hold your Kim so that the keypad is in the lower right-hand corner. Place the computer on the wood so that the 8-in. side of the Kim that is closest to the keypad is parallel to the back leg of the triangular frame. Notice that there are seven mounting holes in the Kim–1. Place a ½-in. spacer under each hole. Screw a No. 6 1-in. wood screw into each of the seven holes until the Kim is anchored solidly to the wood.



By the way, don't forget the footnote on page 21. A 1,000μF 15V capacitor is needed between +5 V (pin A) and ground (pin 1) on the application connector of the microprocessor. The connections to the other pins of the Kim's application connector are shown in Table 4–1.

Table 4–1. Connections to Microprocessor Application Connector

PA	*IN/OUT*	*FUNCTION*	*PIN*
7	In	A/D 1 Comparator	8
6	In	A/D 2 Comparator	7
5	In	A/D 3 Comparator	6
4	Out	Forward/Reverse	5
3	Out	Go/Stop	2
2	Out	Right	3
1	Out	Left	4
0	Out	A/D Capacitor	14

software

Mike's software is written in the form of a program loop. The computer is constantly going through a series of instructions that monitor the joystick and execute its commands. The program is subdivided into four main parts. The first part is the initialization (see Chart 4-1). This portion of the program is begun by setting up the ports as either input or output lines. The motorized wheels are turned off by setting the number of "motor off" cycles to 10 out of every 10 ms. The last function that the initialization routine performs is that of centering the front wheel.

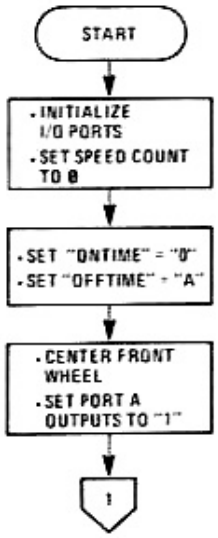


Chart 4–1.

After completing the first section of the program, the computer jumps to the directional control routine (Chart 4-2). This routine determines the setting of the directional control potentiometer. Next, the computer allows the capacitor on the A/D circuit to discharge. The

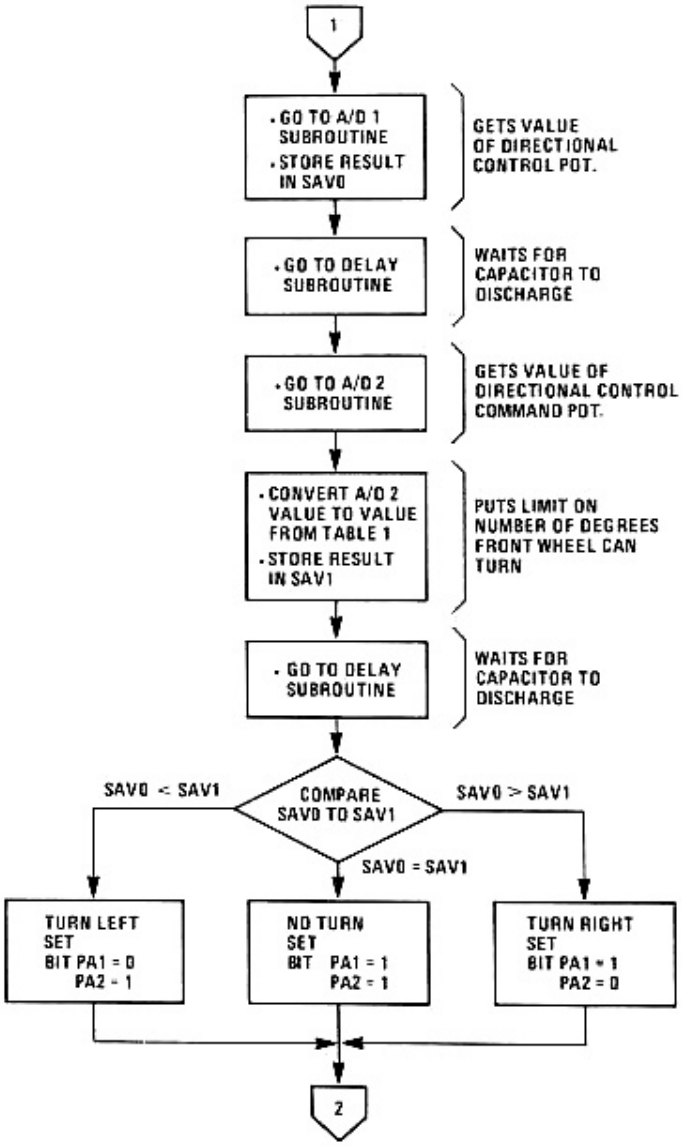


Chart 4–2.

directional control command pot is read to determine its setting, and an input value is obtained. The input value can vary between 0C and 30. The computer compares the input value with each of the table values, starting with 0E (Table 4–2). As soon as the computer finds a table value that is greater than the input value, it takes the corresponding directional control value. The directional control value is compared with the input value from the directional control potentiometer. By converting the input from the directional control command pot to the directional control value, the software prevents the front wheel from rotating more than 60° in either direction. If the directional control value is equal

Table 4–2. Directional Control Table

INPUT VALUE	*TABLE VALUE*	*DIRECTIONAL CONTROL VALUE*	
≤0D	0E	25	Right
0E–12	13	26	
13–17	18	27	
18–1C	1D	28	
1D–21	22	29	
22–26	27	2A	
27–29	2A	2B	
2A–2B	2C	2C	
2C–2D	2E	2D	
2E–2F	30	2E	
30≥	31	2F	Left

to the input value from the directional control pot, the front wheel is not rotated. On the other hand, if the values are unequal, the wheel is turned either left or right.

When the second routine is concluded, the computer jumps to the speed control portion of the program (Chart 4–3). The speed control routine controls the speed of the motorized wheels. Every time the computer cycles through the routine, it decides whether the motorized wheels should be on or off, and the routine allows the computer to execute its decision.

The last part of the program, the speed determination routine, determines the speed of the motorized wheels (Chart 4–4). First, the routine discovers the setting of the speed command potentiometer and determines an input value based on the setting (Table 4–3). If the input value is greater than or equal to 26, the speed control relay is set in forward; if not, it is set in reverse. The computer compares the input value with each of the table values, starting with 0F. As soon as the

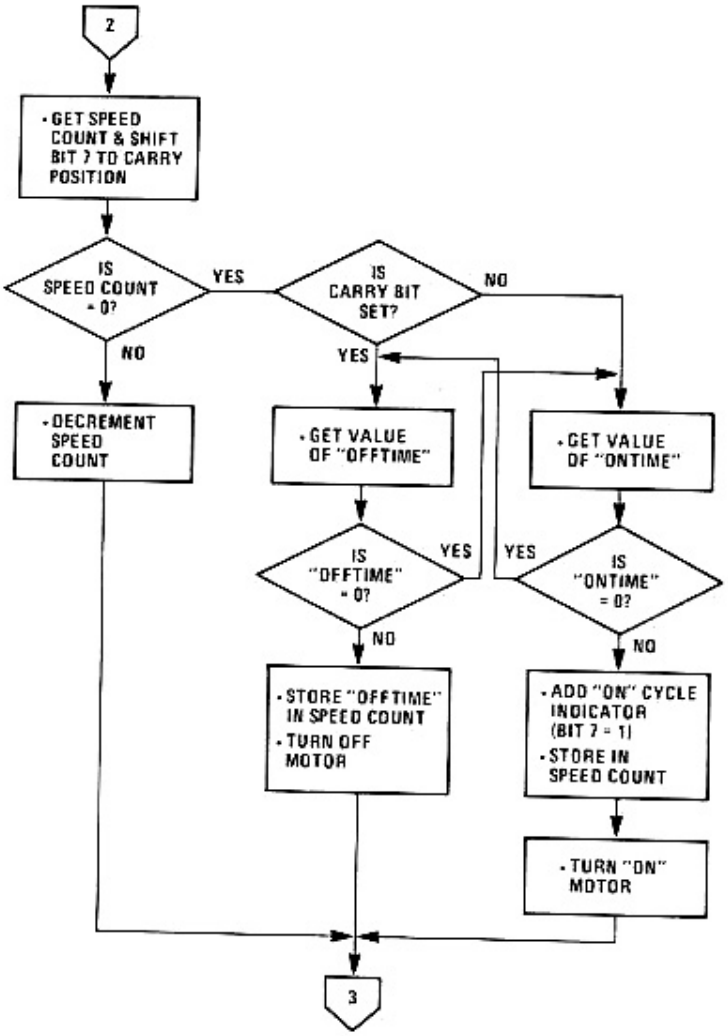

2
• GET SPEED COUNT & SHIFT BIT 7 TO CARRY POSITION
IS SPEED COUNT = 0?
YES
NO
IS CARRY BIT SET?
NO
YES
• DECREMENT SPEED COUNT
• GET VALUE OF "OFFTIME"
• GET VALUE OF "ONTIME"
IS "OFFTIME" = 0?
YES
NO
IS "ONTIME" = 0?
YES
NO
• STORE "OFFTIME" IN SPEED COUNT
• TURN OFF MOTOR
• ADD "ON" CYCLE INDICATOR (BIT 7 = 1)
• STORE IN SPEED COUNT
• TURN "ON" MOTOR
3

Chart 4–3.

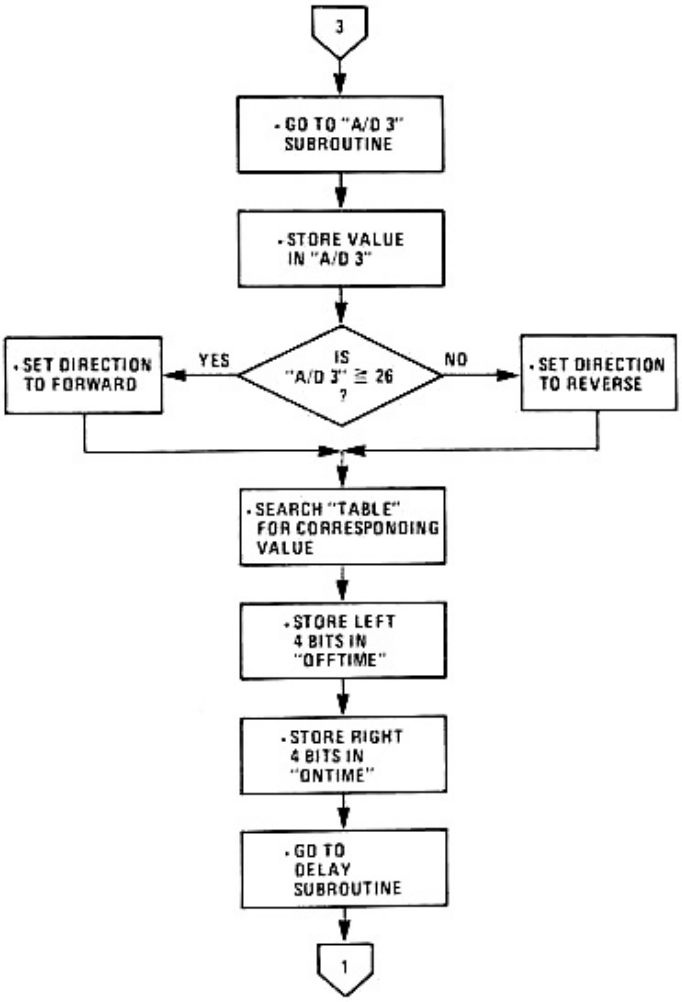

3
• GO TO "A/D 3" SUBROUTINE
• STORE VALUE IN "A/D 3"
IS "A/D 3" ≧ 26 ?
YES
NO
• SET DIRECTION TO FORWARD
• SET DIRECTION TO REVERSE
• SEARCH "TABLE" FOR CORRESPONDING VALUE
• STORE LEFT 4 BITS IN "OFFTIME"
• STORE RIGHT 4 BITS IN "ONTIME"
• GO TO DELAY SUBROUTINE
1

Chart 4–4.

Table 4–3. Speed Control

SPEED INPUT VALUE	*TABLE VALUE*	*SPEED VALUE*	*SPEED MEANING*	*DIRECTION*
≤0E	0F	0A	Fast	Reverse
0F–11	12	55	Medium Fast	Reverse
12–16	17	73	Medium	Reverse
17–1C	1D	82	Medium Slow	Reverse
1D–22	23	91	Slow	Reverse
23–25	26	A0	Off	Reverse
26–28	29	91	Slow	Forward
29–2A	2B	82	Medium Slow	Forward
2B–2C	2D	73	Medium	Forward
2D–2E	2F	55	Medium Fast	Forward
2F≥	30	0A	Fast	Forward

computer finds a table value that is greater than the input value, it takes the corresponding speed value. The first digit of the speed value tells the computer the number of "motor off" cycles out of every 10 cycles. The second digit tells the micro the number of "motor on" cycles out of every 10. The motor speed determined by this routine is executed by the speed control portion of the program. The software is entered into the Kim using the hex keypad.

The Joystick Control Program follows.

```
JOYSTICK CONTROL PROGRAM
WRITTEN BY JOHN W. LOOFBOURROW

                ; Reserved Storage Areas
                ;
0000                    *=$0000
0000            SAV0    *=*+1           A/D 1 count
0001            SAV1    *=*+1           A/D 2 count
0002            SPEED   *=*+1           speed counter
0003            ONTIME  *=*+1           on time count
0004            OFFTIM  *=*+1           off time count
                ;
                ; Table of Speed/Motion Values
                ;
0010                    *=$0010
0010 0F         TABLE   .BYTE $0F,$0A   fast reverse
0011 0A
0012 12                 .BYTE $12,$55   med. fast rev.
0013 55
0014 17                 .BYTE $17,$73   med. reverse
0015 73
0016 1D                 .BYTE $1D,$82   med. slow rev.
0017 82
0018 23                 .BYTE $23,$91   slow reverse
0019 91
001A 26                 .BYTE $26,$A0   off
001B A0
001C 29                 .BYTE $29,$91   slow forward
001D 91
001E 2B                 .BYTE $2B,$82   med. slow for.
001F 82
0020 2D                 .BYTE $2D,$73   med. forward
0021 73
0022 2F                 .BYTE $2F,$55   med. fast for.
0023 55
0024 30                 .BYTE $30,$0A   fast forward
0025 0A
                ;
00F9                    *=$00F9
00F9            SAV3    *=*+1           A/D 3 count
                ;
                ; Table of Directional Control Values
                ;
0373                    *=$0373
0373 0E         TABLE1  .BYTE $0E,$25   60° right
0374 25
0375 13                 .BYTE $13,$26   48° right
0376 26
0377 18                 .BYTE $18,$27   36° right
0378 27
0379 1D                 .BYTE $1D,$28   24° right
037A 28
```

```
037B 22              .BYTE $22,$29     12° right
037C 29
037D 27              .BYTE $27,$2A     0° (straight)
037E 2A
037F 2A              .BYTE $2A,$2B     12° left
0380 2B
0381 2C              .BYTE $2C,$2C     24° left
0382 2C
0383 2E              .BYTE $2E,$2D     36° left
0384 2D
0385 30              .BYTE $30,$2E     48° left
0386 2E
0387 31              .BYTE $31,$2F     60° left
0388 2F
                ;
                ; Initialization Sequence
                ;
                ; This program starts execution
                ; at address $0290.
                ;
0290                 *=$0290
0290 A9 1F    INIT   LDA #$1F          PA0-4 = output
0292 8D 0117         STA $1701         PA5-7 = input
0295 A9 00           LDA #$00          PB0-7 = input
0297 8D 0317         STA $1703
029A A9 00           LDA #$00
029C 85 02           STA SPEED         clear spd. count
029E 85 03           STA ONTIME        clr. on time cnt.
02A0 A9 0A           LDA #$0A
02A2 85 04           STA OFFTIM        OFFTIM = $A
02A4 A9 22           LDA #$22          center wheel
02A6 85 01           STA SAV1
02A8 A9 FF           LDA #$FF
02AA 8D 0017         STA $1700
02AD 4C 3502         JMP SCAN3         start dir. con.
                ;
                ; A/D 1 Subroutine
                ;
0200                 *=$0200
0200 20 1F02 A/D1    JSR START
0203 2C 0017 LOOP1   BIT $1700         test for input
0206 10 FB           BPL LOOP1         bit 7 = 1?
0208 AD 0017 STOP    LDA $1700         read PA
020B 29 FE           AND #$FE          bit 0 = 0
020D 8D 0017         STA $1700         PA0 = 0
0210 AD 0417         LDA $1704         A = timer count
0213 60              RTS
                ;
                ; A/D 2 Subroutine
                ;
0214 20 1F02 A/D2    JSR START
0217 2C 0017 LOOP2   BIT $1700         test for input
021A 50 FB           BVC LOOP2         bit 6 = 1?
```

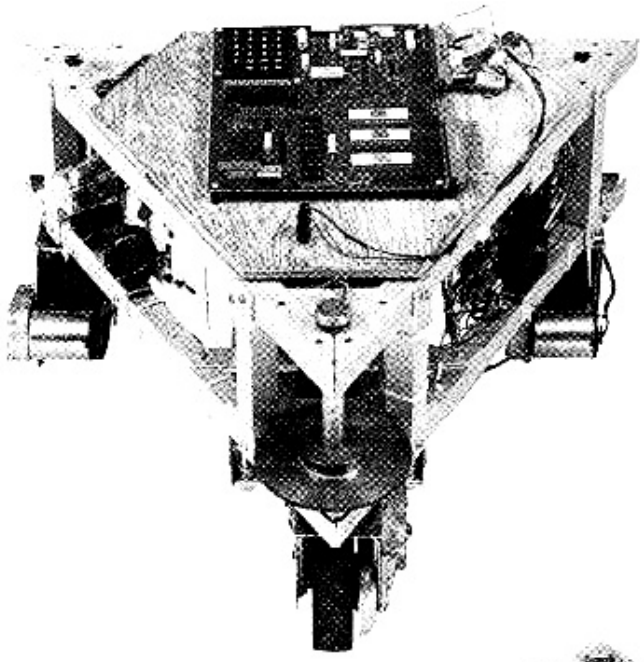
```
021C 4C 0802          JMP STOP
                ;
                ; Common Timer Start Subroutine
                ; for A/D Conversion
                ;
021F A9 42    START   LDA #$42          starting count
0221 8D 0517          STA $1705         store in timer
0224 AD 0017          LDA $1700         read PA (port A)
0227 09 01            ORA #$01          PA0 = 1
0229 8D 0017          STA $1700
022C 60               RTS
                ;
                ; Main Scan Loop - Directional
                ; Control
                ;
0235                  *=$0235
0235 20 0002  SCAN3   JSR A/D1          get dir. value
0238 85 00            STA SAV0          save result
023A 20 7D02          JSR WAIT          delay for cap.
023D 4C 5C03          JMP LIMIT
0240 85 01    LIMRET  STA SAV1          save result
0242 20 7D02          JSR WAIT          delay for cap.
0245 A5 00            LDA SAV0          A = SAV0
0247 E5 01            SBC SAV1          A = A-loc. 1
0249 F0 0F            BEQ EQ
024B 10 18            BPL PL
024D AD 0017          LDA $1700         read PA
0250 09 04            ORA #$04          PA2 = 1
0252 29 FD            AND #$FD          PA1 = 0
0254 8D 0017          STA $1700         (left turn)
0257 4C C002          JMP SCAN          cont. scan
                ;
025A AD 0017  EQ      LDA $1700         read PA
025D 09 06            ORA #$06          PA1-2 = 1
025F 8D 0017          STA $1700         (no turn)
0262 4C C002          JMP SCAN          cont. scan
                ;
0265 AD 0017  PL      LDA $1700         read PA
0268 09 02            ORA #$02          PA1 = 1
026A 29 FB            AND #$FB          PA2 = 0
026C 8D 0017          STA $1700         (right turn)
026F 4C C002          JMP SCAN          cont. scan
                ;
                ; Delay Subroutine - Waits for
                ; Capacitor to Discharge
                ;
027D A9 28    WAIT    LDA #$28          A = timer count
027F 8D 0517          STA $1705         timer = A
0282 A9 00            LDA #$00
0284 CD 0417  CHECK   CMP $1704
0287 D0 FB            BNE CHECK
0289 60               RTS
                ;
```

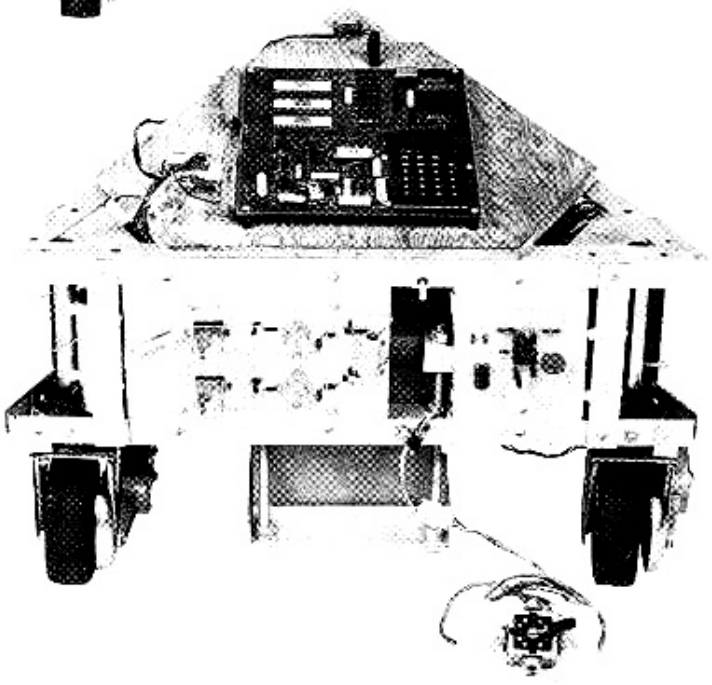
```
                ; Main Scan Loop
                ; The following controls the speed
                ; of the drive motors based on the
                ; number of times through the scan
                ; loop.
                ;
02C0                    *=$02C0
02C0 A5 02      SCAN    LDA SPEED       A = speed count
02C2 0A                 ASL
02C3 F0 05              BEQ CHNG        speed count = 0?
02C5 C6 02              DEC SPEED       dec. count
02C7 4C 0003            JMP SCAN1       cont. scan
02CA B0 13      CHNG    BCS OFF         done?
02CC A5 03      ON      LDA ONTIME      A = ONTIME
02CE F0 0F              BEQ OFF         A = 0?
02D0 09 80              ORA #$80        ind. "on" cycle
02D2 85 02              STA SPEED       A = speed count
02D4 AD 0017            LDA $1700       read PA
02D7 29 F7              AND #$F7        PA3 = 0
02D9 8D 0017            STA $1700       (turn motor on)
02DC 4C 0003            JMP SCAN1       cont. scan
                ; end of "on" cycle
02DF A5 04      OFF     LDA OFFTIM      A = OFFTIM
02E1 F0 E9              BEQ ON          A = 0?
02E3 85 02              STA SPEED       SPEED = A
02E5 AD 0017            LDA $1700       read PA
02E8 09 08              ORA #$08        PA3 = 1
02EA 8D 0017            STA $1700       (speed control off)
02ED 4C 0003            JMP SCAN1       cont. scan
02F0 20 1F02 A/D3       JSR START       start A/D 3
02F3 A9 20   LOOP3      LDA #$20        set mask for PA5
02F5 2D 0017            AND $1700       read PA
02F8 F0 F9              BEQ LOOP3       PA5 = 0?
02FA 20 0802            JSR STOP        A = A/D 3 value
02FD 60                 RTS
                ;
0300                    *=$0300
0300 20 F002 SCAN1      JSR A/D3        A = A/D 3 value
0303 85 F9              STA SAV3        save value
0305 C9 26              CMP #$26        A > = 26?
0307 10 2E              BPL FOR         if so, forward
0309 AD 0017            LDA $1700       read PA
030C 09 10              ORA #$10        PA4 = 1
030E 8D 0017            STA $1700       (reverse)
0311 A5 F9   SCTAB      LDA SAV3        get A/D 3 value
0313 A2 00              LDX #$00        X = 0
0315 D5 10   AGAIN      CMP TABLE,X
0317 EA                 NOP
0318 30 06              BMI FOUND       hit on table?
031A E8                 INX
031B E8                 INX             inc. pointer
031C E0 14              CPX #$14        end of table?
031E D0 F5              BNE AGAIN
0320 E8      FOUND      INX             A = "action"
```

```
0321 B5 10          LDA TABLE,X      value from table
0323 EA             NOP
0324 4A             LSR              shift left 4 bits
0325 4A             LSR              to the right -
0326 4A             LSR              fill in left
0327 4A             LSR              with 0's
0328 85 04          STA OFFTIM       OFFTIM = A
032A B5 10          LDA TABLE,X      get "action" value
032C EA             NOP
032D 29 0F          AND #$0F         mask out left 4 bits
032F 85 03          STA ONTIME       ONTIME = A
0331 20 7D02        JSR WAIT         delay
0334 4C 3502        JMP SCAN3
0337 AD 0017 FOR    LDA $1700
033A 29 EF          AND #$EF
033C 8D 0017        STA $1700        PA4 = 0 (for.)
033F 4C 1103        JMP SCTAB        search table
                ;
                ; Directional Control Limit Routine
                ;
035C                 =$035C
035C 20 1402 LIMIT  JSR A/D2         get count
035F A2 00          LDX #$00         X = 0
0361 DD 7303 AGAIN2 CMP TABLE1,X     hit on table?
0364 30 06          BMI FOUND2       branch if hit
0366 E8             INX              increment to
0367 E8             INX              next value
0368 E0 14          CPX #$14         end of table?
036A D0 F5          BNE AGAIN2
036C E8       FOUND2 INX             get dir. contr.
036D BD 7303        LDA TABLE1,X     A = table value
0370 4C 4002        JMP LIMRET       value.  Return.
```

After entering and testing the software, center the joystick. Remove the directional control pot from the shaft of the front wheel and center the wheel. Replace the pot, making sure that its shaft does not turn as you put it back on the front wheel.

Stage I is complete.





the self-direction program

In Stage I Mike can be made to operate independently of the controller by your loading in a program for self-direction. The self-direction program causes him to move in a predetermined pattern. The pattern is determined by a table of numbers in the computer. By changing the table you can make Mike move in almost any pattern you may want.

The self-direction program consists of two subroutines. The routines allow Mike to move without a controller and are the first step toward his future independence. The first subroutine—the self-turning routine—replaces the A/D 2 subroutine (see Chart 4-5). Notice the

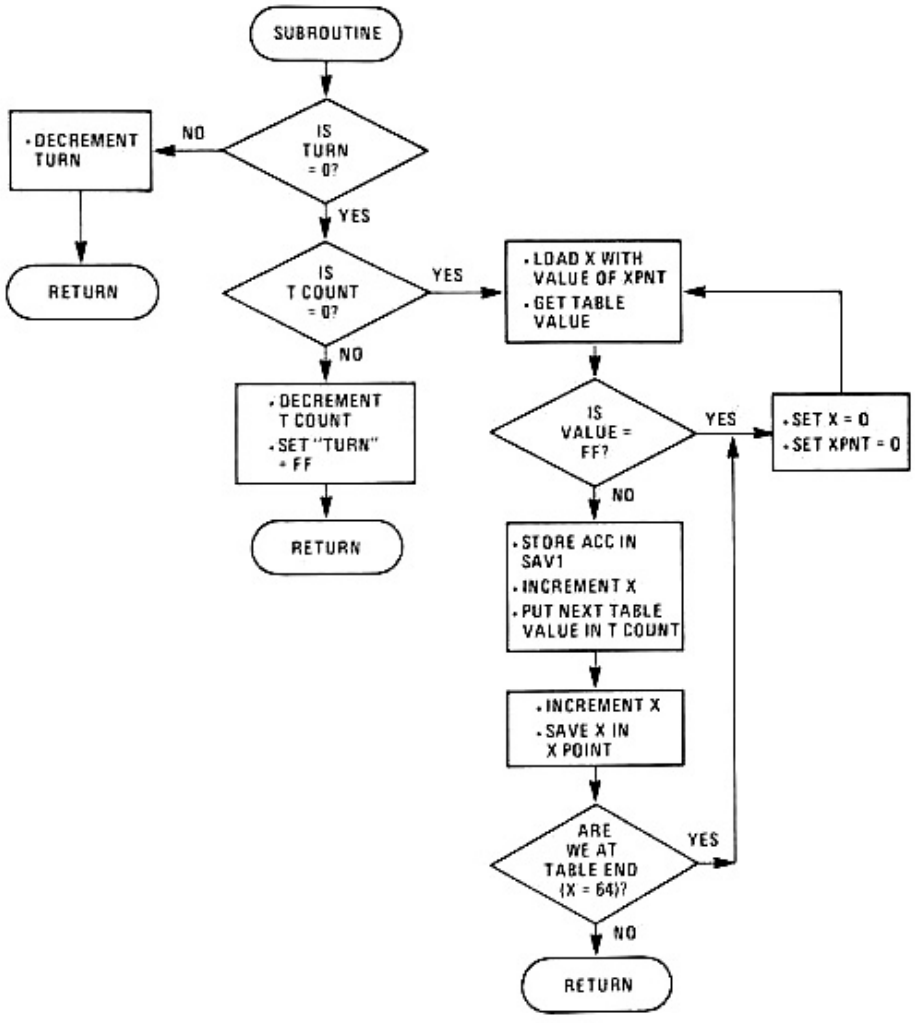


Chart 4–5.

position of this new routine in Chart 4–2. The main purpose of the routine is to replace the commands from the directional control command pot with values from a table stored in the microprocessor. One at a time, the values are stored in "SAV1," where the result from A/D 2 is usually stored. In this manner, the computer is tricked into thinking that it is obeying the joystick's commands.

The directional control table contains four columns (Table 4–4). The first shows the addresses in the computer at which the values are stored. The second column lists the directional control value that determines the turning angle. The third column determines the duration of the turn. The fourth column contains additional comments. An "FF" in

Table 4–4. Sample Directional Control Table

ADDRESS	*DIRECTIONAL CONTROL VALUE*	*TIME*	*COMMENTS*
0030	29	06	Byte 1 = Direction
0032	2C	06	Byte 2 = Time
0034	FF	FF	

both the second and third columns indicates the end of a sequence of actions. When the computer reads an "FF," it returns to the beginning of the table and starts the process all over again. The third column is telling the computer how many times it must count down from "FF." The computer counts down from "FF" to "00" every time it goes through the entire main program loop 256 times. The counter that decrements from "FF" to "00" is labeled "Turn." The number from the third column is stored in a counter labeled "T Count." Every time the "Turn" counter reaches "00," the "T Counter" is reduced by 1. This happens about twice every second. When the value in "T Count" reaches 0, the computer moves on to the next pair of values in the table.

The second subroutine of the self-direction program is the self-speed routine. The self-speed routine replaces the A/D 3 subroutine (Chart 4–6). Notice this routine's position in Chart 4–4. The self-speed routine replaces the commands from the speed command pot with values from a table stored in the microcomputer. These values are stored in "SPEED," where the result from A/D 3 is usually stored. As with the directional control table, the speed table contains four columns (Table 4–5), the first of which contains the addresses at which the table is located. The second column contains the speed input value that determines the speed at which Mike will move. The third column determines

Table 4–5. Sample Speed Table

ADDRESS	*SPEED VALUE*	*TIME*	*COMMENTS*
0070	25	02	Byte 1 = Speed
0072	22	04	Byte 2 = Time
0074	25	01	
0076	28	05	
0078	FF	FF	

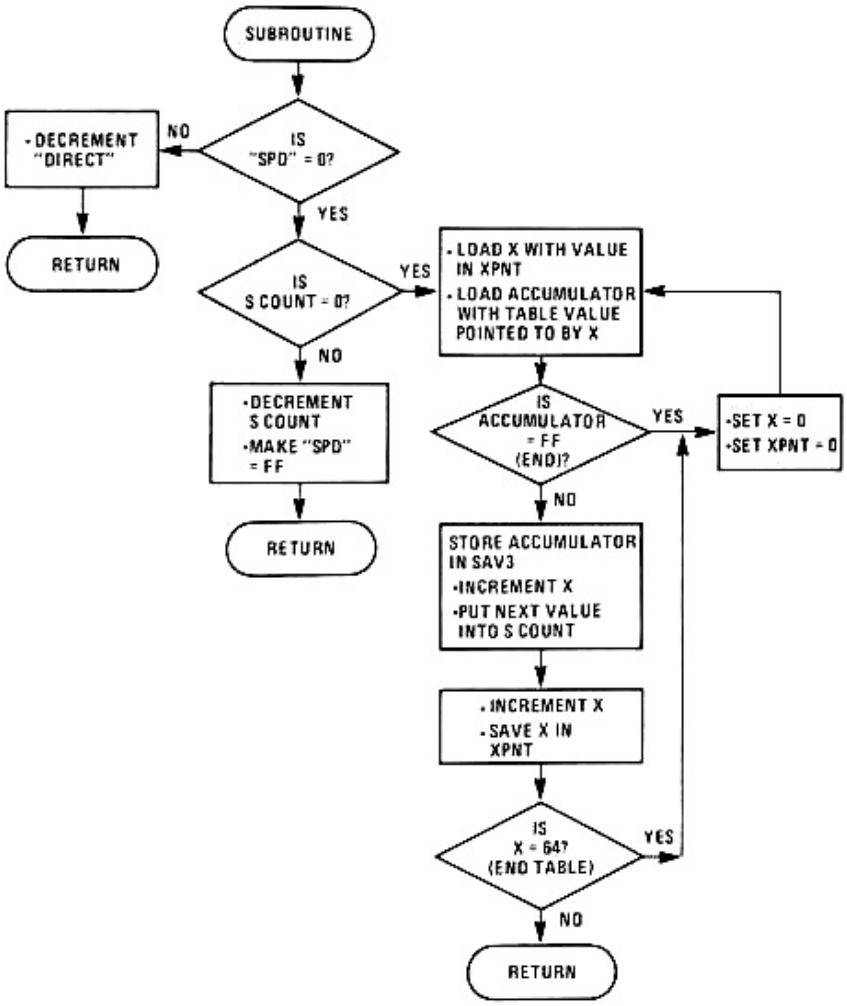


Chart 4–6.

the duration of time that the speed will be maintained. The fourth column contains comments. An "FF" in the second and third columns indicates the end of a sequence of actions, and it causes the micro to repeat the sequence. The counters work in the same way as they do in the self-turn routine. However, in the self-speed routine, the counter that stores the value from the second column of the speed table is labeled "S Count." The counter that counts down from "FF" to "00" is labeled "Speed."

The values given in the sample directional control and speed tables cause Mike to move in a pattern that forms an asterisk. Table 4–6 contains the values that will allow Mike to move in a cloverleaf pattern. You can program Mike to execute patterns of your own design by loading the proper speed and directional control values into the tables. The listing contains the two main subroutines of the Self-Direction Program, as well as certain changes that must be made in the Joystick Control Program in order to allow Mike to be self-directed.

Table 4–6. Tables for Cloverleaf Pattern

Directional Control Table

ADDRESS	*DIRECTIONAL CONTROL VALUE*	*TIME*	*POSITION*
0030	2A	02	Center
0032	25	37	Right
0034	2A	02	Center
0036	2F	37	Left
0038	2A	02	Center
003A	2F	37	Left
003C	2A	02	Center
003E	25	37	Right
0040	FF		

ADDRESS	*SPEED VALUE*	*TIME*	*SPEED*
0070	28	02	Slow
0072	2C	37	Medium
0074	28	02	Slow
0076	2C	37	Medium
0078	28	02	Slow
007A	2C	37	Medium
007C	28	02	Slow
007E	2C	37	Medium
0080	FF		

SELF DIRECTION PROGRAM

WRITTEN BY JOHN W. LOOFBOURROW

```
                ; Table of Additional Storage
                ; Areas Needed for Self Directed
                ; Turns and Speed
                ;
0005                    *=$0005
0005            TURN    *=*+1           turn count (LSB)
0007                    *=$0007
0007            SPD     *=*+1           speed counter (LSB)
0008            TCOUNT  *=*+1           turn cnt. (MSB)
0009            XPOINT  *=*+1           direct. cntl. pntr.
000A            SCOUNT  *=*+1           speed cntr. (MSB)
000B            XPNT    *=*+1           dir. table pntr.
                ;
0030                    *=$0030
0030            TABLE3  *=*+$40         dir. cntl. table
0070            TABLE4  *=*+$40         spd./dir. table
                ;
                ; Changes Required in Joystick
                ; Program to Activate Self Direction
                ; Programs
                ;
023D                    *=$023D
023D 20 0001            JSR SELFT
0240 EA                 NOP
0241 EA                 NOP
                ;
0300                    *=$0300
0300 20 3001            JSR SELF
                ;
02AD                    *=$02AD
02AD A9 00              LDA $00
02AF 85 05              STA TURN
02B1 85 07              STA SPD
02B3 85 08              STA TCOUNT
02B5 85 09              STA XPOINT
02B7 85 0A              STA SCOUNT
02B9 85 0B              STA XPNT
02BB 4C 3502            JMP SCAN3
                ;
                ; This subroutine substitutes for
                ; the A/D2 subroutine.  It reads
                ; the values in Table 3 to determine
                ; the turning angle and how long
                ; that turn should last.
                ;
0100                    *=$0100
0100 A5 05      SELFT   LDA TURN        get turn value
0102 F0 03              BEQ CKMSB
0104 C6 05              DEC TURN        if ≠ 0 sub. 1
```

```
0106 60              RTS
0107 A5 08   CKMSB   LDA TCOUNT          get TCOUNT
0109 F0 07           BEQ GTVALU          branch if = 0
010B C6 08           DEC TCOUNT
010D A9 FF           LDA #$FF
010F 85 05           STA TURN            TURN = $FF
0111 60      RTS     RTS
0112 A6 09   GTVALU  LDX XPOINT          get pointer
0114 B5 30   CKTAB3  LDA TABLE3,X        get turn value
0116 C9 FF           CMP #$FF            check if done
0118 F0 0E           BEQ GOBACK
011A 85 01           STA SAV1            save TABLE3 value
011C E8              INX                 point to next
011D B5 30           LDA TABLE3,X        get time
011F 85 08           STA TCOUNT          store time in cntr.
0121 E8              INX                 move to next value
0122 86 09           STX XPOINT          save pointer
0124 E0 40           CPX #TABLE3+$40     check for end
0126 D0 E9           BNE RTS
0128 A2 00   GOBACK  LDX #$00
012A 86 09           STX XPOINT          set pointer to 0
012C 4C 1401         JMP CKTAB3
                ;
                ; This subroutine substitutes for
                ; the A/D3 subroutine.  It reads
                ; the values in TABLE4 to determine
                ; the speed, direction, and how
                ; long that combination should
                ; last.
                ;
0130                 *=$0130
0130 A5 07   SELF    LDA SPD             check LSB of time
0132 F0 05           BEQ CKBYTE
0134 C6 07           DEC SPD
0136 A5 F9           LDA SAV3            get dir. byte
0138 60              RTS
0139 A5 0A   CKBYTE  LDA SCOUNT          check MSB of time
013B F0 09           BEQ SCHTAB
013D C6 0A           DEC SCOUNT
013F A9 FF           LDA #$FF            reset LSB
0141 85 07           STA SPD
0143 A5 F9   RTS2    LDA SAV3            get dir. byte
0145 60              RTS
0146 A6 0B   SCHTAB  LDX XPNT
0148 B5 70   SCHTB1  LDA TABLE4,X        get TABLE4 value
014A C9 FF           CMP #$FF            check if end
014C F0 0E           BEQ GOEND           branch if end
014E 85 F9           STA SAV3            save table value
0150 E8              INX                 point to time
0151 B5 70           LDA TABLE4,X        get time
0153 85 0A           STA SCOUNT          store in counter
0155 E8              INX                 point to next value
0156 86 0B           STX XPNT            save value
```

```
0158 E0 40          CPX #TABLE4+$40   check for end
015A D0 E7          BNE RTS2
015C A2 00   GOEND  LDX #$00          restore to 0
015E 86 0B          STX XPNT
0160 4C 4801        JMP SCHTB1
                    END
```

chapter five impact sensors

Parts List:

20 feet of 1 x 1 x 1/8 angle aluminum
6 inches of 1 1/2 x 1/8 flat aluminum
A 4- by 4-ft square of 0.050-in. sheet aluminum (see text)
3 3/16- by 1 1/2-in. stove bolts
20 3/16- by 1/2-in. stove bolts
23 3/16-in. nuts
23 3/16-in. lock washers
3 No. 8 5/8-in. wood screws
5 feet of 8-oz ribbon switch (see text)
4 tubes of instant bonding glue
58 inches of black electrical tape

Impact Sensor Detector Parts List:

1 7430 8-input NAND gate IC
1 7404 hex inverter IC

Impact Sensor Selector Parts List

1 74151 IC 1-of-8 data selector

In Stage I Mike is totally under your control, and he has no means of sensing his environment. As you progress with Stage II and Mike becomes independent, he gains various methods of sensing his surroundings. Mike's sense of touch comes from eight impact sensors. The sensors are mounted on an eight-sided frame that surrounds the inner

triangular frame. Each sensor is made up of five ribbon switches mounted between two rectangular sheets of aluminum.

If one of Mike's sensors makes contact with an object when he is moving about an area, he exhibits a basic reflex action and moves away from the object. After retreating from an obstacle, Mike turns in order to avoid it and then returns to his ambulatory activities.

outer frame

The outer frame is an eight-sided structure that supports the impact sensors. I decided to use an eight-sided frame to meet two major requirements. First, the outer frame had to be easy to attach to the inner triangular frame. I designed the eight-sided figure so that, for mounting purposes, it would overlap, at some point each vertex of the triangular frame. The outer frame had to be narrow enough for Mike to fit through doorways. The eight-sided frame satisfies this need. The outer "skeleton" is attached to the triangular frame in five separate places. The two structures are anchored together at the vertices of the triangle and supported by two aluminum braces (Fig. 5–1).

The outer frame is built in two steps. First, an eight-sided structure is built and anchored to the upper triangle. Then an identical

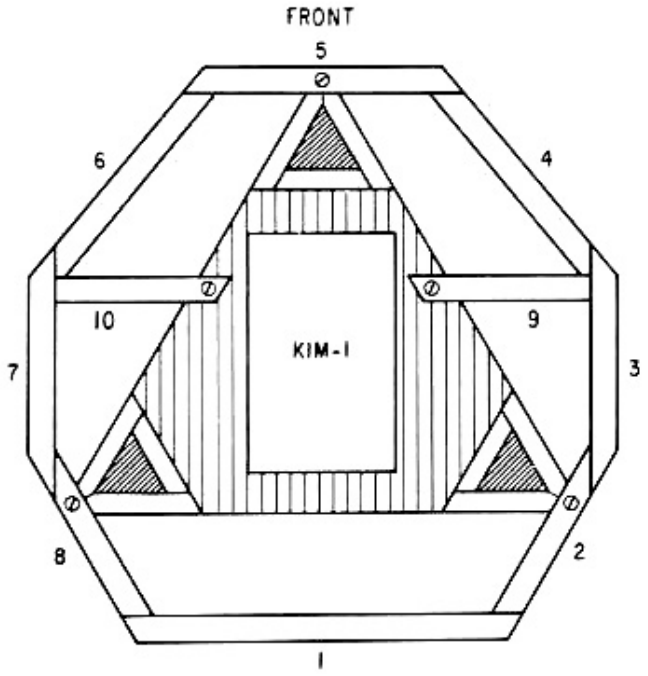


Fig. 5–1. Outer hull attached to triangular frame. Numbers shown are piece numbers.

structure is built and anchored to the lower triangle. The upper section of the outer frame is constructed first. Each leg of the upper section is unique, and therefore must be cut individually. Figure 5–2 shows the angles to be cut in pieces 1 through 4. Figure 5–3 shows the angles to be cut in pieces 4 through 8. Figure 5–4 shows the angles to be cut in pieces 9 and 10. Notice the position of both faces of the angle aluminum as you

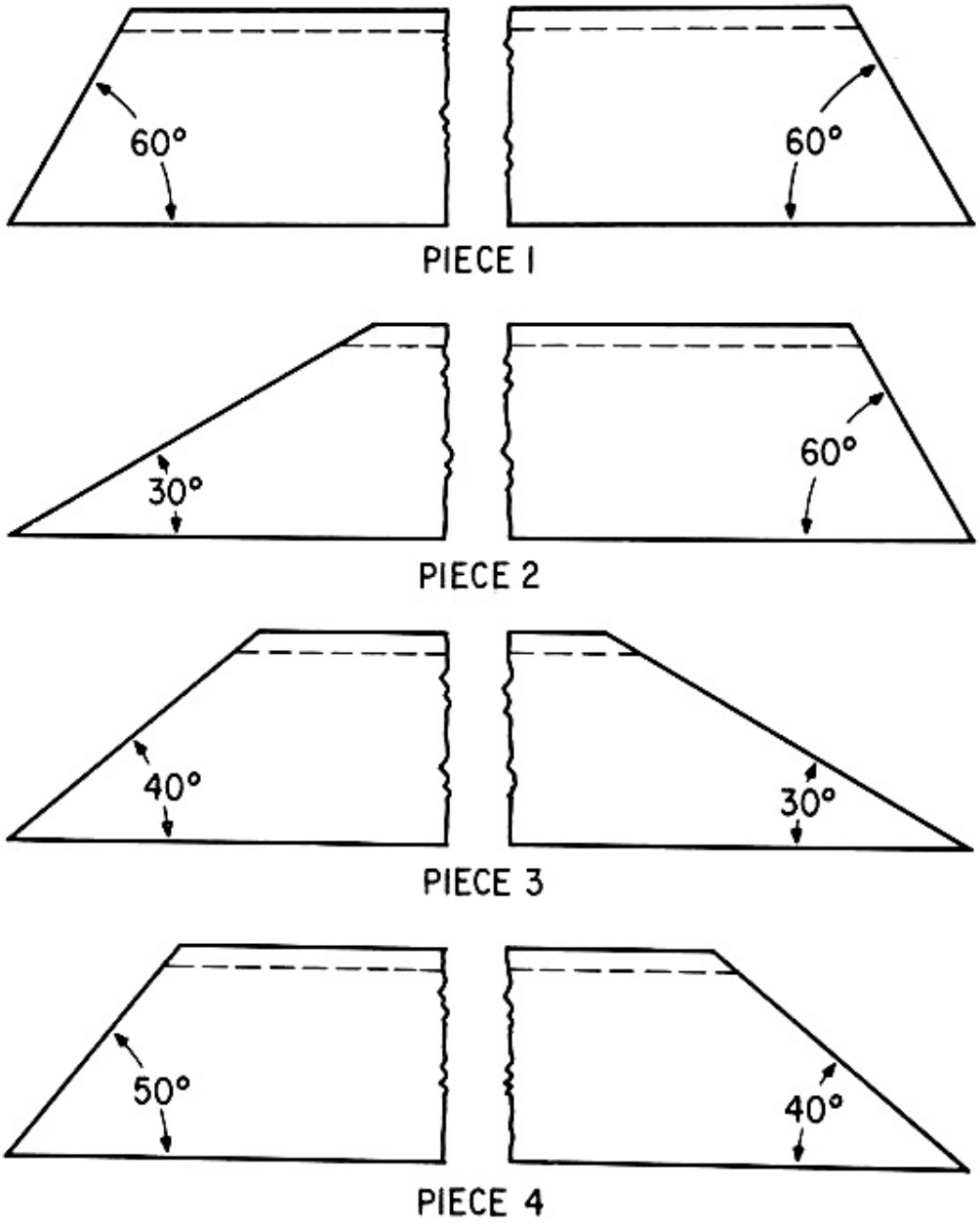


Fig. 5–2. Angles of pieces 1 through 4 in the impact sensor frame.

cut each piece. The face in which you are not cutting the angles should be pointing downward. In pieces 9 and 10, 1¾ in. should be cut off at each end, as shown in Fig. 5–4. Figures 5–2, 5–3, and 5–4 show only the angles to be cut. The length of each piece before cutting the angles is shown in Table 5–1.

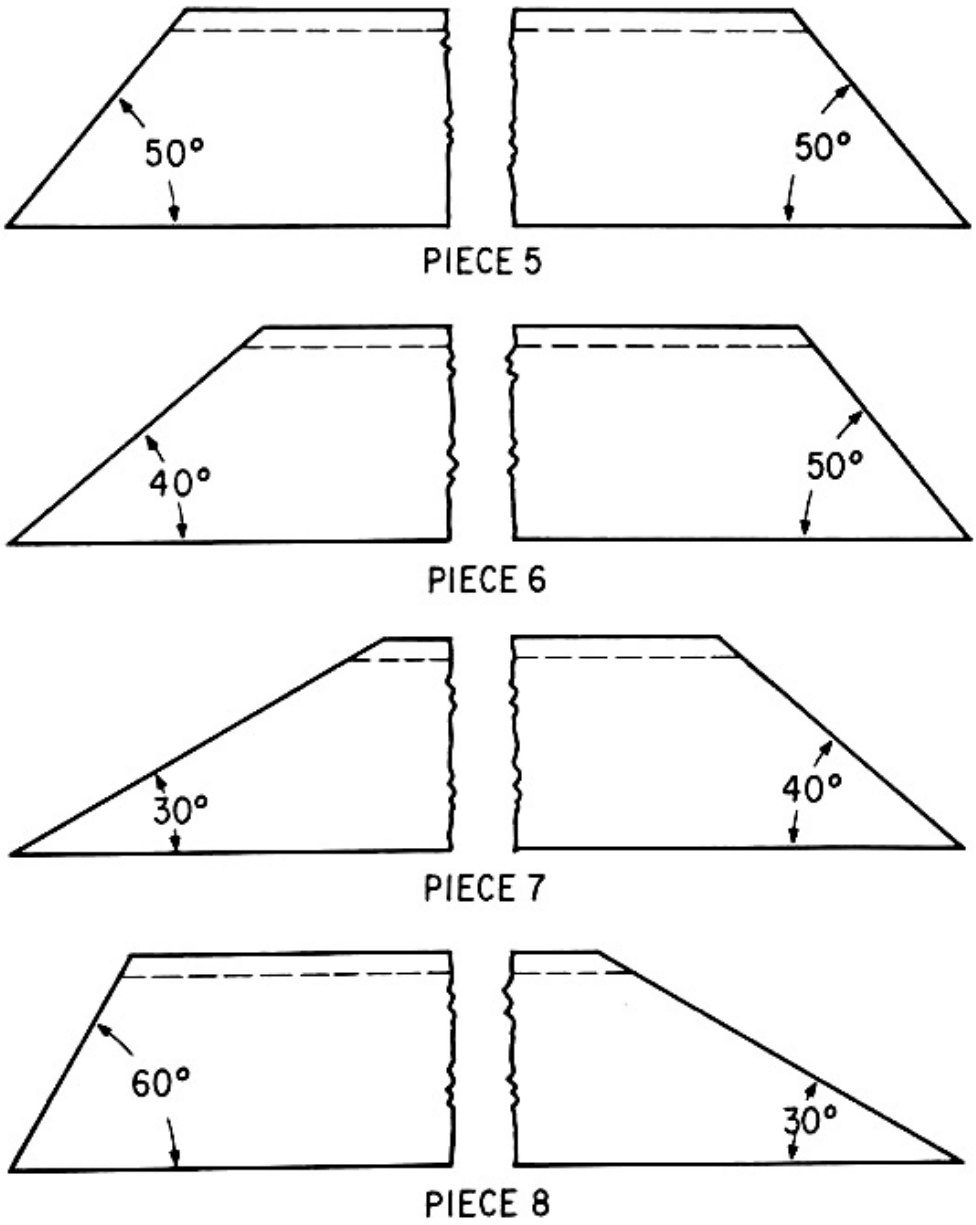


Fig. 5–3. Angles of pieces 5 through 8 in the impact sensor frame.

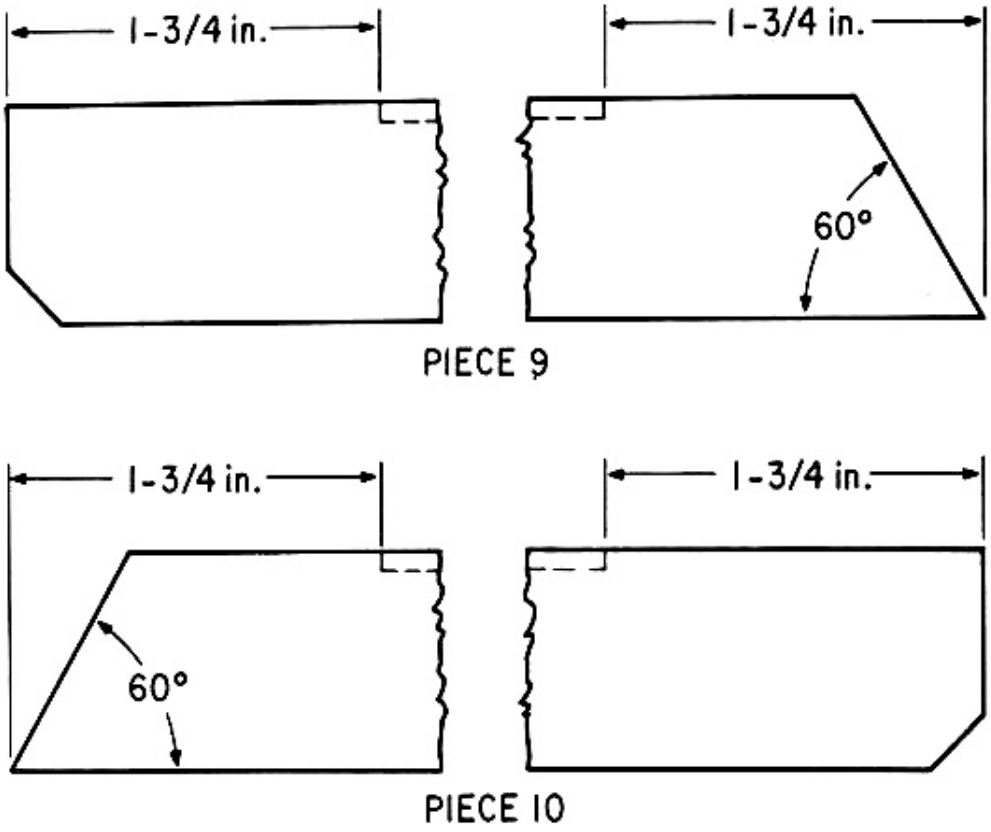


Fig. 5–4. Angles of pieces 9 and 10 in the impact sensor frame.

Table 5–1. Lengths of Angle Aluminum

ANGLE ALUMINUM	*LENGTH*
Piece 1	18 1/4 in.
Piece 2	11 in.
Piece 3	10 1/2 in.
Piece 4	14 5/8 in.
Piece 5	11 3/4 in.
Piece 6	14 5/8 in.
Piece 7	10 1/2 in.
Piece 8	11 in.
Piece 9	8 5/8 in.
Piece 10	8 5/8 in.

Cut the pieces of the outer frame out of 1-in. by 1-in. by 1/8-in. angle aluminum. Copy the proper angles for each piece from the appropriate figure, using a sliding "T" bevel or a protractor. Cut the pieces and compare their angles with the angles in the diagrams. Because each piece is cut with one face of the aluminum pointing downward, it might be hard to compare the angles of each piece to its appropriate figure. To check each piece to make sure it is properly cut, turn the face so that it points upward and lay the piece on its complementary figure. In other words, lay piece 2 on the figure for piece 8, piece 3 on the figure for piece 10, piece 4 on the figure for piece 6, and piece 9 on the figure for piece 10.

Place the pieces as shown in Fig. 5–5 and bolt all joints. Make sure that the face of each piece without the angles is pointing downward. The letter shown on each piece in the illustration indicates its placement level in the frame's construction. An "A" indicates the lowest level. A "B" is always placed on top of an "A," or under a "C." You may find it easier to assemble the frame two pieces at a time, placing each piece at the proper level as you go.

After bolting the upper section together, place it on the inner triangular frame. You may have to loosen the bolts of the upper section and ease it onto the triangle. Pieces 9 and 10 will have to be turned temporarily toward pieces 4 and 6, respectively. Using No. 8 5/8-in. wood

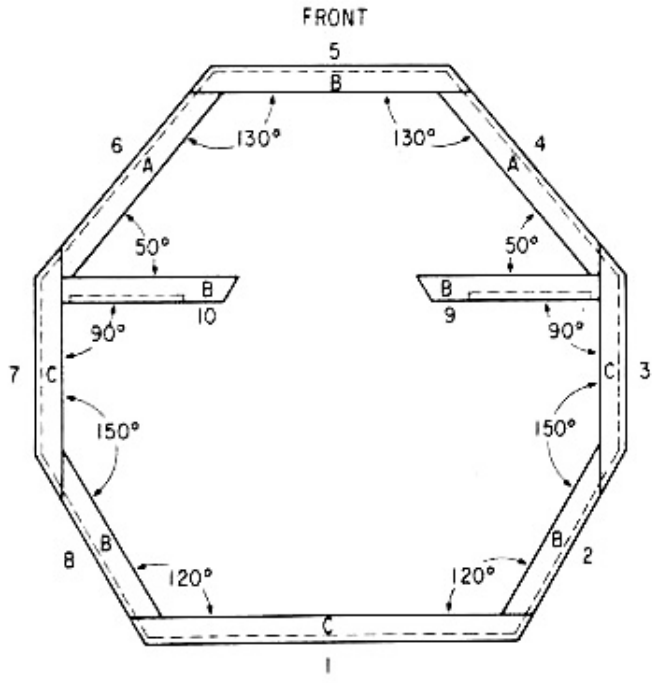


Fig. 5–5. Layout of impact sensor frame.

screws, attach the frame to each of the vertices of the upper triangle. The ends of pieces 9 and 10 that are not attached to the impact sensor frame should be bolted to the triangle. In order for pieces 9 and 10 to anchor to the triangular frame, part of the wood on which the Kim is mounted must be cut out. Remove the upper section of the outer hull and straighten pieces 9 and 10 to the proper angles. Place the frame back on the triangle and mark the places where 9 and 10 hit the wood. Remove the wood and cut out two sections to allow room for the aluminum braces. Replace the wood and bolt pieces 9 and 10 to the triangular frame.

The lower section of the outer frame is an exact duplicate of the upper section. The lower section is much easier to construct than the upper section. Merely take all the pieces from the upper section and trace their angles onto the aluminum to be used for the lower section. To avoid the slight amount of enlargement obtained by tracing, cut on the inside of all your lines. Bolt pieces 3, 4, and 9 together and pieces 7, 6, and 10 together. Pieces 2, 8, and 5 cannot be mounted to the triangular frame with wood screws as you did with the upper section. In the lower section the pieces are mounted by placing a $\frac{3}{16}$- by $1\frac{1}{2}$-in. stove bolt through each piece. The bolt should dangle between two sides of the lower triangle. Now place a 2-in. by 1-in. piece of flat aluminum under each bolt and drill in the center of the aluminum a hole large enough to accommodate the shaft of the bolt. Place the aluminum under the lower triangle and attach it to the bolt with a lock washer and a nut. Make sure that both ends of the aluminum are touching the lower triangle. Bolt the remaining pieces of the lower section to pieces 2, 8, and 5. Last, bolt pieces 9 and 10 to the lower triangle. The outer frame is complete.

impact sensors

The impact sensors are quite simple in concept. However, they were not simple to design. I considered many arrangements of switches, foam rubber, and springs before I settled on the present system. The main factors that I had to keep under consideration were cost, sensitivity, durability, and the ease with which a system could be interfaced with the software.

Sheer luck led me to the ribbon switches used in the impact sensors (see Appendix). The ribbon switches have a sensitivity of 8 oz and can support the weight of a car without breaking. Moreover, they are relatively inexpensive, and their closure is easily detected by the software. They can be mounted to the eight-sided frame on pieces of sheet aluminum.

Obtain a supply of 0.050-in. sheet aluminum. I was able to find aluminum with baked-on black enamel paint. The pieces that make up

Mike's outer hull are bolted on top of one another. If the sheet aluminum is cut to fit between each piece and its identical counterpart on the lower section, the impact sensors would all end up at different levels and would look terrible. To eliminate this problem, make all the sensors as high as the distance from the highest piece on the upper section of the frame to the lowest point on the lower section—7¼ in. The length of each impact sensor is shown in Table 5–2. The sensor numbers correspond to the piece numbers of the outer frame. I suggest that as you cut each sensor, cut both the inside and outside plates of the sensor. In other words, cut two pieces of aluminum exactly the same size when you cut each sensor. The ribbon switches are sandwiched between the two pieces of aluminum.

Table 5–2. Lengths of Impact Sensors

IMPACT SENSOR	*LENGTH*
Sensor 1	17 in.
Sensor 2	9¼ in.
Sensor 3	7½ in.
Sensor 4	13 in.
Sensor 5	10 in.
Sensor 6	13 in.
Sensor 7	7½ in.
Sensor 8	9¼ in.

After cutting the sensors, place one plate of each sensor against the outer frame in its proper place. Make sure that the plate lies flat against the aluminum of the frame. Line up each plate so that its top edge is at the same level as the tops of pieces 1, 3, and 7 of the upper section of the frame. Now attach the plate to the outer frame, using instant bonding glue. The plates should be clamped to the aluminum frame to allow the glue to dry thoroughly.

The ribbon switches are attached directly to the inner plates. Cut 40 pieces of ribbon switch, each about 1 in. long. Attach two wires to each piece at the places shown in Fig. 5–6. Be careful that wires do not short the top and bottom conductors. Lay out five ribbon switches on each plate as in Fig. 5–7. Attach the switches to the plate, using instant bonding glue. Run the wires through the small spaces between plates. Place a small amount of instant bonding glue in the bump of each ribbon switch of one plate. Firmly press the outer plate against the five ribbon switches, making sure that the outer plate is aligned with the inner plate. Allow the glue to dry. Repeat the process with the remaining seven plates.

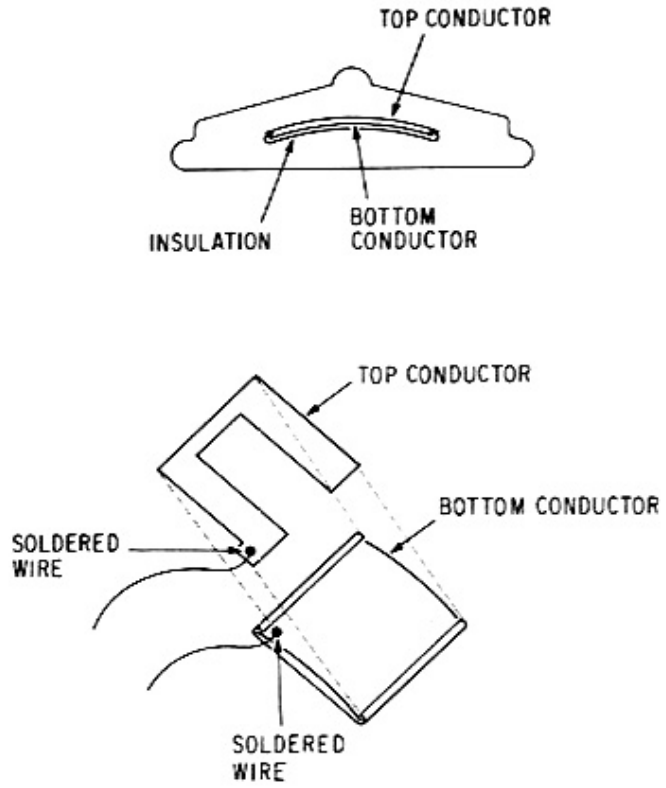


Fig. 5–6. Anchoring wires to each ribbon switch.

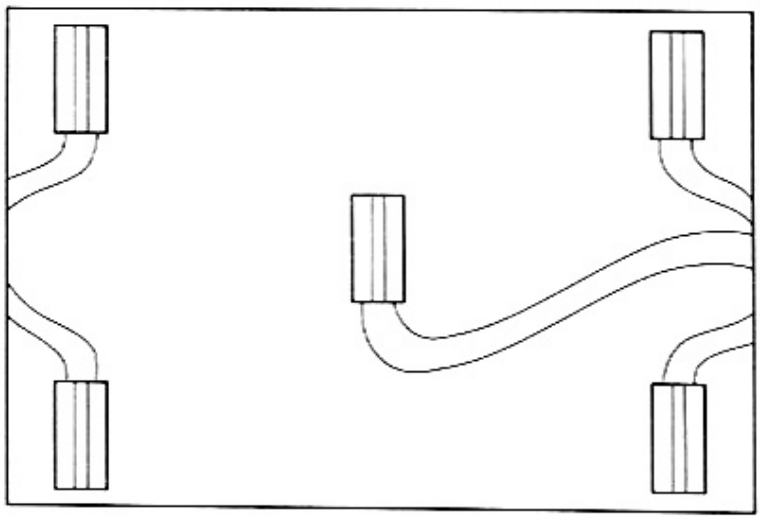


Fig. 5–7. Ribbon switches mounted to inner plate of each impact sensor.

You will notice that between the impact sensors are gaps about ¼ in. wide. These gaps allow the impact sensors to compress when hit without coming into contact with each other. If you want to cover these gaps, merely run a piece of black tape along each gap. This method will be effective only if your impact sensors are constructed out of black aluminum. The impact sensors are now complete.

Occasionally, Mike will run into an object that is too low-lying to make contact with his impact sensors. Usually, he will be able to run over the object, but sometimes it will be just high enough to obstruct his movement. A system for sensing obstacles that are not high enough to trigger an impact sensor would prevent the problem. A system of feelers was suggested to me whereby Mike could detect objects as low as 2 in. Since Mike very seldom runs into obstacles that don't trigger his impact sensors, I have not yet installed the system. I have included it as a suggested backup to Mike's impact sensors. Because it is only a suggestion, the parts necessary for its implementation are not listed in the parts list of this chapter.

To build the backup sensory system, cut four pieces of aluminum, each 4 in. by 1½ in. out of your 1½- by ⅛-in. flat aluminum. Mount an SPDT-center-off-momentary-contact toggle switch on one end of each piece. Mount it by drilling a hole large enough to accommodate its threaded shaft in one end of the aluminum and attaching it to the aluminum with a nut. The switches are placed on pieces 2, 4, 6, and 8 of the outer frame's lower section. The end of the flat aluminum on which the switch has not been mounted should be glued with instant bonding glue to the center of each piece, on the piece's vertical face. Attach about 4 in. of stiff plastic tubing to the lever of each switch. Surround the plastic with a thin sheet of rubber to prevent it from marring a contacted surface. Each feeler should be hooked up in parallel with the sensor below which it is installed.

impact sensor circuitry

In order for the microprocessor to detect hits on the impact sensors, two circuits are required. The first circuit, the impact sensor detector, informs the micro that an impact sensor has been hit (Fig. 5–8). Each impact sensor contains five ribbon switches. Take one wire from one of these ribbon switches and solder it to one wire from each of the other four switches. It doesn't matter which wire you use from each switch. Solder the remaining five wires together. Do the same to all eight impact sensors. Each sensor should now have two wires leading from it. Connect one wire from each sensor to ground. The remaining wire from each sensor is connected to the impact sensor detector. The impact sensor numbers, given in Figure 5–1, correspond to the "S" numbers (S1, S2,

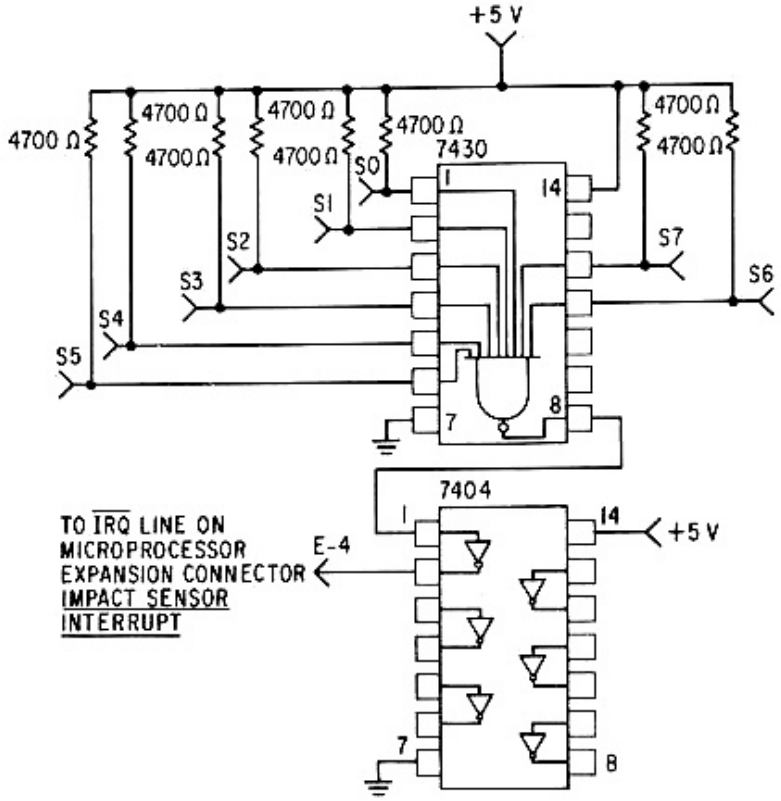


Fig. 5–8. Impact sensor detector.

S3, etc.) given in the impact sensor detector (Fig. 5–8). The only exception is sensor 8, the number of which has been changed to sensor 0 (S0), for software purposes.

If an impact sensor is hit, one of the inputs to the 7430 8-input NAND gate on the impact sensor detector goes to logic 0. The output of the 7430 goes to logic 1, which is sent to the 7404. The 7404 hex inverter transforms the output of the 7430 to logic 0, which it puts out on the IRQ (Interrupt Request) line (microprocessor expansion corrector pin E–4), and the micro knows that an impact sensor has been hit. The impact sensor detector ICs can be mounted on any board. If you have no board space, simply glue the ICs, pins up, on any part of the aluminum frame.

To determine which impact sensor has been hit, the computer uses the impact sensor selector (Fig. 5–9). The circuit uses a 74151 1-of-8 data selector. The microprocessor sets input/output (I/O) lines PB0, PB1, and PB2 to "000." If impact sensor 0 is hit, I/O line PB3 goes to "1." If PB3 stays at "0," the micro changes PB0–PB2 to "001" to see whether impact sensor 1 was hit. The micro continues in a like manner until all

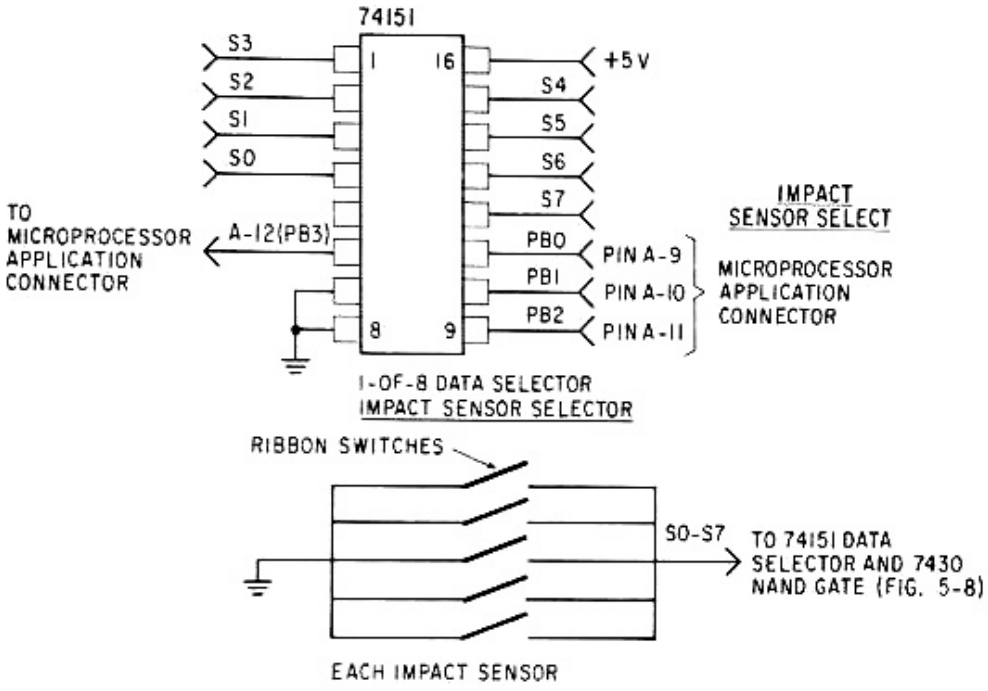
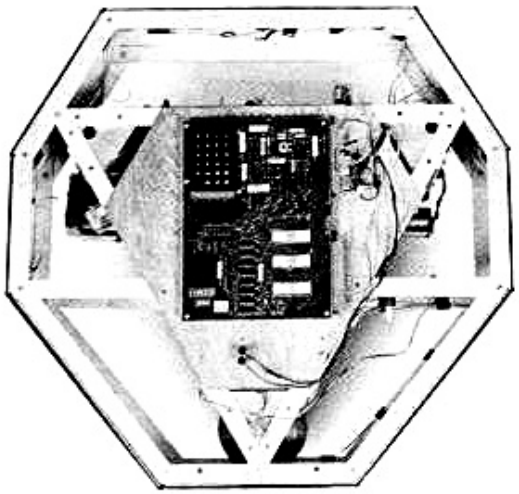


Fig. 5–9. Impact sensor selector.

the impact sensors have been tested. By testing all the impact sensors even after it has been determined that a sensor has been hit, the microprocessor can detect multiple as well as single hits. The impact sensor selector, like the impact sensor detector, should be mounted anywhere that you have extra room. This circuit also can be glued to the aluminum of the triangular frame.



software

The main purpose of the impact sensor control routine is to alert the microprocessor that an impact sensor has been hit, so that Mike can take the appropriate action. The routine works on interrupts (Chart 5-1). When an impact sensor is hit, the micro immediately turns off the motorized wheels. Then it determines which impact sensor or sensors were hit and stores their numbers in "S Byte."

After storing the number in "S Byte," the computer proceeds to the second section of the program (Chart 5-2). The computer compares the value in "S Byte" with the first column of the table in Table 5-3. If none of the values matches "S Byte," the micro determines that a sensor condition for which it has not been programmed has occurred. In this case, the micro stops Mike and lights the display. If one of the values in the impact sensor action table (Table 5-3) matches the value in "S Byte," the computer takes the two offset values in the second and third columns of the table. The directional control offset is stored in a byte called "X Point," and it tells the micro how many bytes, starting with

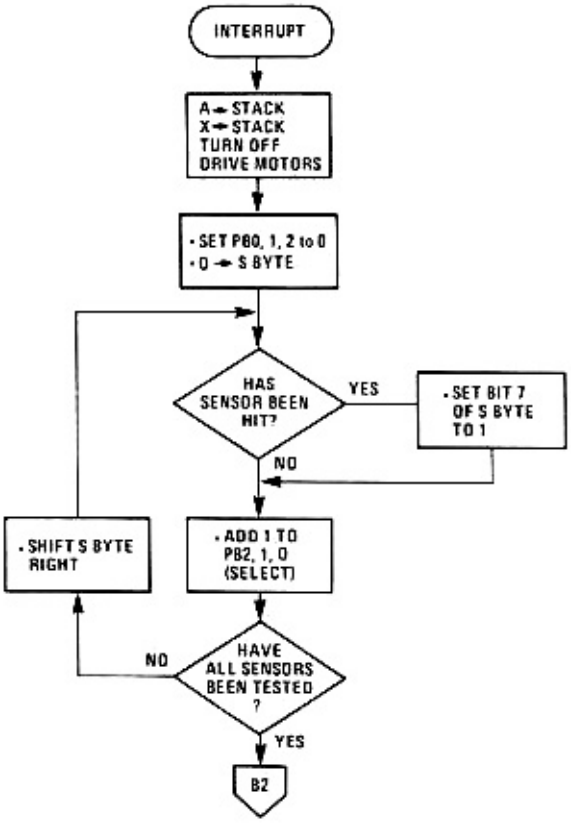


Chart 5-1.

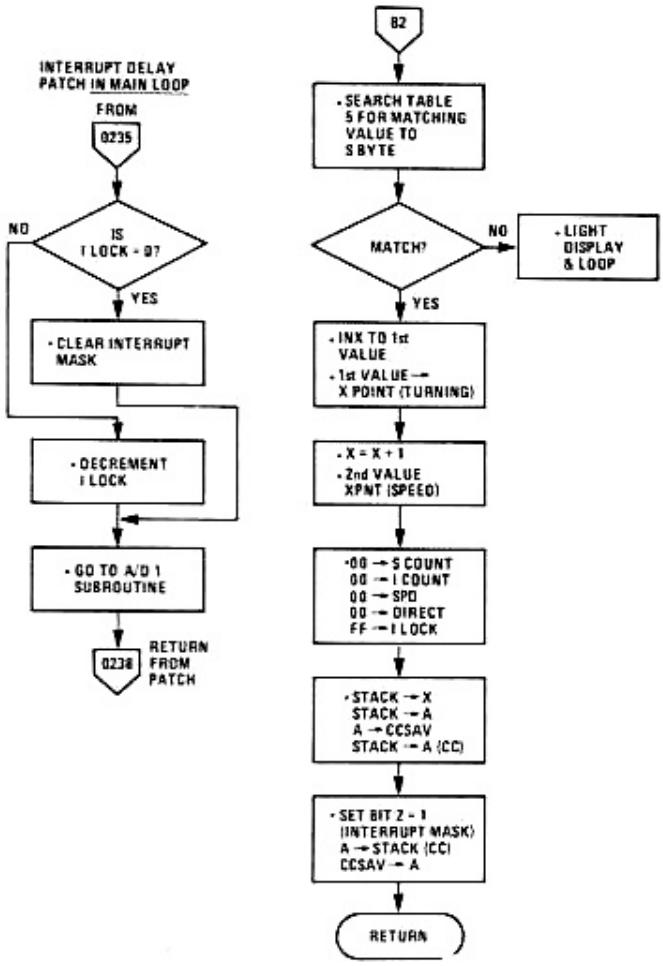


Chart 5–2.

Table 5–3. Impact Sensor Action Table

ADDRESS	*IMPACT SENSORS*	*DIRECTIONAL CONTROL OFFSET*	*SPEED OFFSET*	*SENSOR No.*
00B0	01	03	03	Sensor 0
00B3	02	03	03	Sensor 1
00B6	04	03	03	Sensor 2
00B9	08	06	08	Sensor 3
00BC	10	0B	0F	Sensor 4
00BF	20	0E	08	Sensor 5
00C2	40	13	14	Sensor 6
00C5	80	16	08	Sensor 7
00C8	50	1B	08	Sensors 4 & 6

"00," it must move down the directional control table before it starts executing directional control commands (Table 5–4). The speed offset is stored in a byte called "X Pnt," and it tells the computer how many bytes, starting with "00," it must move down the speed table before it starts executing speed commands (Table 5–5).

Next, the micro sets the interrupt mask and sets a counter so that the interrupts are disabled for about a second. This interrupt lockout prevents the processor from receiving repeated interrupts while the impact sensor remains depressed against an obstacle. If interrupts were not disabled, the micro would never get out of the interrupt routine long enough to move Mike away from an obstacle. After completing the routine, the micro returns to the point in the program at which the interrupt hit and goes back to what it was doing before the interrupt. When the computer reaches the self-directed routines, it causes Mike to move in accordance with the values placed in the directional control and speed tables by the impact sensor control routine (Table 5–6).

Table 5–4. Directional Control Table

ADDRESS	*DIRECTIONAL CONTROL VALUE*	*TIME*	*SENSOR No.*	*OFFSET*
0030	2A	02	Main Drive (no hit)	00
0032	FF			
0033	28	06	Sensor 0, 1, 2	03
0035	FF			
0036	2A	06	Sensor 3	06
0038	2B	06		
003A	FF			
003B	29	04	Sensor 4	0B
003D	FF			
003E	2A	06	Sensor 5	0E
0040	28	06		
0042	FF			
0043	2C	06	Sensor 6	13
0045	FF			
0046	2A	06	Sensor 7	16
0048	28	06		
004A	FF			
004B	2A	06	Sensors 4 & 6 (simultaneously)	1B
004D	27	06		
004F	FF			

Table can go to 006F

Table 5–5. Speed Table

ADDRESS	*INPUT VALUE*	*TIME*	*SENSOR No.*	*OFFSET*
0070	28	FE	Main Drive	00
0072	FF			
0073	2F	01	Sensor 0, 1, 2	03
0075	28	05		
0077	FF			
0078	0E	01	Sensor 3, 5, 7	08
007A	22	05	4 & 6	
007C	28	06	(simultaneously)	
007E	FF			
007F	0E	01	Sensor 4	0F
0081	22	03		
0083	FF			
0084	0E	01	Sensor 6	14
0086	22	05		
0088	FF			

Table can go to 00AF

Table 5–6. Impact Sensor Reactions

SENSOR #	*SPEED*	*DIRECTION*	*TIME*
Sensors 0, 1 & 2	Fast Forward	24° Right	1/2
	Slow Forward	24° Right	2 1/2
Sensor 3	Fast Reverse	Center	1/2
	Slow Reverse	Center	2 1/2
	Slow Forward	12° Left	3
Sensor 4	Fast Reverse	12° Right	1/2
	Slow Reverse	12° Right	1 1/2
Sensor 5	Fast Reverse	Center	1/2
	Slow Reverse	Center	2 1/2
	Slow Forward	24° Right	3
Sensor 6	Fast Reverse	24° Left	1/2
	Slow Reverse	24° Left	2 1/2
Sensor 7	Fast Reverse	Center	1/2
	Slow Reverse	Center	2 1/2
	Slow Forward	24° Right	3
Sensors 4 & 6	Fast Reverse	Center	1/2
	Slow Reverse	Center	2 1/2
	Slow Forward	36° Right	3

```
IMPACT SENSOR CONTROL ROUTINE
WRITTEN BY JOHN W. LOOFBOURROW

             ; Table of Added Storage Areas
             ; Needed for Impact Sensor Control
             ;
0006                *=$0006
0006         IMPACT *=*+1            impact cycle count
000C                *=$000C
000C         SBYTE  *=*+1            impact sensors hit
000D         ILOCK  *=*+1            interrupt lockout
000E         CCSAV  *=*+1            int. save area
             ;
00B0                *=$00B0
00B0         TABLE5 *=*+$30          action table
             ;
             ; Changes Required in Self Direction
             ; Programs to Activate Impact
             ; Sensor Control
             ;
02BB                *=$02BB
02BB 20 4203        JMP CONTIN      adds initialization
0295                *=$0295
0295 A9 07          LDA #$07        changes DDRB
0235                *=$0235
0235 4C 9003        JMP INTDEL      enable int. delay
0342                *=$0342
0342 A9 70   CONTIN LDA #$70        set IRQ
0344 8D FE17        STA $17FE       interrupt
0347 A9 01          LDA #$01        vector
0349 8D FF17        STA $17FF       address
034C 58             CLI             clear int. mask
034D 4C 3502        JMP SCAN3
             ;
             ; Interrupt Processing Routine
             ;
0170                *=$0170
0170 EA             NOP
0171 48             PHA             save A
0172 8A             TXA             A = X
0173 48             PHA             save A (X)
0174 AD 0017        LDA $1700       read PA
0177 09 08          ORA #$08        set PA3 = 1
0179 8D 0017        STA $1700       turn off spd cntrl.
             ; scan impact sensors
017C AD 0217        LDA $1702       PB0-2 = 0
017F 29 F8          AND #$F8
0181 8D 0217        STA $1702
0184 A9 00          LDA #$00        clear imp. sens.
0186 85 0C          STA SBYTE       hit byte
0188 46 0C   LOOPB  LSR SBYTE
018A A9 08          LDA #$08        set mask
```

```
018C 2C 0217          BIT $1702          test sensor
018F D0 0D            BNE HIT
0191 EE 0217 CONT9    INC $1702          add 1 to select
0194 AD 0217          LDA $1702
0197 29 07            AND #$07           check PB0-2
0199 F0 0C            BEQ DONE9          branch if = 0
019B 4C 8801          JMP LOOPB          not finished
019E A5 0C    HIT     LDA SBYTE          A = SBYTE
01A0 09 80            ORA #$80           bit 7 = 1
01A2 85 0C            STA SBYTE          SBYTE = A
01A4 4C 9101          JMP CONT9
                ; search table for XPOINT & XPNT
                ; values.
01A7 A2 00    DONE9   LDX #$00           set x to beginning
01A9 B5 B0    AGN9    LDA TABLE5,X       get sensor value
01AB C5 0C            CMP SBYTE          match to actual
01AD F0 0D            BEQ IHIT           branch if equal
01AF E8               INX                increment
01B0 E8               INX                to next
01B1 E8               INX                table value
01B2 E0 30            CPX #TABLE5+$30    check for end
01B4 D0 F3            BNE AGN9           branch if ≠ end
01B6 20 1F1F LOOPP    JSR SCANS          light display
01B9 4C B601          JMP LOOPP          loop - error
01BC E8       IHIT    INX                move pointer
01BD B5 B0            LDA TABLE5,X       get first value
01BF 85 09            STA XPOINT         steering pointer
01C1 E8               INX
01C2 B5 B0            LDA,TABLE5,X       get next value
01C4 85 0B            STA XPNT           speed pointer
                ; clean-up before returning
01C6 A2 00            LDX #$00
01C8 86 0A            STX SCOUNT         SCOUNT = 0
01CA 86 08            STX TCOUNT         TCOUNT = 0
01CC 86 05            STX TURN           TURN = 0
01CE 86 07            STX SPD            SPD = 0
01D0 CA               DEX
01D1 86 0D            STX ILOCK          ILOCK = $FF
01D3 68               PLA                restore
01D4 AA               TAX                X register
01D5 68               PLA                restore A
01D6 85 0E            STA CCSAV          save temporarily
01D8 68               PLA                get CC reg.
01D9 09 04            ORA #$04           set int. mask
01DB 48               PHA                put CC on stack
01DC A5 0E            LDA CCSAV          restore A
01DE 40               RTI
                ;
                ; Interrupt Delay Routine
                ;
0390                  *=$0390
0390 A5 0D    INTDEL  LDA ILOCK
0392 F0 08            BEQ RESTOR
```

```
0394 C6 0D          DEC ILOCK
0396 20 0002 LOOPA  JSR A/D1
0399 4C 3802        JMP $0238
039C 58      RESTOR CLI
039D 4C 9603        JMP LOOPA
                    END
```

chapter six ultrasonics

Parts List:

1 LM 1812 ultrasonic transceiver IC (see text)
1 7404 hex inverter IC
1 23-kHz transducer (see text)
2 inductors
  1 16–42 mH (Miller 6211)
  1 2–18 mH (Miller 6314)
2 potentiometers
  1 6 kΩ
  1 25 kΩ
5 resistors (all 1/2 watt)
  1 15 Ω
  1 100 Ω
  1 500 Ω
  1 20 kΩ
  1 27 kΩ
8 capacitors
  1 0.001 μF 50 V disc
  4 0.01 μF 50 V disc
  1 0.1 μF 50 V disc
  1 0.68 μF 50 V disc
  1 1 μF 15 V electrolytic
1 transistor
  1 2N2907
Perf board, 7 in. by 4 1/2 in.
2 3/16- by 1/2-in. stove bolts
2 3/16-in. nuts
2 3/16-in. lock washers
8 inches of 1 1/2- by 1/8-in. flat aluminum
1 tube instant bonding glue

In Stage II Mike is enabled to "see," using ultrasonic sound. His sight is accomplished by sending out a short burst of 23-kHz sound every quarter of a second. Since sound travels at approximately one foot per

millisecond, Mike can determine the distance of an object in his path by counting the number of milliseconds required for the sound to be reflected back from an object. In order for the ultrasonic sound to reach an object and return to Mike, it must travel twice as far as the distance between Mike and the object. The amount of time required to receive a reflection from an object increases by 2 ms for every foot of Mike's distance from it.

The problem of designing a non-tactile sensory system puzzled me for some time. The first method that I tried was proximity detection. The proximity sensor detected an increase in capacitance when Mike approached an object. Unfortunately, this method detected only objects with capacitance, and it could not detect wood, plastic, or any other non-conducting material. The next sensory system that I tried was infrared detection. This method used an infrared light-emitting diode (LED), which sent out a continuous beam of infrared light. The circuit's infrared detector sensed infrared light whenever Mike approached an object. This system had two major drawbacks. First of all, when Mike "saw" an object, he didn't know its distance from him. Also, the infrared light would not reflect off of a dark-colored surface. My last attempt at designing a non-tactile sensory system was ultrasonic detection. Fortunately, this method worked.

ultrasonic detection circuitry

The ultrasonic detection circuit allows the micro to detect objects in Mike's path (Fig. 6–1). The circuit uses an LM 1812 ultrasonic transceiver (see Appendix). This IC contains a transmitter and a receiver. The transmitter can be activated by using standard logic inputs. As soon as the transmitter is turned off, the receiver is activated. The receiver puts out a logic 1 on pin 14 whenever it detects a 23-kHz tone.

The 2N2907 transistor shown in the circuit is used as a pulse stretcher. The purpose of this transistor is to lengthen the short duration pulses coming from the LM 1812. These lengthened pulses provide greater energy to the 23-kHz transducer for greater signal strength. I chose a 23-kHz transducer because the lower ultrasonic frequencies don't attenuate as fast and can attain greater distance (see Appendix).

The ultrasonic detection circuit is built on a piece of perf board 7 in. by 4½ in. Leave a 1-in. border on all sides of the board. You may have to solder wires on the potentiometers and inductors in order to attach them to the perf board. Now build the circuit, as shown in Fig. 6–1.

The ultrasonic detection circuit board is mounted between the vertical face of piece 10 on the upper section of the outer frame and the

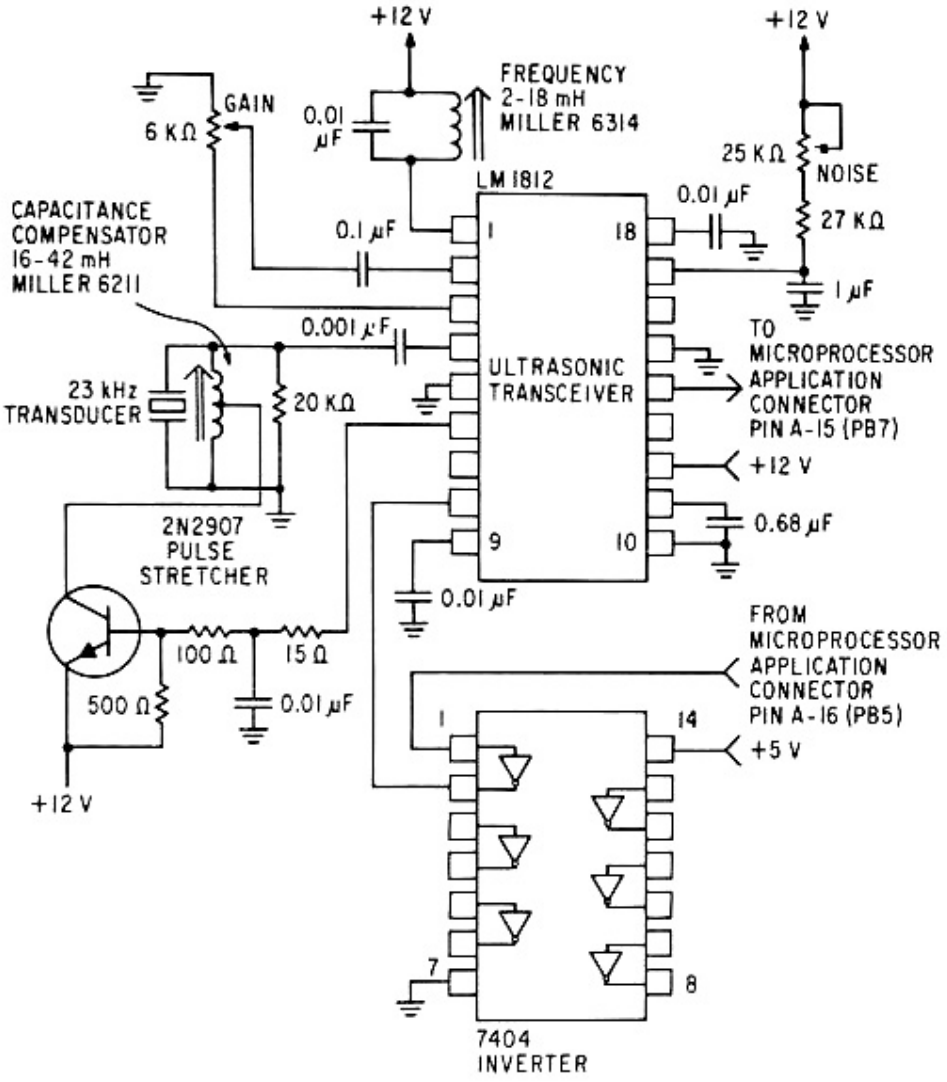
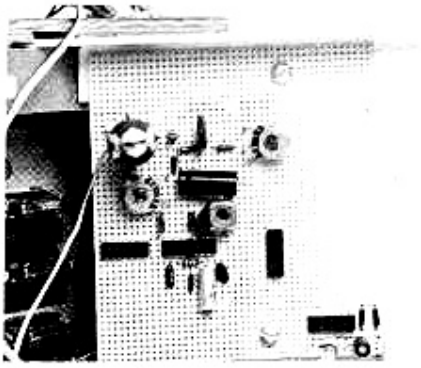


Fig. 6–1. Schematic of ultrasonic detection circuit.



vertical face of piece 10 on the lower section of the outer frame. Bolt the circuit to the frame with the 7-in. sides of the board being vertical. Make sure that the pots and inductors on the board are easily accessible for tuning. Carefully route all wires going to the micro so that they do not come within 2 in. of the power supply circuit, where they could pick up noise.

The 23-kHz transducer is mounted directly below impact sensor 5 (Fig. 6–2). Cut a piece of aluminum 4 in. by 1 in. out of 1½- by ⅛-in. flat aluminum. Now cut from your 1½- by ⅛-in. flat aluminum two pieces, each 2 in. by 1 in. As shown in Fig. 6–2, glue these pieces to the front of your 4- by 1-in. piece, using instant bonding glue. Next, glue the transducer to the center of the 4-in. piece. You will have to drill two ⅛-in. holes in the aluminum to allow room for the leads of the transducer. The protruding ends of the two 2-in. pieces should be bonded to the back of piece 5 of the lower frame.

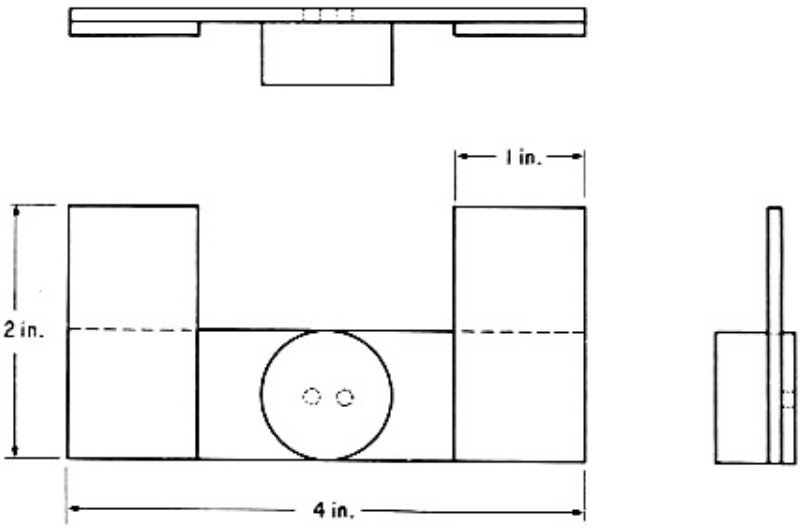


Fig. 6–2. Mounting ultrasonic transducer.

The ultrasonic detection circuit is adjusted by the following procedure:

Step 1—Turn both potentiometers to maximum.

Step 2—Attach an oscilloscope probe to pin 4 of the LM 1812.

Step 3—Turn the transmitter on by grounding pin 1 of the 7404 hex inverter.

Step 4—Adjust the inductor attached to pin 1 of the LM 1812 for maximum amplitude on the oscilloscope.

Step 5—Adjust the inductor across the transducer until maximum amplitude is obtained on the oscilloscope.

Further adjustments will be required when the micro is attached to the circuit.

software

The ultrasonic detection program is designed to activate the transmitter to send out a short burst of 23-kHz sound (Chart 6-1). After turning off the transmitter, the micro delays until the transmit signal has decayed. Next the computer waits for a predetermined period of time

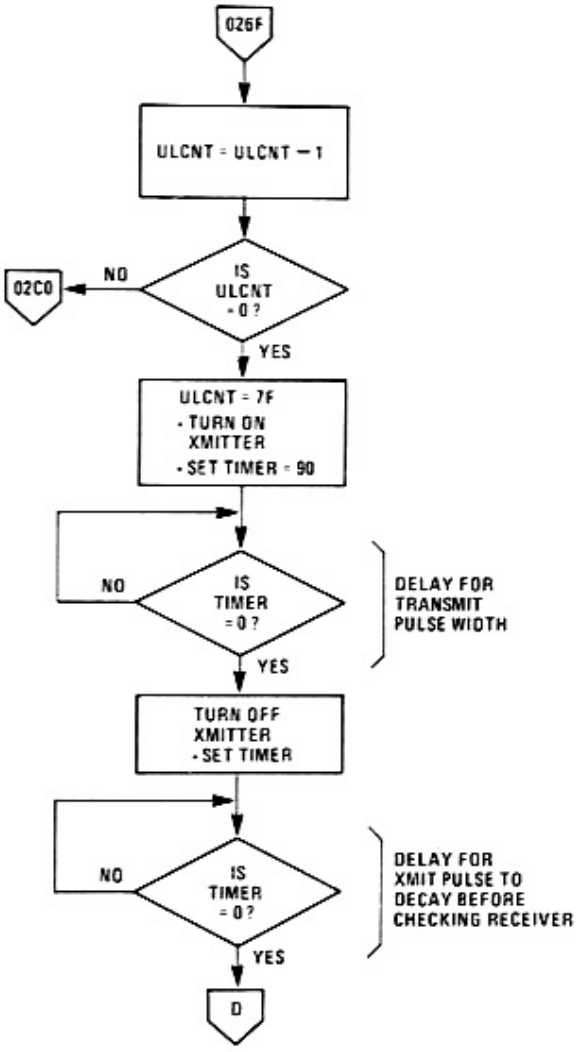


Chart 6–1.

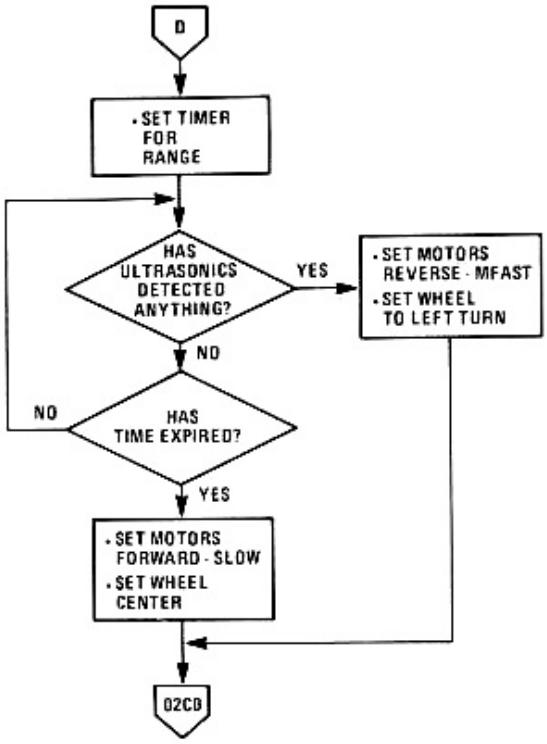


Chart 6-2.

for a reflected signal to be received (Chart 6-2). The amount of time that the computer waits and, consequently, the range of the ultrasonics can be varied by changing the data at location "03BB." If no signal is received within the allotted time, the micro assumes that there is no obstacle in Mike's path. Then the program sets the speed at "slow-forward" and centers the front wheel. On the other hand, if a reflected signal is received within the allotted time, the computer turns the front wheel to the left and causes Mike to move backward.

Since the ultrasonic transducer is activated every quarter of a second, a counter must be inserted in the main loop. This counter (called ULCNT) starts at "7F" and is decremented every time the computer goes through the main loop. When the counter reaches "00," the computer can enter the ultrasonic detection program and activate the transmitter. By changing the data in address "03E7," the pulsing rate of the ultrasonic transducer can be varied. By increasing the value in location "03E7," the ultrasonic transducer will be activated less frequently. Mike's response to "seeing" an object can be varied by changing the data in locations "03D5" and "03D9."

```
ULTRASONIC DETECTION PROGRAM
WRITTEN BY JOHN W. LOOFBOURROW


                ; Table of Added Storage Required
                ; by Ultrasonics
                ;
000F                    *=$00F
000F            ULCNT   *=*+1
                ;
                ; Changes to Impact Sensor Program
                ; Required for Ultrasonics
                ;
0295                    *=$0295
0295 A9 27              LDA #$27        sets DDRB
026F                    *=$026F
026F 4C A003            JMP ULTRA
                ;
                ; Main Ultrasonic Program Scan
                ; Loop
                ;
03A0                    *=$03A0
03A0 4C DF03 ULTRA      JMP CKCNT
03A3 29 DF   ULOK       AND #$DF        PB5 = 0
03A5 8D 0217            STA $1702       (turn on xmit)
03A8 A9 90              LDA #$90        xmit pulse width
03AA 8D 0617            STA $1706       store in ÷ 64 count
03AD AD 0417 LOOP4      LDA $1704       read timer
03B0 30 FB              BMI LOOP4       loop if not time yet
03B2 AD 0217            LDA $1702       turn off xmitter
03B5 09 20              ORA #$20
03B7 4C F003            JMP ULWAT
03BA A9 05   UCONT      LDA #$05        set timer range
03BC 8D 0717            STA $1707       store in ÷ 1024
03BF 2C 0217 LOOP2      BIT $1702       timer.  test port
03C2 10 10              BPL DONE        PB7 = 0?
03C4 AD 0417            LDA $1704       check timer
03C7 10 F6              BPL LOOP2       time up?
03C9 A9 28              LDA #$28        no target - set spd.
03CB 85 70              STA TABLE4
03CD A9 2A              LDA #$2A        turn = center
03CF 85 30              STA TABLE3
03D1 4C C002 RET1       JMP SCAN
03D4 A9 1C   DONE       LDA #$1C        target - rev. spd.
03D6 85 70              STA TABLE4
03D8 A9 2B              LDA #$2B        set turn w/target
03DA 85 30              STA TABLE3
03DC 4C C002            JMP SCAN
                ;
                ; This routine determines how
                ; often to send the pulse.
                ;
03DF C6 0F   CKCNT      DEC ULCNT
```

```
03E1 F0 03          BEQ TSTULT
03E3 4C C002        JMP SCAN
03E6 A9 7F   TSTULT LDA #$7F        set count for next
03E8 85 0F          STA ULCNT       cycle.
03EA AD 0217        LDA $1702
03ED 4C A303        JMP ULOK
                    ;
                    ; This routine sets the minimum
                    ; range of the ultrasonics.
                    ;
03F0 8D 0217 ULWAT  STA $1702
03F3 A9 03          LDA #$03        set minimum range
03F5 8D 0717        STA $1707       store in * 1024 cnt.
03F8 AD 0417 LOOP7  LDA $1704
03FB D0 FB          BNE LOOP7
03FD 4C BA03        JMP UCONT
                    END
```

tuning the ultrasonic detection circuit

The ultrasonic detection circuit requires a certain amount of tuning after loading the software. Start the program at location "0290." Make sure that the disable switch is turned off because the ultrasonic detection program will move Mike in reverse and turn the front wheel to the left. This action is caused by the fact that the receiver is set on its highest possible gain.

To eliminate this problem, go through the following tune-up procedure:

Step 1—Turn down the 6 kΩ pot until the ultrasonic detection circuit responds by centering the front wheel. Be sure that there are no objects within the range of the ultrasonics (5 ft.)

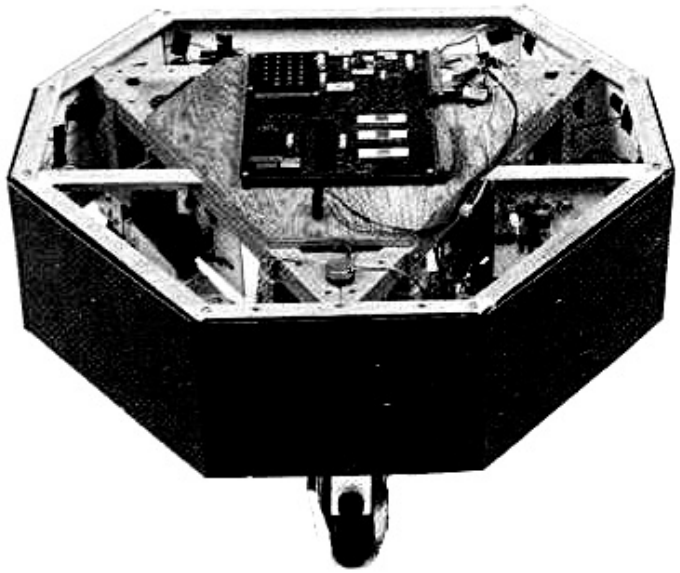
Step 2—Put an object about 1 in. in front of Mike's ultrasonic transducer. Mike should respond by turning the front wheel to the left.

Step 3—Turn down the 25 kΩ pot until the ultrasonic detection circuit is reliably detecting objects in front of the transducer. You may need to readjust the 6 kΩ gain pot in order to optimize the circuit's operation.

Mike's ultrasonic "sight" is complete.

After tuning the ultrasonic detection circuit, I placed Mike on the floor of the basement and let him move about. I found that Mike was very adept at finding his way around a basement crowded with obstacles. After thoroughly exploring the main room of the basement, Mike confidently exited through the only open doorway.

At this point Stage II will have been successfully completed.



chapter seven voice recognition

Voice Recognition Parts List:

1 Econoram II® 8 K memory board (see text)
1 74151 IC 1-of-8 data selector
1 2.2 kΩ resistor
1 7406 open collector hex inverter

Voice Recognition Circuit Parts List:

Perf board, 7 in. by 4½ in.
1 crystal microphone
2 LM 3900 quad operational amplifier IC
9 capacitors
- 2 300 pF 50 V disc
- 3 470 pF 50 V disc
- 3 0.1 μF 50 V disc
- 1 1 μF 50 V electrolytic

20 resistors (all ½ watt)
- 1 560 Ω
- 2 1kΩ
- 1 10 kΩ
- 1 11 kΩ
- 1 100 kΩ
- 1 270 kΩ
- 10 1 MΩ
- 1 2.7 MΩ
- 2 10 MΩ

2 3/16- by ½-in. stove bolts
2 3/16 in. nuts
2 3/16 in. lock washers
A 2- by 2-in. square of ½-in. foam rubber

In Stage III Mike gains the ability to recognize spoken words. Voice recognition is one of Mike's most awesome features and one that is the most fun to develop. Before the addition of voice recognition, the only way to control Mike was by attaching his joystick control and loading in the joystick control program. Now Mike can be controlled merely by talking to him.

The voice recognition does not take away in any manner from Mike's independence. When Mike recognizes a command, the range of his ultrasonics is shortened so that he reacts only to obstacles less than 2 ft away. His sense of "touch" (impact sensors) has an even higher priority than your commands. This hierarchy prevents Mike from running headlong into a wall in compliance with one of your commands.

Mike is able to recognize up to eight different words. Each word can command him to respond in a different way. Surprisingly, everything that you will need to accomplish the voice recognition costs only about $10.

voice recognition circuitry

The voice recognition system uses zero-crossing analysis. This technique permits simple measurement of the frequency of a given waveform. Figure 7–1 shows a typical voice waveform. Notice the horizontal line, labeled "0 axis," going through the center of the waveform. The position of this line is determined by averaging all the points of the waveform. The zero-crossing analysis technique measures the number of times the waveform crosses the zero axis.

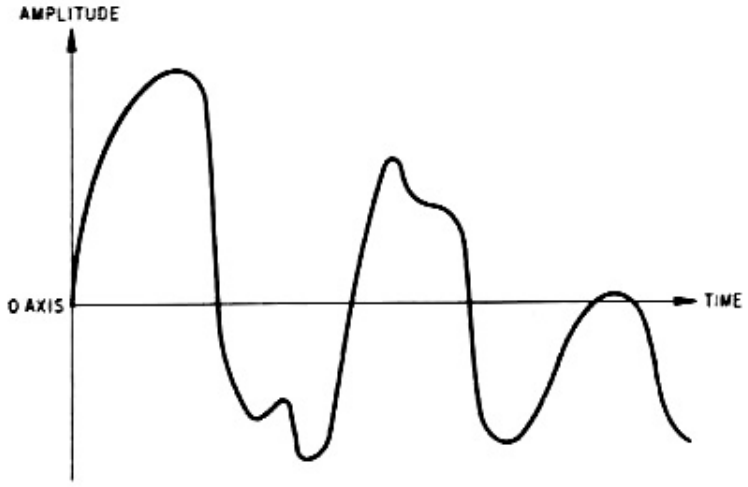


Fig. 7–1. Typical voice waveform.

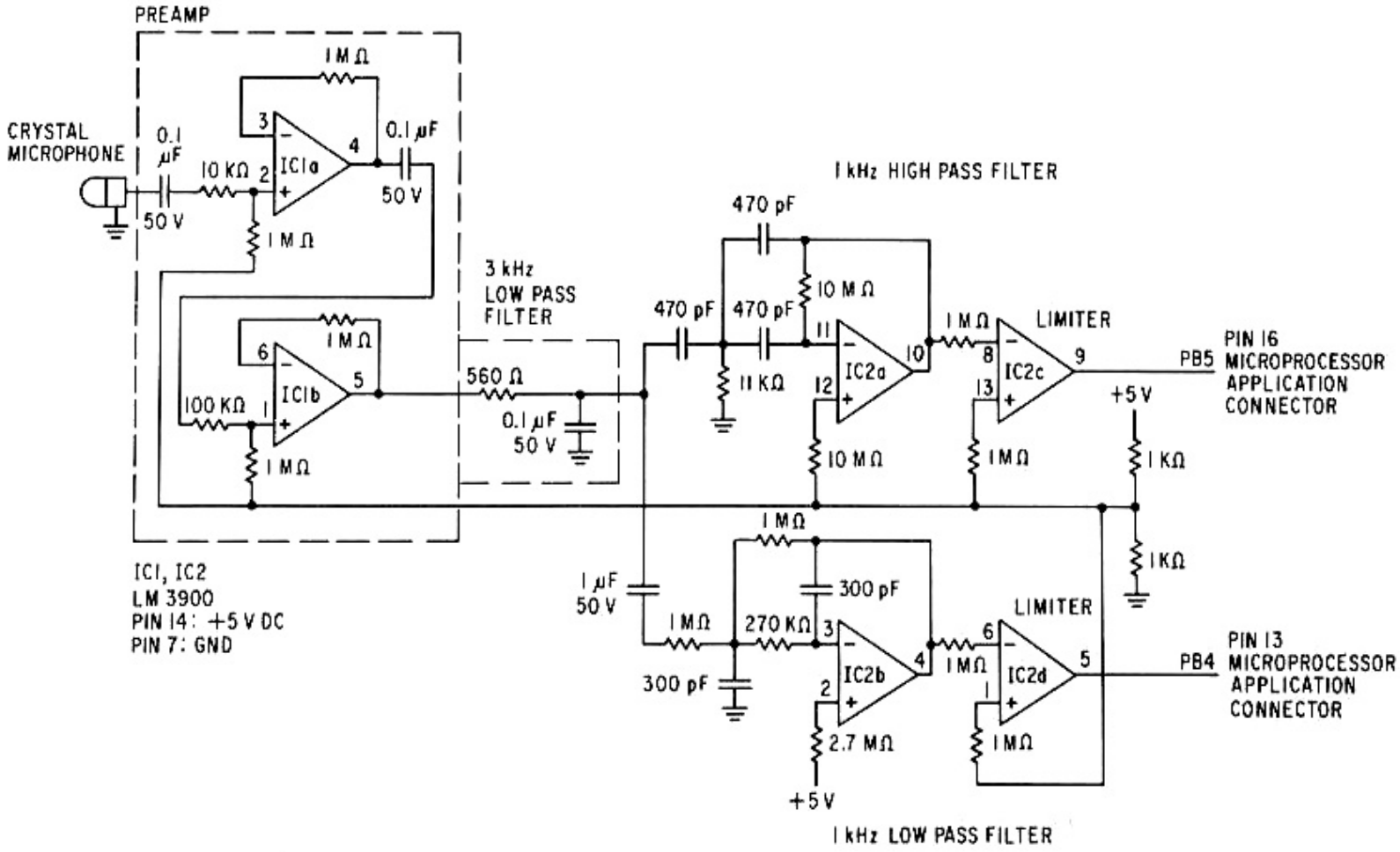


Fig. 7–2. Voice recognition circuit.

The voice recognition circuit begins with a microphone (Fig. 7–2). Sound entering the microphone generates electrical energy, which is amplified by a preamp. Then the amplified energy is sent to the high pass and low pass filters. Sound energy above approximately 1 kHz is sent to the high pass filter. The remainder is sent to the low pass filter. Following each filter is a limiter, which prepares the signal for microprocessor zero-crossing detection.

The output of the high pass filter might look something like Fig. 7–3. The vertical lines indicate 160 μs intervals. Every 160 μs the computer samples the output of the high pass filter to determine whether it is above or below the zero axis. Every time the output crosses the zero axis, the computer increments a counter. The output of the low pass filter might look something like Fig. 7–4. The output is treated in exactly the same way as the high pass filter output.

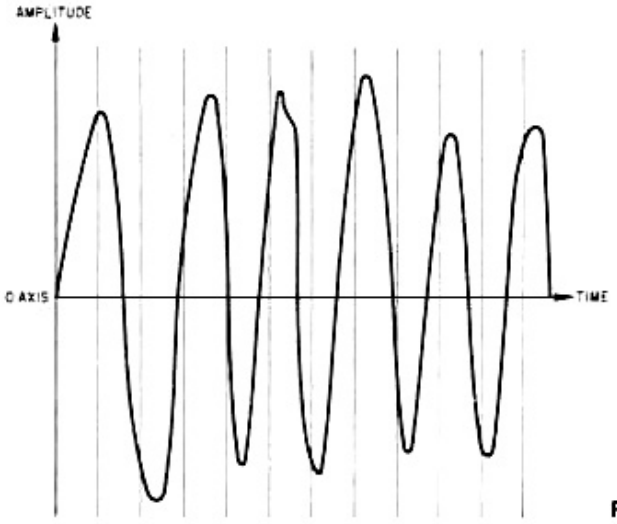


Fig. 7–3. High pass filter output.

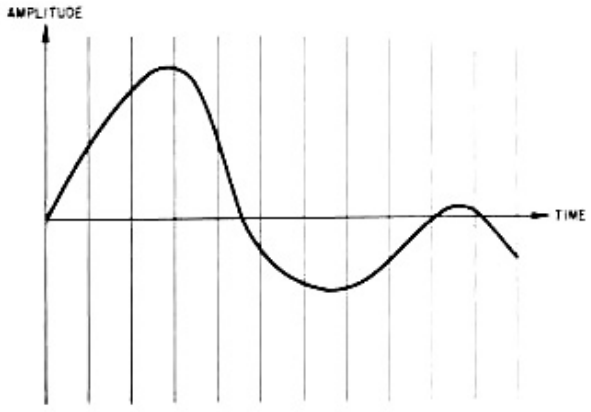


Fig. 7–4. Low pass filter output.

The voice recognition circuit is built on a piece of 7- by 4½-in. perf board. Leave a 1-in. border on all sides. The board is mounted between piece 9 on the upper section of the outer frame and piece 9 on the lower section of the outer frame. With the 7-in. side vertical, bolt the perf board exactly as you did when mounting the ultrasonic detection circuit. The only component that is not mounted on the perf board is the microphone. The back of the microphone should be attached to a piece of material (such as foam rubber) that will transmit as little vibration as possible. This material should be glued to the hexagonal piece of plywood on which the Kim is mounted. The microphone should be facing upward.

software

Because of the specialized requirements of the voice recognition system, the software you have used in Mike up to this point must be abandoned. The entire strategy of the software will be changed. You will implement a new software system that fully utilizes the interrupt capability of the Kim in order to meet the stringent time demands of voice recognition.

Before reworking the software, you should make several additions and changes in the hardware. The interrupt software requires more than 1 K of memory. Therefore, more memory must be added to the Kim in order to accommodate the program. The entire program can be contained in 4 K of memory. I decided to use an 8 K board to allow memory for Mike's future functions. I chose the Econoram II® (see Appendix). The Econoram II® is an "S-100 buss compatible" memory board. My reasons for choosing an S-100 compatible board were its low cost and its ready availability. Several jumpers are required on the Econoram II® in order to interface it with the Kim (Table 7-1). I used 30-gauge wire-wrapping wire to make the jumpers. In addition to adding jumpers, you will need to use one 7406 open collector hex inverter (Fig. 7-5). Notice that the ground connection on the application connector pin K has been removed, and a jumper has been placed to expansion connector pin R. To attach the 8 K memory board to the Kim, use wire jumpers, as shown in Table 7-2 and Figures 7-6 and 7-7. The 7805 regulators on the memory board require 7 to 9 V for proper operation. A 3.3 Ω 10 W resistor is

Table 7-1. Jumpers required on Econoram II®

1)	Pin 2	IC5	to	Pin 8 IC11
2)	Pin 15	IC1	to	GND
3)	Pin 1	IC1	to	GND
4)	Pin 5 & 2	IC4	to	+ 5 V
5)	Pin 12	IC11	to	GND

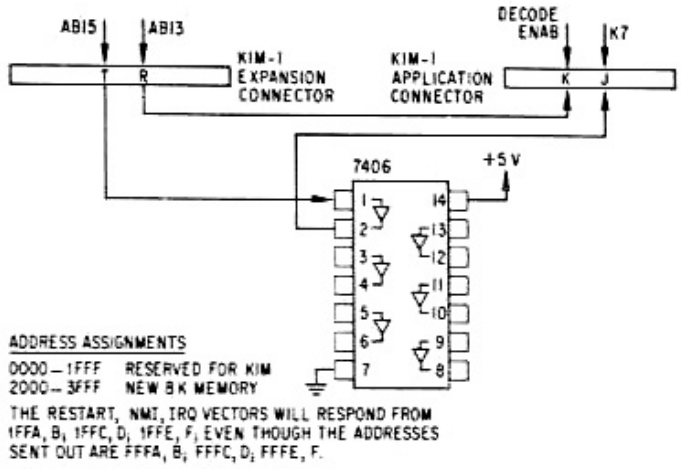


Fig. 7–5. Inverter and address assignments.

Table 7–2. Connections to 8K Econoram II® Memory Board

KIM–1 EXPANSION CONNECTOR		*ECONORAM II® MEMORY BUSS CONNECTOR*	
PIN No.	**PIN DESIGNATION**	**PIN No.**	**PIN DESIGNATION**
A	AB0	79	A0
B	AB1	80	A1
C	AB2	81	A2
D	AB3	31	A3
E	AB4	30	A4
F	AB5	29	A5
H	AB6	82	A6
J	AB7	83	A7
K	AB8	84	A8
L	AB9	34	A9
M	AB10	37	A10
N	AB11	87	A11
P	AB12	33	A12
R	AB13	85	A13
S	AB14	86	A14
T	AB15	32	A15
15	DB0	95 & 36	D0
14	DB1	94 & 35	D1
13	DB2	41 & 88	D2
12	DB3	42 & 89	D3
11	DB4	91 & 38	D4
10	DB5	92 & 39	D5
9	DB6	93 & 40	D6
8	DB7	43 & 90	D7
V	R/W	47	$\overline{\text{SMEMR}}$
Z	$\overline{\text{RAM}}$ R/W	77	$\overline{\text{PWR}}$
Y	$\overline{\text{Ø2}}$	25	Ø1

Note: Pins 1 and 51 = +9V, using 3.3Ω 10w from +12 V
Pins 50 and 100 = ground.

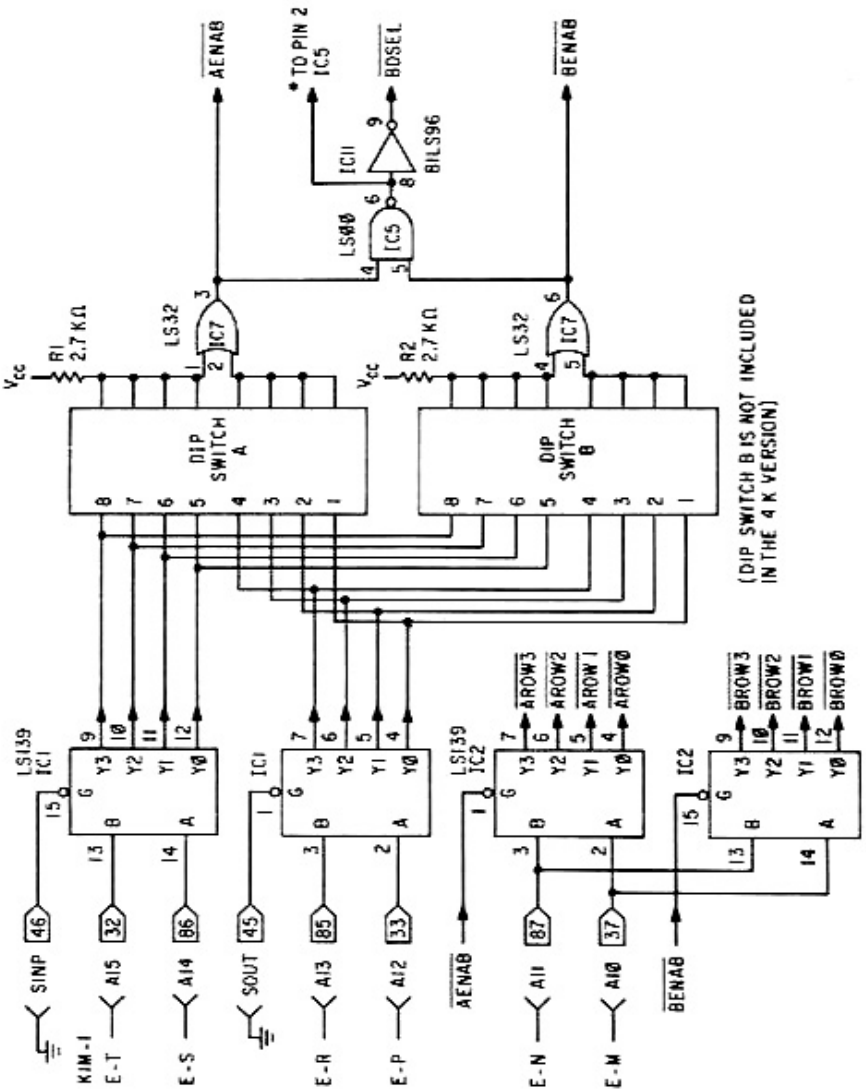


Fig. 7–6. Schematic of Econoram II™ 8 K memory board interfaced with Kim-1 (top).

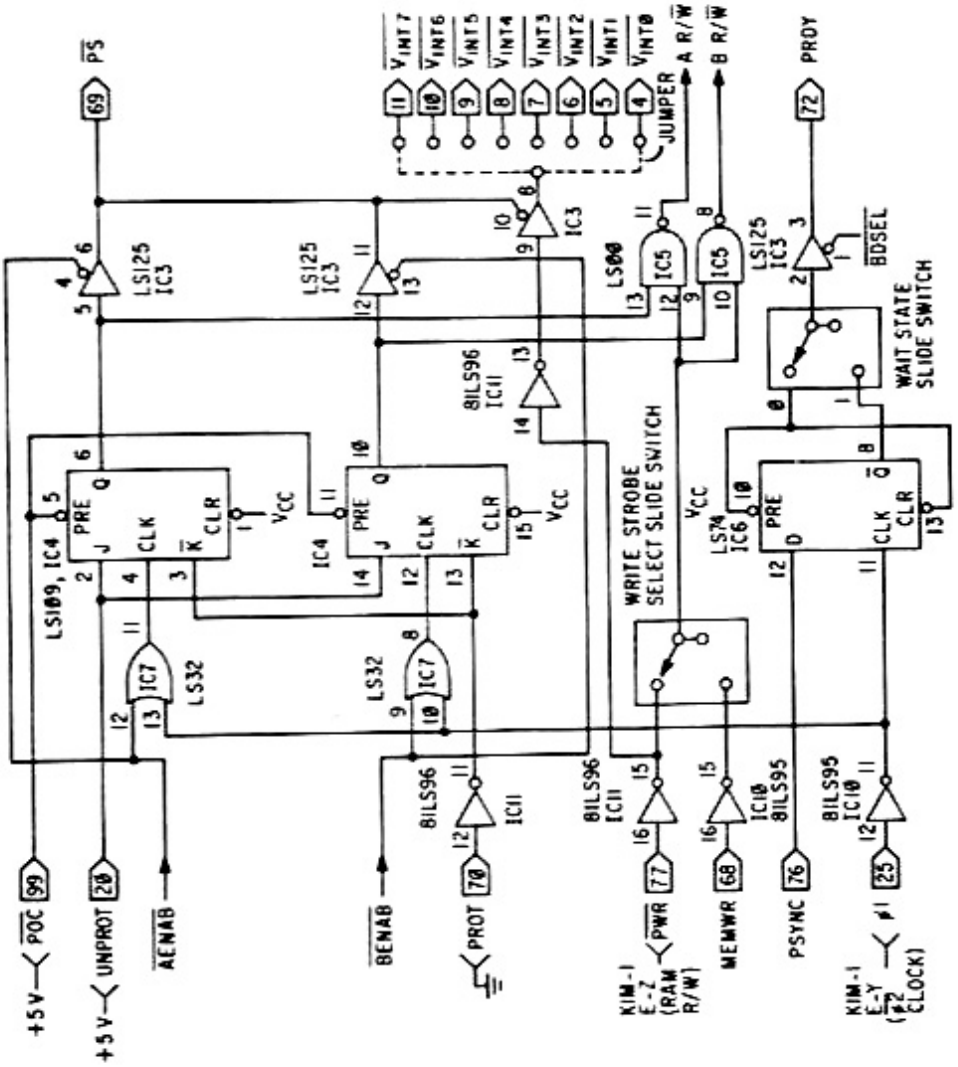


Fig. 7–6 (bottom). Reprinted with permission of Bill Godboul Electronics.

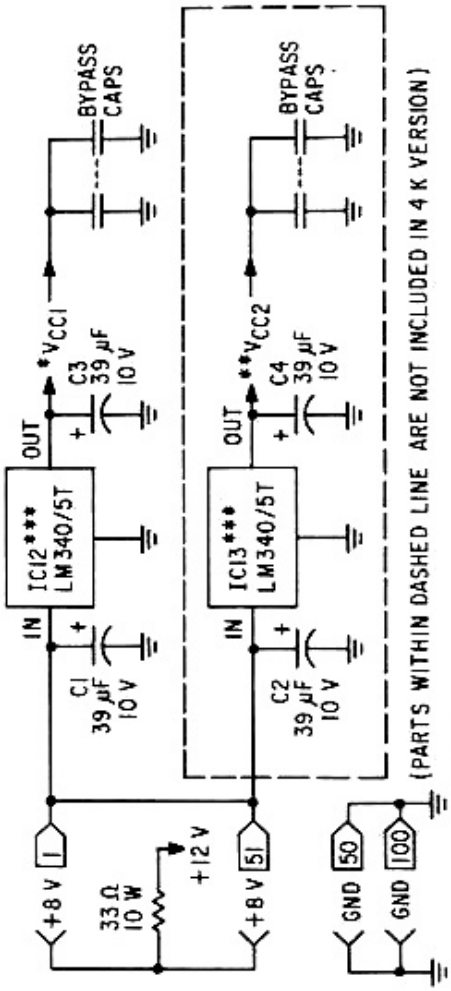


Fig. 7-7. Continuation of Fig. 7-6 (top).

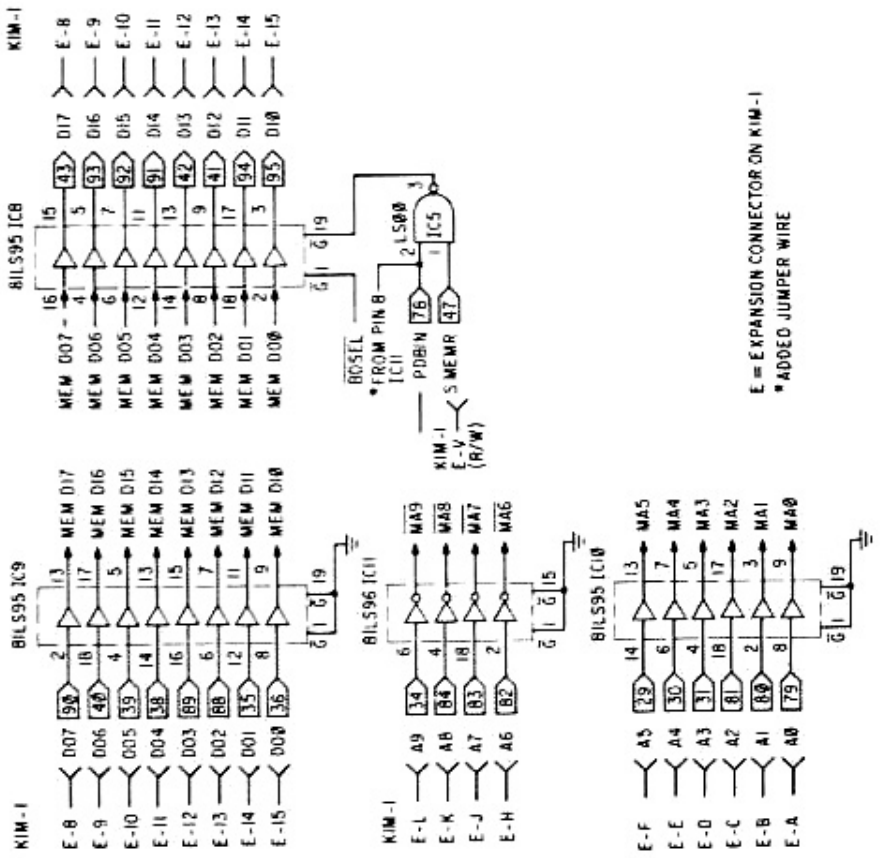
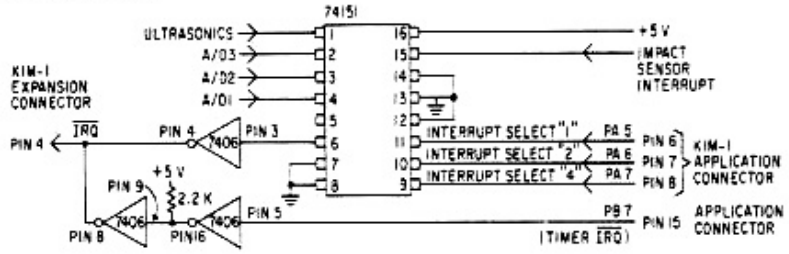


Fig. 7–7 (bottom). Reprinted with permission of Bill Godbout Electronics.

required to reduce the +12 V supply from the battery to the required voltage. The 8 K memory board can be mounted above the Kim. I anchored the connectors of the Kim and the memory board on sheet aluminum. The edges of the memory board are held by card guides, supported by angle aluminum. When you mount the 8 K board, leave about 1 in. between it and the Kim for cooling purposes.

The next hardware addition is an interrupt selection circuit (Fig. 7–8). The circuit uses a 74151 1-of-8 data selector. The circuit is used by the microprocessor to allow selection of external interrupts. Notice that the 74151 is connected to the 7406 that is shown in Fig. 7–5. The interrupt selection circuit can be mounted on any piece of perf board with extra room.



INTERRUPT SELECTION

Select Code		
000	Analog-to-Digital 1	Directional Control Pot
001	Analog-to-Digital 2	Speed Pot
010	Analog-to-Digital 3	Directional Control Command Pot
011	Ultrasonics	
100	Impact Sensor Interrupt	
101	Future	
110	Future	
111	Future	

Fig. 7–8. Interrupt selection.

Because of the requirements of the voice recognition system, the application connector pin assignments must be revised (Table 7–3). Notice that the A/D inputs, the ultrasonics input, and the impact sensor interrupt have been transferred to the interrupt selection circuit. Now you're ready for the software.

The new software breaks up the processing into "tasks." The tasks are scheduled to be executed by a master scheduler program. The scheduler program can start a task at a specific time. These tasks are called "timed tasks." The scheduler can also activate a task that had been initiated by an external event. Such tasks are called "scheduled tasks." The voice recognition software makes use of the timed task feature of the scheduler.

Table 7–3. Application Connector—I/0 Assignments

PIN No.	*PORT*	*IN/OUT*	*FUNCTION*
16	PB5	In	voice analysis input (high)
15	PB7	In	timer interrupt
14	PA0	Out	A/D capacitor
13	PB4	In	voice analysis input (low)
12	PB3	In	I. Sensor select (input)
11	PB2	Out	I. Sensor select
10	PB1	Out	I. Sensor select
9	PB0	Out	I. Sensor select
8	PA7	Out	interrupt select "4"
7	PA6	Out	interrupt select "2"
6	PA5	Out	interrupt select "1"
5	PA4	Out	forward/reverse
4	PA1	Out	left
3	PA2	Out	right
2	PA3	Out	go/stop

Mike's former software made use of a main loop to control his timing. The new software uses instead the Kim–1 interval timer, which is set to cause an interrupt every 160 μs. The 160-μs interrupt is used by the voice recognition system to check the high and low pass lines for a "1" or a "0." If either input has changed from the last sample, the micro increments either the "VOICE H" or the "VOICE L" counter. Every 16 ms (100 timer intervals), the computer takes the count in "VOICE H" (Voice High) and categorizes it into one of five groups (Table 7–4). The computer categorizes the count in "VOICE L" (Voice Low) into one of six groups (Table 7–5). The software uses the group numbers from this

Table 7–4. Voice Analysis—High Pass Groupings

GROUP	*No. OF ZERO CROSSINGS*	*HEXADECIMAL*
1	0–32	0–20
2	33–48	21–30
3	49–64	31–40
4	65–80	41–50
5	81+	51+

Table 7–5. Voice Analysis—Low Pass Groupings

GROUP	*No. OF ZERO CROSSINGS*	*HEXADECIMAL*
1	0–6	0–06
2	7–13	07–0D
3	14–19	0E–13
4	20–25	14–19
5	26–32	1A–20
6	32+	20+

point forward. Coordinates are determined by the low and high group numbers (Fig. 7-9). The memory location represented by this pair of coordinates is increased by one. For example, suppose that the low group number is 2 and the high group number is 1. The coordinates determined by the groups are (2,1). The byte in memory that represents the coordinates (2,1), the sixth byte from the beginning of the table, is incremented. The incoming coordinate samples are used to increment the byte that they represent in a table called "BUFFER"—Fig. 7-7.

HIGH GROUP						
5	BYTE 5 00	BYTE 10 00	BYTE 15 01	BYTE 20 00	BYTE 25 00	BYTE 30 00
4	BYTE 4 00	BYTE 9 01	BYTE 14 00	BYTE 19 00	BYTE 24 02	BYTE 29 04
3	BYTE 3 00	BYTE 8 00	BYTE 13 02	BYTE 18 01	BYTE 23 0C	BYTE 28 17
2	BYTE 2 00	BYTE 7 00	BYTE 12 00	BYTE 17 00	BYTE 22 01	BYTE 27 09
1	BYTE 1 03	BYTE 6 01	BYTE 11 00	BYTE 16 00	BYTE 21 00	BYTE 26 00
	1	2	3	4	5	6

LOW GROUP

Fig. 7-9. "BUFFER" Table. Count numbers are hexadecimal. Byte numbers refer to bytes in "BUFFER" Table only, not addresses.

After 0.92 s (60 coordinate samples), the program begins the process of checking to see whether it has received a recognizable command. The program checks by taking each byte in the "BUFFER" and comparing it with the corresponding byte in a previously stored word template. The program subtracts the smaller value from the larger value to ensure that the result is always positive. The results of these subtractions are added together to form a "match count." Each of the eight templates is compared with the "BUFFER" to obtain a match count. If the lowest match count is less than a predetermined threshold, the word represented by the match count is recognized as the command spoken.

Templates are word patterns generated from spoken commands. These must be stored in memory before you attempt to run the voice

recognition program. The templates contain the words for which Mike will listen. By changing two addresses (noted in a comment at the beginning of the template generation routine), you can activate the template generation routine. Start the routine at address "2000." When the display blanks, say your first word. The display will light after 0.92 s. It will remain lit for 1 s. When it blanks again, repeat the same word in about the same tone. Your two samples are averaged together to form the first word template. After the first template is formed, press "RESET." Now press "GO" to form your second word template. Repeat the procedure until all eight templates are formed. The templates are numbered 0 to 7 consecutively. When a command is recognized, Mike briefly displays the template number of the matched word in the rightmost LED display. Then he continues listening for another word.

The tasks control Mike's functions. The interrupt-driven "timed tasks" are A/D 1, A/D 2, A/D 3, ultrasonics (in part), and impact sensors. These tasks generate interrupts when the attention of the microcomputer is required. The timed tasks that are not interrupt driven are voice analysis, speed control, directional control, and ultrasonics (in part). Each of these tasks is scheduled for execution when its timer so indicates. The remaining tasks are scheduled tasks. These tasks are initiated by external events, which activate the processing. These tasks are ultrasonic analysis, impact sensor analysis, no-response analysis, and voice command analysis.

Flowcharts for the entire program are shown in Charts 7–1 through 7–21. Since the software is presently in progress, many of the task flowcharts have not yet been converted into programs. By examining the flowcharts and the programs completed thus far, you should be able to determine how to write the software to complete the remaining tasks. You will notice that the tasks' names are already defined in the task tables, shown in the Voice Recognition Program. The names are INTTSK (interrupt driven tasks), TASKST (timed tasks), and TASKSS (scheduled tasks). The task address of a task is the location where the scheduler jumps when the task is initiated. Any task that has not been written has as its task address the scheduler routine. When you have written the software for a particular task, you should change the task address to the starting address of your task.

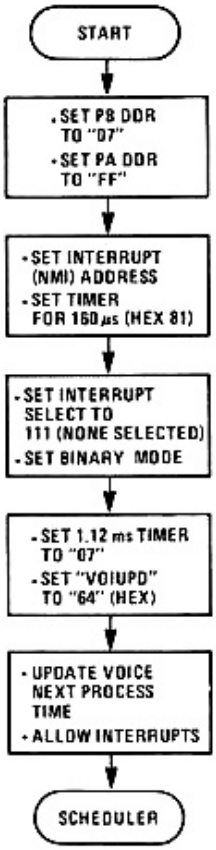


Chart 7–1.

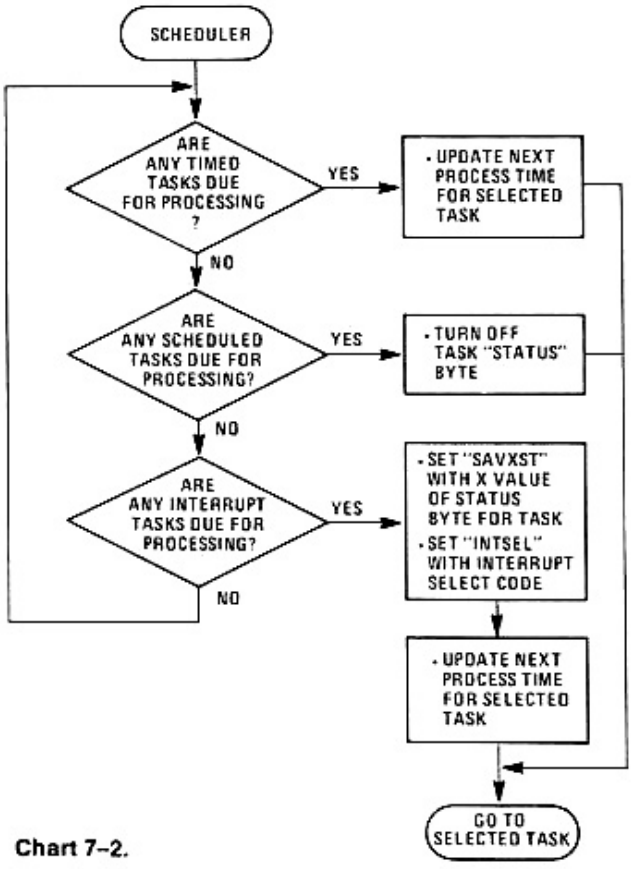


Chart 7–2.

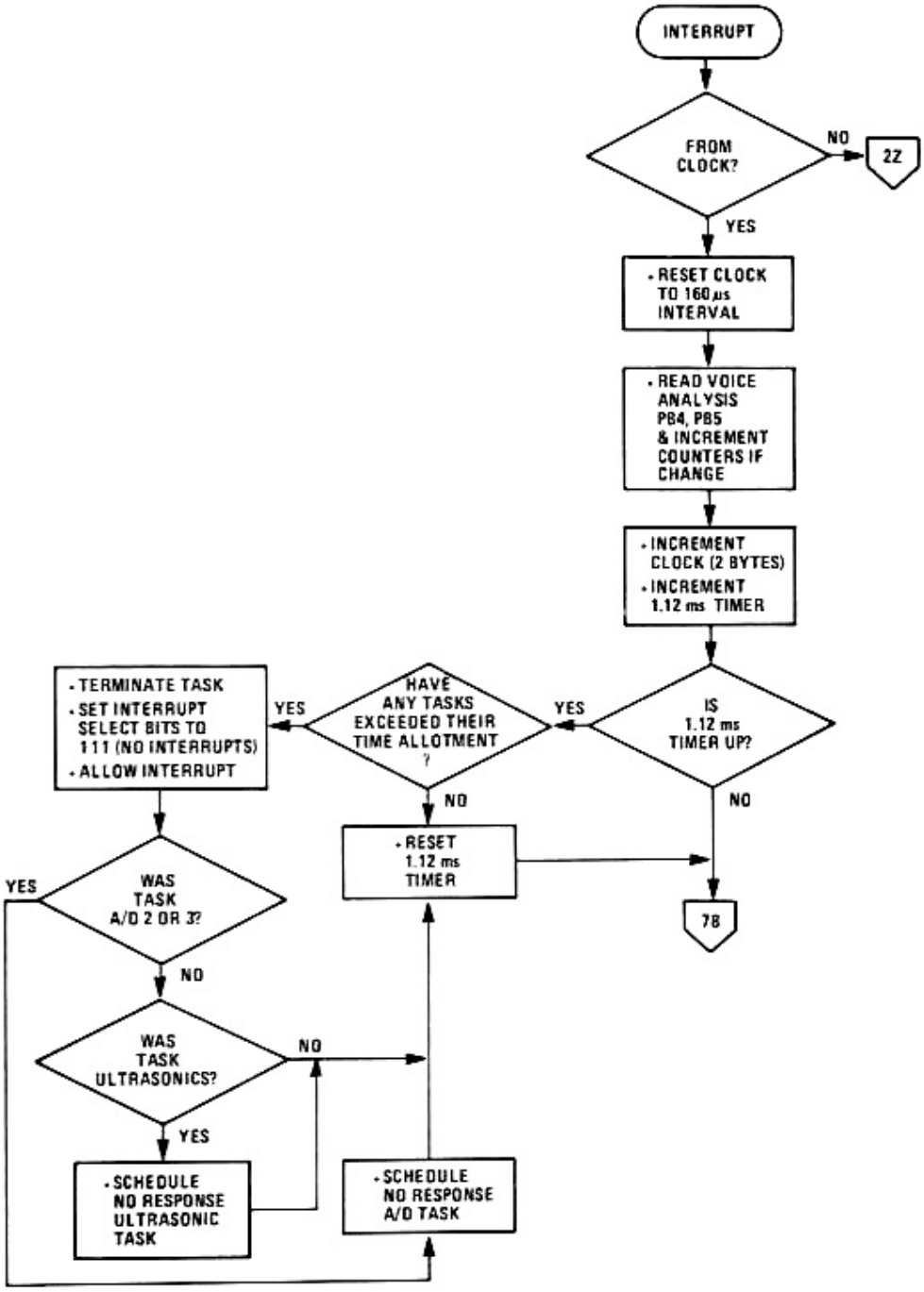

INTERRUPT
FROM CLOCK?
NO
22
YES
• RESET CLOCK TO 160 μs INTERVAL
• READ VOICE ANALYSIS PB4, PB5 & INCREMENT COUNTERS IF CHANGE
• INCREMENT CLOCK (2 BYTES)
• INCREMENT 1.12 ms TIMER
IS 1.12 ms TIMER UP?
YES
NO
HAVE ANY TASKS EXCEEDED THEIR TIME ALLOTMENT ?
YES
NO
• TERMINATE TASK
• SET INTERRUPT SELECT BITS TO 111 (NO INTERRUPTS)
• ALLOW INTERRUPT
• RESET 1.12 ms TIMER
78
WAS TASK A/D 2 OR 3?
YES
NO
WAS TASK ULTRASONICS?
NO
YES
• SCHEDULE NO RESPONSE ULTRASONIC TASK
• SCHEDULE NO RESPONSE A/D TASK

Chart 7–3.

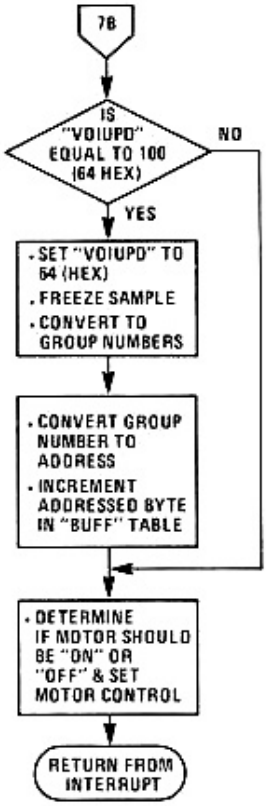


Chart 7–4.

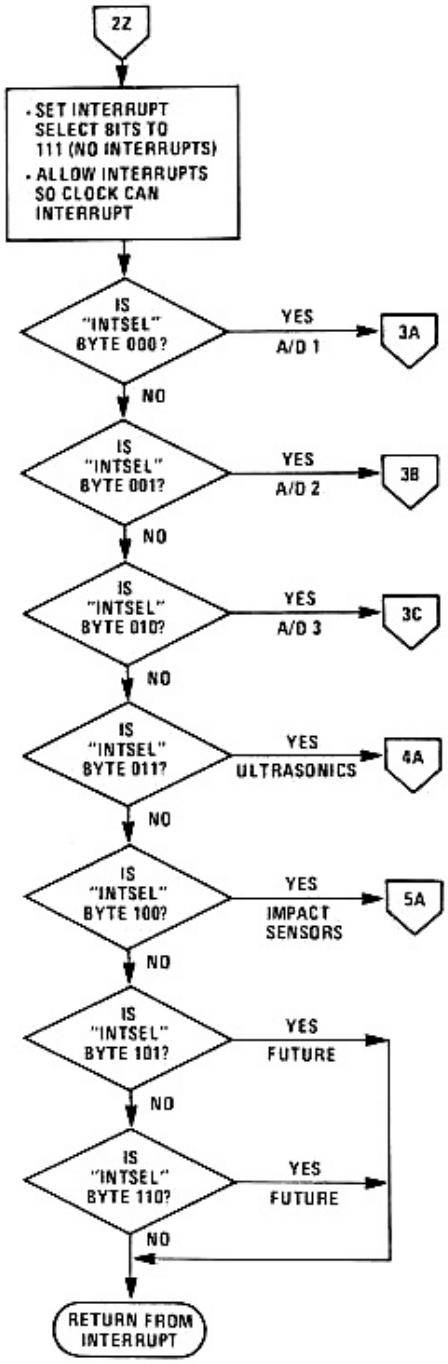


Chart 7–5.

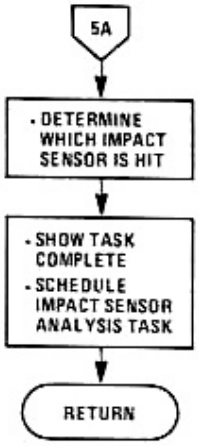


Chart 7–6.

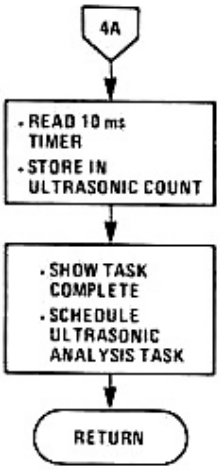


Chart 7–7.

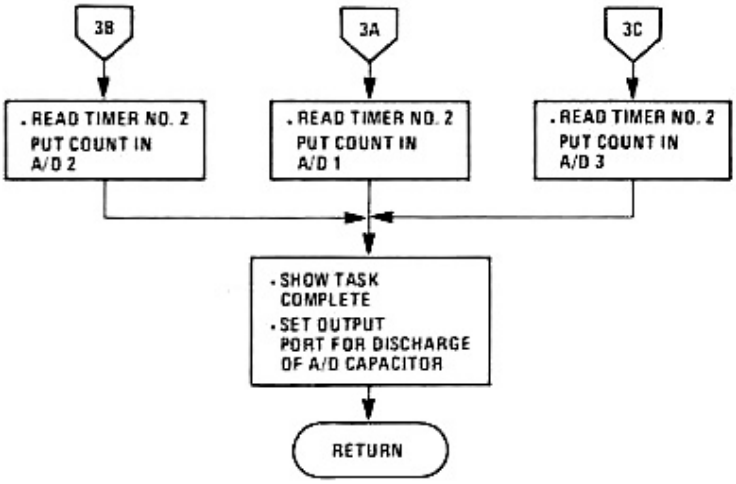


Chart 7–8.

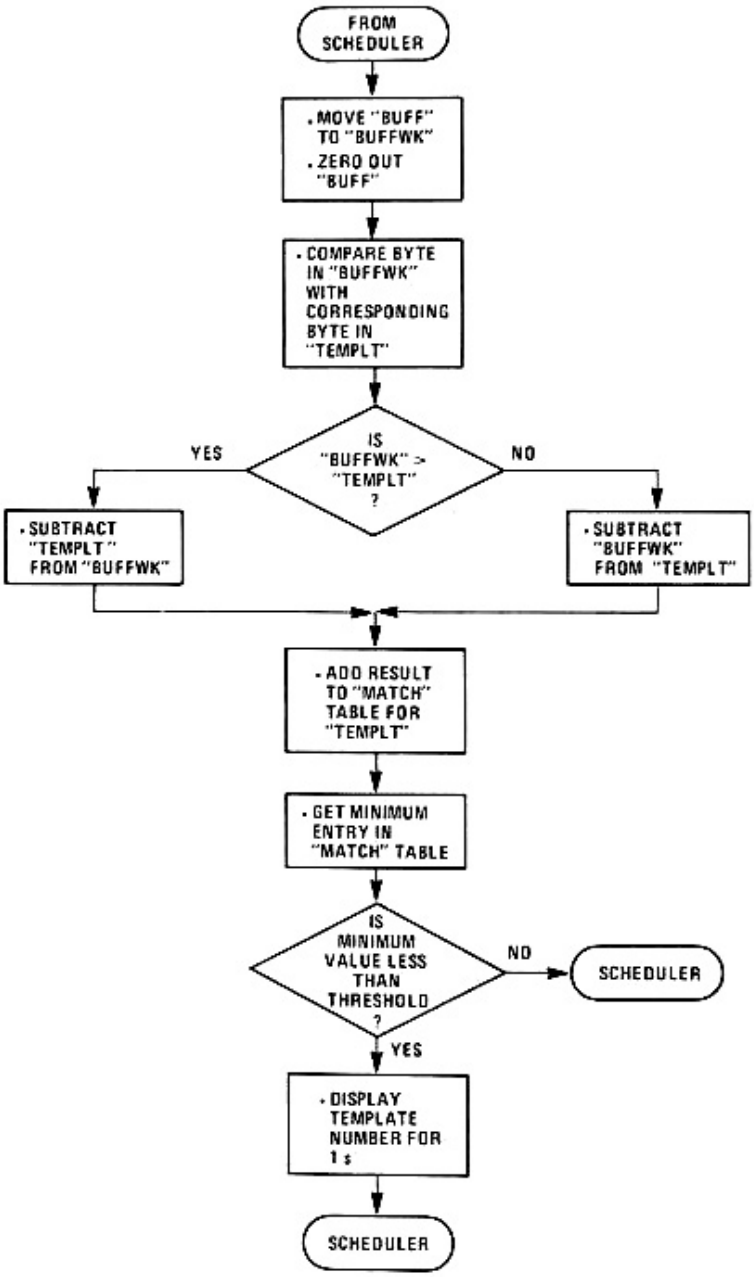

FROM SCHEDULER
. MOVE "BUFF" TO "BUFFWK"
. ZERO OUT "BUFF"
. COMPARE BYTE IN "BUFFWK" WITH CORRESPONDING BYTE IN "TEMPLT"
IS "BUFFWK" > "TEMPLT" ?
YES
NO
. SUBTRACT "TEMPLT" FROM "BUFFWK"
. SUBTRACT "BUFFWK" FROM "TEMPLT"
. ADD RESULT TO "MATCH" TABLE FOR "TEMPLT"
. GET MINIMUM ENTRY IN "MATCH" TABLE
IS MINIMUM VALUE LESS THAN THRESHOLD ?
NO
SCHEDULER
YES
. DISPLAY TEMPLATE NUMBER FOR 1 s
SCHEDULER

Chart 7–9.

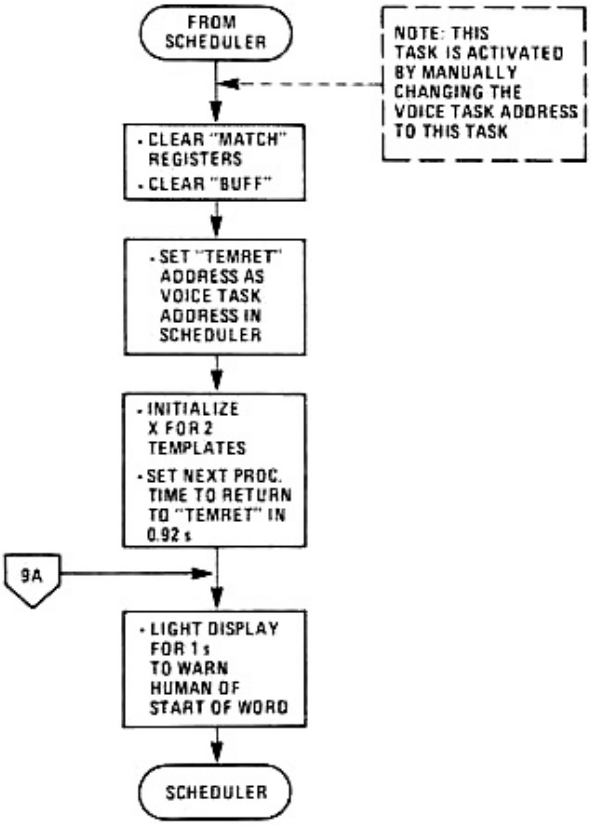

FROM SCHEDULER
NOTE: THIS TASK IS ACTIVATED BY MANUALLY CHANGING THE VOICE TASK ADDRESS TO THIS TASK
• CLEAR "MATCH" REGISTERS
• CLEAR "BUFF"
• SET "TEMRET" ADDRESS AS VOICE TASK ADDRESS IN SCHEDULER
• INITIALIZE X FOR 2 TEMPLATES
• SET NEXT PROC. TIME TO RETURN TO "TEMRET" IN 0.92 s
9A
• LIGHT DISPLAY FOR 1 s TO WARN HUMAN OF START OF WORD
SCHEDULER

Chart 7–10.

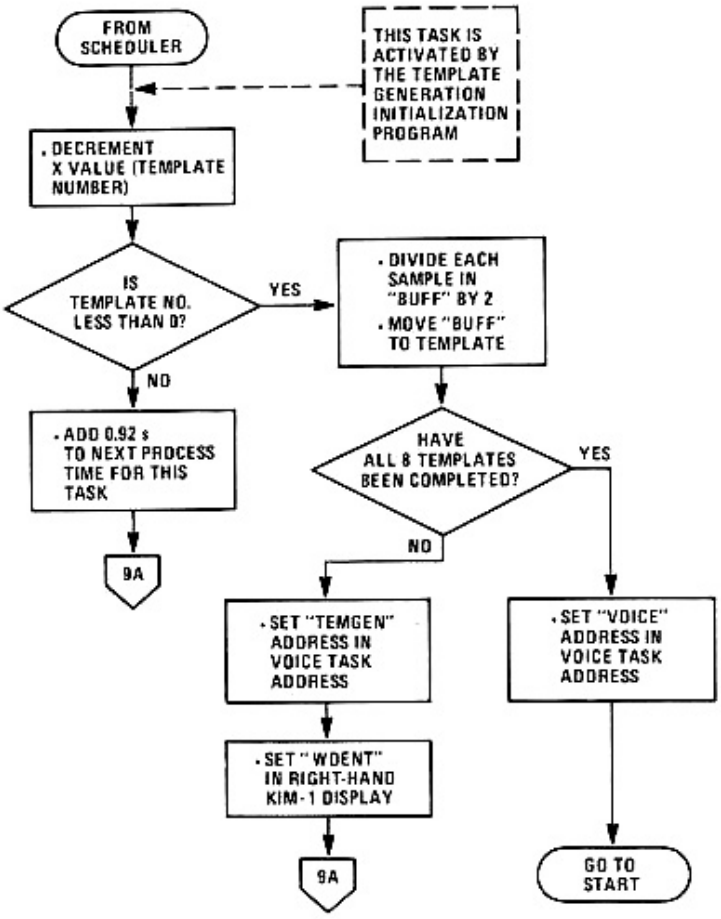


Chart 7–11.

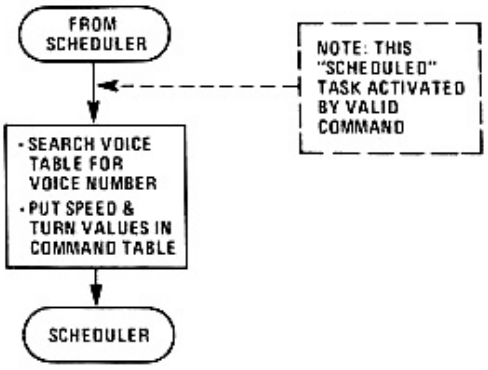


Chart 7–12.

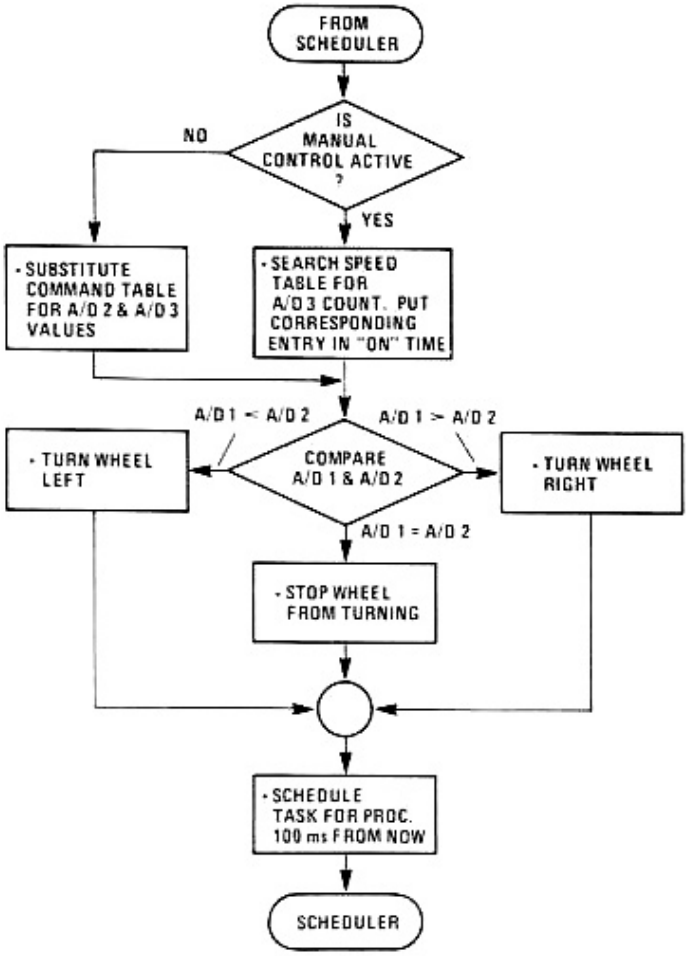


Chart 7–13.

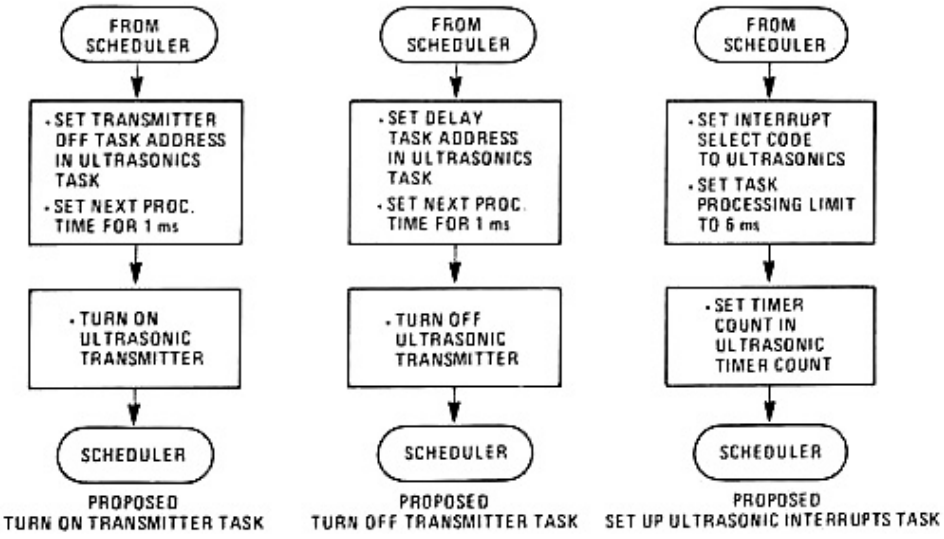


Chart 7–14.

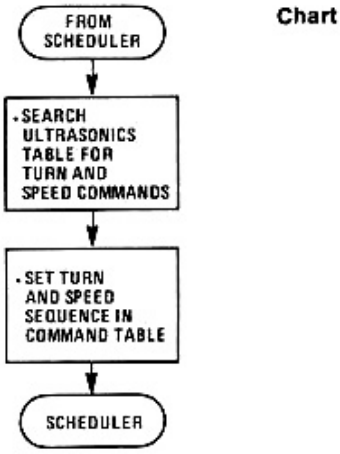


Chart 7–15.

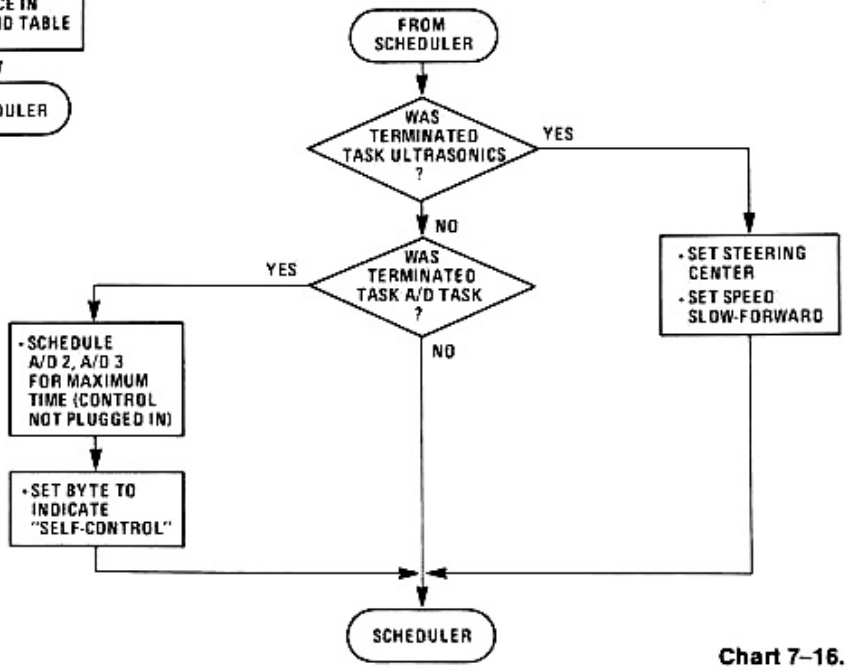


Chart 7–16.

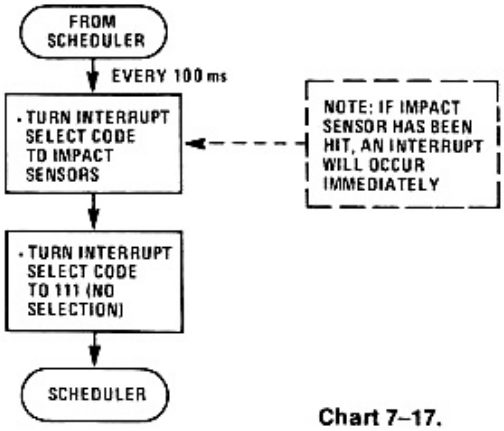


Chart 7–17.

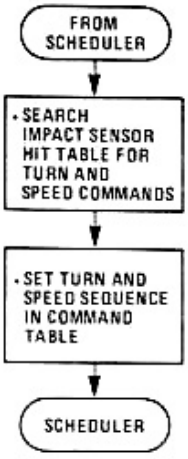


Chart 7–18.

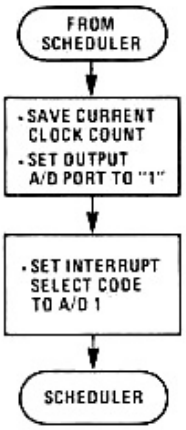


Chart 7–19.

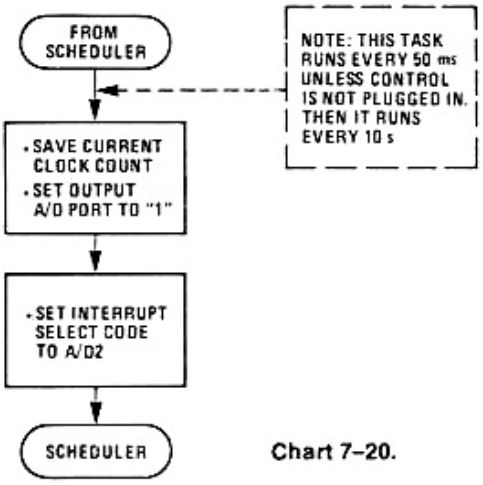


Chart 7–20.

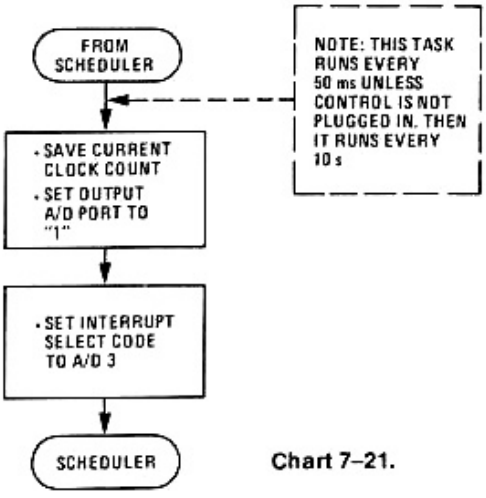


Chart 7–21.

When I was first experimenting with voice recognition, I started off by having Mike recognize whistles. By building each template with a whistle at a certain frequency, I enabled Mike to recognize eight different tones. It was quite a sight to see my entire family gathered around Mike, whistling at him. Whistles are much easier for Mike to recognize than are spoken words. If you are having trouble getting Mike to recognize words, try forming templates with whistles until you can perfect your ability to duplicate your voice inflection. Templates made from whistles are also useful in debugging software for voice recognition because they are more easily duplicated than are word templates.

Some time after experimenting with whistles, I began to try word recognition. At this time the software had not been perfected, and Mike was not recognizing words satisfactorily. For testing purposes, I had set up the software so that when Mike recognized a word, he would display the template number of the word that was recognized. I set up several word templates. The word "stop" was in template 3. I said the word "stop" several times, but Mike was unsuccessful in recognizing it. I finally gave up, and as I went to hit the "RESET" button I casually remarked, "I guess I'll have to stop the program and see what went wrong." Immediately upon hearing the word "stop," Mike lit the display to show a match with template 3. Since this experience, Mike's software has been changed, allowing him to recognize words more reliably.

```
VOICE RECOGNITION PROGRAM
WRITTEN BY JOHN W. LOOFBOURROW

                ; Reserved Storage for Interrupt
                ; Tasks
                ;
0000                    *=$0000
0000 01         INTTSK .BYTE $01,$38      task 1: next
0001 38                                   processing time
0002 3F                 .BYTE $3F         task 1: clock value
0003 20                 .BYTE $20,$50     task 1: start addr.
0004 50
0005 00                 .BYTE $00         task 1: status
                ;
0006 02                 .BYTE $02,$70     task 2
0007 70
0008 77                 .BYTE $77         task 2
0009 20                 .BYTE $20,$50     task 2
000A 50
000B 00                 .BYTE $00         task 2
                ;
000C 03                 .BYTE $03,$A8     task 3
000D A8
000E AF                 .BYTE $AF         task 3
000F 20                 .BYTE $20,$50     task 3
0010 50
0011 00                 .BYTE $00         task 3
                ;
0012 04                 .BYTE $04,$00     task 4
0013 00
0014 38                 .BYTE $38         task 4
0015 20                 .BYTE $20,$50     task 4
0016 50
0017 00                 .BYTE $00         task 4
                ;
0018 05                 .BYTE $05,$00     task 5
0019 00
001A 07                 .BYTE $07         task 5
001B 20                 .BYTE $20,$50     task 5
001C 50
001D 00                 .BYTE $00
                ;
                ; Reserved Storage for "Timed"
                ; Non-interrupt Tasks
                ;
0030                    *=$0030
0030 04         TASKST .BYTE $04,$10      next processing time
0031 10
0032 23                 .BYTE $23,$00     task address
0033 00
                ;
0034 04         FORREV .BYTE $04,$30      forward/reverse
```

```
0035 30
0036 20              .BYTE $20,$50
0037 50
                ;
0038 05         MOTCLK .BYTE $05,$00      motor control
0039 00
003A 20              .BYTE $20,$50
003B 50
                ;
003C 07         TRNSTK .BYTE $07,$00      steering
003D 00
003E 20              .BYTE $20,$50
003F 50
                ;
0040 FF         ULTCLK .BYTE $FF,$FF      ultrasonics
0041 FF
0042 20              .BYTE $20,$50
0043 50
                ;
                ; Reserved Storage for "Scheduled"
                ; Non-interrupt Tasks
                ;
0053                 *=$0053
0053 00         TASKSS .BYTE $00          status byte
0054 20              .BYTE $20,$50        task addr.
0055 50
                ;
0056 00              .BYTE $00            impact sensors
0057 20              .BYTE $20,$50
0058 50
                ;
0059 00              .BYTE $00            speed analysis
005A 20              .BYTE $20,$50
005B 50
                ;
005C 00         NORSPS .BYTE $00          no response task
005D 20              .BYTE $20,$50
005E 50
                ;
005F 00              .BYTE $00            voice commands
0060 20              .BYTE $20,$50
0061 50
                ;
                ; Data Definitions
                ;
006B                 *=$006B
006B            CLOCK  *=*+2              master clock
006D            MSTIME *=*+1              1.12 ms timer
006E            VCESAV *=*+1              prior voice sample
006F            VOICEH *=*+1              high pass count
0070            VOICEL *=*+1              low pass count
0071            INTSEL *=*+1              interrupt select
0072            NORSPN *=*+1              no response byte
```

```
0073            SAVXST  *=*+1           task pointer
0074            SAMPLH  *=*+1           high pass count
0075            SAMPLL  *=*+1           low pass count
0076            GROUPH  *=*+1           high pass group #
0077            GROUPL  *=*+1           low pass group #
0078            BUFF    *=*+30          buffer area
0096            BUFWK   *=*+30          work area
                ;
00BC                    *=$00BC
00BC            VCOMMD  *=*+1           voice command no.
00BD            TEMPNT  *=*+2           template pointer
00BF            JMPADD  *=*+2           used for indirect jumps
00C1            WDCNT   *=*+1           no. of word being gnrtd.
00C2            SMPLCT  *=*+1           current sample
00C3            NUMON   *=*+1           1st no.
00C4            NUMTW   *=*+1           2nd no.
00C5            MAX     *=*+2           maximum value
00C7            ANSON   *=*+1           1st byte ans.
00C8            ANSTW   *=*+1           2nd byte ans.
00C9            VOIUPD  *=*+1
00CA            MATCH   *=*+16          match table for 8 words
00EA                    *=$00EA
00EA            MINUS   *=*+1           minus indicator
00F9                    *=$00F9
00F9            MAXNO   *=*+1           no. of matched word
                ;
0200                    *=$0200
0200            TEMPLT  *=*+240         8 templates
                ;
                ; Beginning of Program - Initialization
                ;
2000                    *=$2000
2000 A9 FF      START   LDA #$FF        set PA for all outputs
2002 8D 0117            STA $1701
2005 A9 07              LDA #$07        PB0-2 = output
2007 8D 0317            STA $1703       PB3-7 = input
200A A9 00              LDA #INTADD     set interrupts low
200C 8D FE17            STA $17FE
200F A9 21              LDA #INTADD+1   set interrupts high
2011 8D FF17            STA $17FF
2014 A9 FE              LDA #$FE        initialize outputs
2016 8D 0017            STA $1700       in PA
2019 A9 81              LDA #$81        timer = 160 µs.
201B 8D 0C17            STA $170C
201E A9 07              LDA #$07        set 1.12 ms timer
2020 85 6D              STA MSTIME
2022 A9 64              LDA #$64        initialize voice update
2024 85 C9              STA VOIUPD      count
2026 D8                 CLD             set binary mode
2027 58                 CLI             remove interrupts
2028 A9 64              LDA #$64
202A A2 01              LDX #$01
202C 20 E022            JSR UPDATE
```

```
202F A5 6B          LDA CLOCK
2031 85 30          STA TASKST
2033 4C 5020        JMP SCHED
                    ;
                    ; Scheduler Program -
                    ; This program determines which
                    ; tasks are to be executed next.
                    ;
2050                *=$2050
2050 A2 00    SCHED LDX #$00          X = top of table
2052 B5 30    LOOP1 LDA TASKST,X      get MSB of nxt. proc. tm.
2054 C5 6B          CMP CLOCK
2056 30 3D          BMI EXCUT2        time overdue
2058 F0 49          BEQ CHKNXT        check LSB
205A E8             INX
205B E8       LOOP5 INX
205C E8             INX
205D E8             INX
205E E0 14          CPX #$14          check for end
2060 30 F0          BMI LOOP1
2062 A2 00          LDX #$00          X = 0
2064 B5 53    LOOP2 LDA TASKSS,X      get status
2066 30 47          BMI PROCES        branch if request pend.
2068 E8             INX
2069 E8             INX
206A E8             INX
206B E0 0F          CPX #$0F          check for end
206D 30 F5          BMI LOOP2
                    ;
                    ; Routine to Check Interrupt Table
                    ; for Any Active Interrupt Task
                    ;
206F A2 05          LDX #$05
2071 B5 00    LOOP3 LDA INTTSK,X      get status
2073 30 0B          BMI SCHED         branch if task active
2075 E8             INX
2076 E8             INX
2077 E8             INX
2078 E8             INX
2079 E8             INX
207A E0 28          CPX #$28          check for end
207C 30 F3          BMI LOOP3         branch if not
207E A2 00          LDX #$00          point to beginning
2080 B5 00    LOOP4 LDA INTTSK,X      get MSB of nxt. proc. tm.
2082 C5 6B          CMP CLOCK         compare to MSB clock
2084 30 3A          BMI GOTSK1        time overdue
2086 F0 1B          BEQ CKNXT         check LSB
2088 E8             INX
2089 E8       LOOP6 INX
208A E8             INX
208B E8             INX
208C E8             INX
208D E8             INX
```

```
208E E0 24          CPX #$24          check for end
2090 30 EE          BMI LOOP4
2092 4C 5020        JMP SCHED         no tasks to be run
2095 E8       EXCUT2 INX
2096 E8       EXCUT1 INX
2097 B5 30          LDA TASKST,X      get MSB address
2099 85 C0          STA JMPADD+1
209B E8             INX
209C B5 30          LDA TASKST,X      get LSB address
209E 85 BF          STA JMPADD
20A0 6C BF00        JMP (JMPADD)      jump ind. to task
20A3 E8       CHKNXT INX
20A4 B5 30          LDA TASKST,X      get LSB
20A6 C5 6C          CMP CLOCK+1       compare to clock
20A8 30 EC          BMI EXCUT1        time overdue
20AA F0 EA          BEQ EXCUT1        branch if time up
20AC 4C 5B20        JMP LOOP5         get next task
                    ;
20AF 29 7F    PROCES AND #$7F         set pend. bit to 0
20B1 95 53          STA TASKSS,X      reset status byte
20B3 E8             INX               point to task addr.
20B4 B5 53          LDA TASKSS,X      get MSB of addr.
20B6 85 C0          STA JMPADD+1
20B8 E8             INX
20B9 B5 53          LDA TASKSS,X      get LSB of address
20BB 85 BF          STA JMPADD
20BD 6C BF00        JMP (JMPADD)      jump ind. to task
                    ;
20C0 E8       GOTSK1 INX              increment to addr.
20C1 E8       GOTSK2 INX
20C2 E8             INX
20C3 B5 00          LDA INTTSK,X      get MSB of addr.
20C5 85 C0          STA JMPADD+1
20C7 E8             INX
20C8 B5 00          LDA INTTSK,X      get LSB of addr.
20CA 85 BF          STA JMPADD
20CC 6C BF00        JMP (JMPADD)      jump ind. to task
                    ;
20CF E8       CKNXT  INX              point to LSB
20D0 B5 00          LDA INTTSK,X      get LSB of proc. tm.
20D2 C5 6C          CMP CLOCK+1       compare to clock
20D4 30 EB          BMI GOTSK2        time overdue
20D6 F0 E9          BEQ GOTSK2        time up
20D8 4C 8920        JMP LOOP6         not ready yet
                    ;
20DB 4C B721 VOIUPB JMP VOIUP         from $2133
                    ;
                    ; Interrupt Processing Routine
                    ;
2100                *=$2100
2100 48       INTADD PHA              save A
2101 8A             TXA
2102 48             PHA               save X
```

```
2103 98              TYA
2104 48              PHA                 save Y
2105 2C 0217         BIT $1702           clock interrupt?
2108 30 70           BMI CHKSEL          branch if not
210A AD 0417         LDA $1704           reset interrupts
210D A9 81           LDA #$81            timer = 160 µs
210F 8D 0C17         STA $170C
2112 AD 0217         LDA $1702           read PB
2115 29 30           AND #$30            mask out all but PB4-5
2117 48              PHA                 save result
2118 45 6E           EOR VCESAV          compare to last sample
211A F0 08           BEQ NOCHG           branch if no change
211C 48              PHA                 save result
211D 29 10           AND #$10            mask out bit 5
211F D0 16           BNE BIT4            branch if change
2121 E6 6F           INC VOICEH
2123 68        FINISH PLA                pull back EOR result
2124 68        NOCHG  PLA                pull back PB4-5 value
2125 85 6E           STA VCESAV          save result
2127 E6 6C           INC CLOCK+1         incr. clock
2129 D0 02           BNE COMPL           branch if no carry
212B E6 6B           INC CLOCK
212D C6 6D     COMPL  DEC MSTIME         decrement 1.12 ms timer
212F F0 11           BEQ TIMEUP
2131 C6 C9     RESTOR DEC VOIUPD
2133 F0 A6           BEQ VOIUPB
2135 D0 79           BNE RETINT
                     ;
2137 E6 70     BIT4   INC VOICEL         add 1 to low pass
2139 68              PLA                 get back net change
213A 29 20           AND #$20            mask out bit 5
213C F0 E6           BEQ NOCHG           branch if no change
213E E6 6F           INC VOICEH          add 1 to high pass
2140 D0 E2           BNE NOCHG           (used instd. of jump)
2142 A9 07     TIMEUP LDA #$07           reset 1.12 ms timer
2144 85 6D           STA MSTIME
2146 A6 73           LDX SAVXST          point to status
2148 B5 00     LOOP7  LDA INTTSK,X       get status
214A 30 02           BMI ACTIVE          branch if task active
214C 10 E3           BPL RESTOR
214E CA        ACTIVE DEX                back to time-up byte
214F CA              DEX
2150 CA              DEX
2151 B5 00           LDA INTTSK,X        get time value
2153 C5 6C           CMP CLOCK+1         compare to LSB of clock
2155 30 02           BMI TIMPST          time-up - branch
2157 10 D8           BPL RESTOR          return
2159 E8        TIMPST INX                move to status
215A E8              INX
215B E8              INX
215C A9 7F           LDA #$7F            set mask
215E 35 00           AND INTTSK,X        turn off status bit
2160 95 00           STA INTTSK,X        task now inactive
```

```
2162 AD 0017        LDA $1700         read PA
2165 09 E0          ORA #$E0          turn PA5-7 on
2167 8D 0017        STA $1700
216A 58             CLI               restore interrupts
216B A5 71          LDA INTSEL        what task was terminated?
216D 85 72          STA NORSPN        save in NORSPN
216F A9 07          LDA #$07          set selector ind.
2171 85 71          STA INTSEL        for no interrupts
2173 A9 80          LDA #$80          set bit 7 = 1
2175 85 5C          STA NORSPS        ind. task on req.
2177 4C 3121        JMP RESTOR        task term. complete
                    ;
                    ; Interrupt Caused by External
                    ; Event
                    ;
217A AD 0017 CHKSEL LDA $1700         read PA
217D 09 E0          ORA #$E0          PA5-7 = 1
217F 8D 0017        STA $1700
2182 58             CLI               restore interrupts
2183 A5 71          LDA INTSEL        what task interrupted?
2185 F0 15          BEQ C000          branch if code 000 selected
2187 C9 02          CMP #$02
2189 30 14          BMI C001          branch if 001
218B F0 15          BEQ C002          branch if 002
218D C9 04          CMP #$04
218F 30 14          BMI C003          branch if 003
2191 F0 15          BEQ C004          branch if 004
2193 C9 05          CMP #$05
2195 F0 14          BEQ C005          branch if 005
2197 EA             NOP
2198 EA             NOP
2199 4C 3121        JMP RESTOR        branch if over 005
219C 4C 3121 C000   JMP RESTOR        A/D 1
219F 4C 3121 C001   JMP RESTOR        A/D 2
21A2 4C 3121 C002   JMP RESTOR        A/D 3
21A5 4C 3121 C003   JMP RESTOR        ultrasonics
21A8 4C 3121 C004   JMP RESTOR        impact sensors
21AB 4C 3121 C005   JMP RESTOR
                    ;
21B0                *=$21B0
21B0 68      RETINT PLA
21B1 A8             TAY
21B2 68             PLA
21B3 AA             TAX
21B4 68             PLA
21B5 40             RTI
21B6 EA             NOP
21B7 A9 64   VOIUP  LDA #$64
21B9 85 C9          STA VOIUPD
21BB 58             CLI
21BC 20 0022        JSR FREEZE        freeze sample
21BF 46 76          LSR GROUPH        move no. to
21C1 46 76          LSR GROUPH        lower nybble
```

```
21C3 46 76          LSR GROUPH
21C5 46 76          LSR GROUPH
21C7 C6 76          DEC GROUPH          (H-1)
21C9 A5 77          LDA GROUPL
21CB 18             CLC
21CC 06 76          ASL GROUPH          (H-1)*2
21CE 65 76          ADC GROUPH          (H-1)*2+L
21D0 06 76          ASL GROUPH          (H-1)*4
21D2 65 76          ADC GROUPH          (H-1)*2+(H-1)*4+L
21D4 AA             TAX
21D5 CA             DEX
21D6 F0 02          BEQ END
21D8 F6 78          INC BUFF,X
21DA 4C B021 END    JMP RETINT
                    ;
2200                *=$2200
2200 A5 6F    FREEZE LDA VOICEH         get high pass count
2202 85 74          STA SAMPLH          freeze count
2204 A5 70          LDA VOICEL          get low pass count
2206 85 75          STA SAMPLL          freeze count
2208 A9 00          LDA #$00            reset to 0
220A 85 6F          STA VOICEH          high count = 0
220C 85 70          STA VOICEL          low count = 0
220E 20 2022        JSR CONVRT
2211 60             RTS
                    ;
                    ; Subroutine to Convert Sample Count
                    ; to Group Numbers
                    ;
2220 A5 74    CONVRT LDA SAMPLH         get high pass count
2222 C9 10          CMP #$10            count = 10?
2224 F0 05          BEQ LOOP7
2226 30 22          BMI ADD10
2228 38             SEC                 set carry (i.e., no borrow)
2229 E9 01          SBC #$01
222B 29 F0    LOOP7 AND #$F0            mask through top 4 bits
222D 20 6922        JSR LIMIT
2230 EA             NOP
2231 C9 07          CMP #$07            check if 0-6
2233 30 1B          BMI GRP1
2235 C9 0E          CMP #$0E            check if 7-13
2237 30 1C          BMI GRP2
2239 C9 14          CMP #$14            check if 14-19
223B 30 1D          BMI GRP3
223D C9 1A          CMP #$1A            check if 20-25
223F 30 1E          BMI GRP4
2241 C9 20          CMP #$20            check if 26-32
2243 30 1F          BMI GRP5
2245 A9 06          LDA #$06            over 32
2247 85 77          STA GROUPL
2249 60             RTS
224A 18       ADD10 CLC
224B 69 10          ADC #$10
```

```
224D 4C 2822          JMP LOOP7
2250 A9 01    GRP1    LDA #$01
2252 85 77            STA GROUPL
2254 60               RTS
2255 A9 02    GRP2    LDA #$02
2257 85 77            STA GROUPL
2259 60               RTS
225A A9 03    GRP3    LDA #$03
225C 85 77            STA GROUPL
225E 60               RTS
225F A9 04    GRP4    LDA #$04
2261 85 77            STA GROUPL
2263 60               RTS
2264 A9 05    GRP5    LDA #$05
2266 85 77            STA GROUPL
2268 60               RTS
                ;
                ; Limits Maximum Value
                ;
2269 C9 60    LIMIT   CMP #$60
226B 10 05            BPL MK50
226D 85 76    LIMCON  STA GROUPH
226F A5 75            LDA SAMPLL
2271 60               RTS
2272 A9 50    MK50    LDA #$50
2274 4C 6D22          JMP LIMCON
                ;
                ; Voice Task
                ;
2300                  *=$2300
2300 20 8025  VOICE   JSR CLRCLK
2303 A2 1D            LDX #$1D          point to buffer end
2305 B5 78    BUFLP   LDA BUFF,X        move buffer to
2307 95 96            STA BUFFWK,X      work area
2309 A9 00            LDA #$00
230B 95 78            STA BUFF,X        zero out BUFF
230D CA               DEX
230E 10 F5            BPL BUFLP
2310 A2 0F            LDX #$0F          2*(#words-1)
2312 A9 00            LDA #TEMPLT+1
2314 85 BD            STA TEMPNT
2316 A9 02            LDA #TEMPLT
2318 85 BE            STA TEMPNT+1
231A A0 1D    VOILPB  LDY #$1D
231C 4C 7723  VOILPA  JMP VOIPTB
231F EA               NOP
2320 B1 BD    VOIPTA  LDA (TEMPNT),Y
2322 85 C4            STA NUMTW
2324 20 8024          JSR ADDTOT        add to total
2327 88               DEY
2328 10 F2            BPL VOILPA
232A 20 7024          JSR ADDPNT
232D CA               DEX
```

```
232E CA             DEX
232F 10 E9          BPL VOILPB
2331 30 4D          BMI CLR
2333 B5 C9   MCHLPA LDA MATCH-1,X     get MSB
2335 C5 C5          CMP MAX
2337 F0 16          BEQ CKLSB
2339 90 27          BCC NUMAX
233B CA      MCHLPC DEX
233C CA             DEX
233D 10 F4          BPL MCHLPA
233F A5 96          LDA BUFFWK
2341 C9 2B          CMP #$2B          threshold MSB
2343 90 22          BCC THCKLS
2345 B0 26          BCS NOMET
2347 46 F9   MCHLPE LSR MAXNO         divide by 2
2349 20 7026 MCHLPB JSR SETSCN        loop with match
234C 4C 5020        JMP SCHED
234F B5 CA   CKLSB  LDA MATCH,X
2351 C5 C6          CMP MAX+1
2353 F0 E6          BEQ MCHLPC
2355 B0 E4          BCS MCHLPC
2357 85 C6   MCHLPD STA MAX+1
2359 B5 C9          LDA MATCH-1,X
235B 85 C5          STA MAX
235D 86 F9          STX MAXNO
235F 4C 3B23        JMP MCHLPC
2362 B5 CA   NUMAX  LDA MATCH,X
2364 4C 5723        JMP MCHLPD
2367 A5 C6   THCKLS LDA MAX+1
2369 C9 17          CMP #$17          threshold LSB
236B 90 DA          BCC MCHLPE
236D 4C 5020 NOMET  JMP SCHED
2377                *=$2377
2377 B9 9600 VOIPTB LDA BUFFWK,X
237A 85 C3          STA NUMON
237C 4C 2023        JMP VOIPTA
             ;
             ; TO ACTIVATE; Change $0033 to $90
             ; & $00C1 to $07.
             ;
             ; Template Generation Program
             ;
2390                *=$2390
2390 78      TEMGEN SEI               stop interrupts
2391 20 7325        JSR CLRMAT        clear match reg.
2394 20 6024        JSR ADDCLK        update nxt. proc. tm.
2397 20 0022        JSR FREEZE        clear out buffers
239A A9 23          LDA #TEMRET       set up template
239C 85 32          STA TASKST+2      generation addr.
239E A9 BA          LDA #TEMRET+1
23A0 85 33          STA TASKST+3
23A2 A9 01          LDA #$01          set up for 2 templates
23A4 48             PHA
```

```
23A5 A2 1D          LDX #$1D          point to buffer end
23A7 A9 00          LDA #$00
23A9 95 78   TEMLPA STA BUFF,X        clear buffer
23AB CA             DEX
23AC 10 FB          BPL TEMLPA
23AE A2 80   TEMLPC LDX #$80
23B0 20 C025 TEMLPB JSR LNGSCN        warn user of start
23B3 20 1F1F        JSR SCANDS
23B6 58             CLI
23B7 4C 5020        JMP SCHED
23BA 78      TEMRET SEI
23BB 68             PLA
23BC AA             TAX
23BD CA             DEX
23BE 30 10          BMI TEMFIN
23C0 8A             TXA
23C1 48             PHA
23C2 20 6024        JSR ADDCLK
23C5 4C AE23        JMP TEMLPC
                ;
23D0                *=$23D0
23D0 A2 1D   TEMFIN LDX #$1D
23D2 86 C2          STX SMPLCT
23D4 B5 78   TEMLPF LDA BUFF,X
23D6 4A             LSR
23D7 4C DC23        JMP STODIR
23DC                *=$23DC
23DC 95 78   STODIR STA BUFF,X
23DE 4C F623        JMP JMPARN
23F6                *=$23F6
23F6 C6 C2   JMPARN DEC SMPLCT
23F8 A6 C2          LDX SMPLCT        test for end buffer
23FA 10 D8          BPL TEMLPF
23FC EA             NOP
23FD EA             NOP
23FE EA             NOP
23FF EA             NOP
2400 4C 1624        JMP JMPAST
2416                *=$2416
2416 A9 00   JMPAST LDA #TEMPLT+1
2418 85 BD          STA TEMPNT
241A A9 02          LDA #TEMPLT
241C 85 BE          STA TEMPNT+1
241E A6 C1          LDX WDCNT
2420 F0 06   TEMPLH BEQ TEMBRA
2422 20 7024        JSR ADDPNT
2425 CA             DEX
2426 10 F8          BPL TEMLPH
2428 A0 1D   TEMBRA LDY #$1D          point to buffer end
242A B9 7800 TEMLPI LDA BUFF,Y
242D 91 BD          STA (TEMPNT),Y    move buffer to
242F 88             DEY               template
2430 10 F8          BPL TEMLPI
```

```
2432 C6 C1          DEC WDCNT
2434 30 12          BMI TEMDON
2436 A9 90          LDA #TEMGEN+1
2438 85 33          STA TASKST+3      point to generation
243A A9 23          LDA #TEMGEN       address
243C 85 32          STA TASKST+2
243E A5 C1          LDA WDCNT         set up word #
2440 85 F9          STA MAXNO
2442 20 1F1F TEMLPJ JSR SCANDS        display next word #
2445 4C 4224        JMP TEMLPJ
                    ;
                    ; Subroutine to Add 6000 to Next
                    ; Process Time
                    ;
2460                *=$2460
2460 A2 01   ADDCLK LDX #$01
2462 A9 70          LDA #$70          add 70 to update
2464 20 E022        JSR UPDATE
2467 B5 2F          LDA TASKST-1,X
2469 18             CLC
246A 69 17          ADC #$17          add 17 to MSB
246C 95 2F          STA TASKST-1,X
246E 60             RTS
                    ;
                    ; Subroutine to Add 30 to Template
                    ; Pointer
                    ;
2470                *=$2470
2470 A9 1E   ADDPNT LDA #$1E
2472 18             CLC
2473 65 BD          ADC TEMPNT
2475 85 BD          STA TEMPNT
2477 90 02          BCC ADDDON
2479 E6 BE          INC TEMPNT+1
247B 60      ADDDON RTS
                    ;
2448                *=$2448
2448 A9 00   TEMDON LDA #VOICE+1      set up voice
244A 85 33          STA TASKST+3      analysis task
244C A9 23          LDA #VOICE
244E 85 32          STA TASKST+2
2450 4C 0020        JMP START         restart
                    ;
2380                *=$2380
2380 A2 0F   CLR    LDX #$0F
2382 A9 FF          LDA #$FF
2384 85 C5          STA MAX
2386 85 C6          STA MAX+1
2388 4C 3323        JMP MCHLPA
                    ;
                    ; Subroutine to Update Next Process
                    ; Time
                    ;
```

```
                ; INPUT ROUTINE: Acc. = amount to add
                ;                    X = pointing to
                ;                        LSB of next
                ;                        processing time
                ;                        bytes.
                ;
22E0                    *=$22E0
22E0 18         UPDATE  CLC
22E1 65 6C              ADC CLOCK+1
22E3 95 30              STA TASKST,X
22E5 90 07              BCC RETURN
22E7 A5 6B              LDA CLOCK
22E9 69 00              ADC #$00
22EB EA                 NOP
22EC 95 2F              STA TASKST-1,X
22EE 60         RETURN  RTS
                ;
2580                    *=$2580
2580 20 6024    CLRCLK  JSR ADDCLK
2583 20 7325            JSR CLRMAT
2586 60                 RTS
                ;
2480                    *=$2480
2480 20 4026    ADDTOT  JSR ABSUB
2483 A5 C8              LDA ANSTW
2485 18                 CLC
2486 75 CA              ADC MATCH,X
2488 95 CA              STA MATCH,X
248A B5 C9              LDA MATCH-1,X
248C 65 C7              ADC ANSON
248E 95 C9              STA MATCH-1,X
2490 60                 RTS
                ;
                ; Subtract Routine - Absolute Value
                ;
2640                    *=$2640
2640 A5 C3      ABSUB   LDA NUMON
2642 C5 C4              CMP NUMTW
2644 90 09              BCC REVERS
2646 E5 C4              SBC NUMTW
2648 85 C8      SUBCON  STA ANSTW
264A A9 00              LDA #$00
264C 85 C7              STA ANSON
264E 60                 RTS
264F A5 C4      REVERS  LDA NUMTW
2651 38                 SEC
2652 E5 C3              SBC NUMON
2654 4C 4826            JMP SUBCON
                ;
2670                    *=$2670
2670 78         SETSON  SEI
2671 A2 80              LDX #$80
2673 20 C025            JSR LNGSCN
```

```
2676 58          CLI
2677 60          RTS
                 END
```

chapter eight mike's future

Mike's construction will never be completed. Ideas for new systems keep coming faster than I can implement them. Even now, advanced as he is, Mike is far from finished. A number of senses, systems, and functions can still be added.

I have included a description of some of the features I plan to add to Mike. I must admit that I have let my imagination run wild in a few places, but in time I hope to accomplish everything set forth in this chapter. By then, I will probably have still more ideas. The following should be used merely as a guide, to suggest possible directions you can take with your Mike's construction.

The next step I plan in constructing Mike is building him a "body." Although Mike is 27 in. wide, he is only 14 in. high. Therefore, he does not look complete but rather looks like the base of a robot. I plan to add to Mike's frame to increase his height to 4 or 5 ft. Besides providing aesthetic appeal, a body provides room for the features that are planned in this chapter.

I have designed two frames, either of which I may use as Mike's body. The first design is called Plan A and the second design Plan B. Plan A details a one-piece unit (Fig. 8–1). The unit is separate from Mike's base (the triangular and eight-sided frames) so that he can fit into a car when he is being transported. The one-unit body is between 3

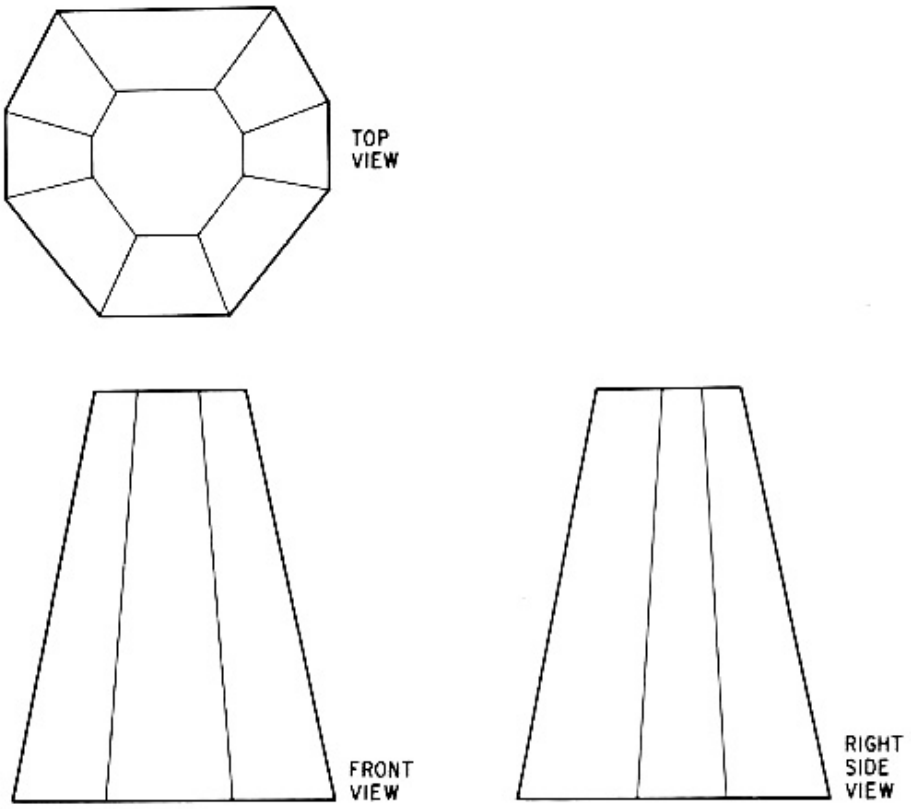


Fig. 8–1. Plan A: One-piece body.

ft and 4 ft in height and plugs directly into the base. The top of this unit has sides that are exactly half as long as the sides of its bottom. Its bottom is exactly the same size as the impact sensor frame. The top and sides of the unit are cut from sheet aluminum.

The second body design, Plan B, utilizes the modular concept (Fig. 8–2). The lowest module is Mike's present frame, which contains the major circuitry. The other modules would each house several of Mike's new functions. Every time you build Mike a new system, merely add another module to his frame.

After completing his body, I plan to give Mike a voice. A voice would provide Mike with a distinct personality. I am planning to use true computer-generated speech, not a tape recording. Computer-generated speech could be produced in either of two ways. One method is generating speech directly from phonemes. This method takes up very little memory in the computer, but it produces speech that is hardly intelligible. The second method is called pulse code modulation. This method digitizes speech at approximately 10,000 to 20,000 samples per

second. The resulting speech is extremely high in quality, but unfortunately it requires 10 K to 20 K bytes of computer memory per second.

There are several speech synthesis systems that are a compromise between these two methods. Of these systems, the highest quality speech output is, to the best of my knowledge, produced by Computalker. The Computalker is designed for S–100 bus connectors, but I hope to interface it to the Kim. Every second of speech produced by Com-

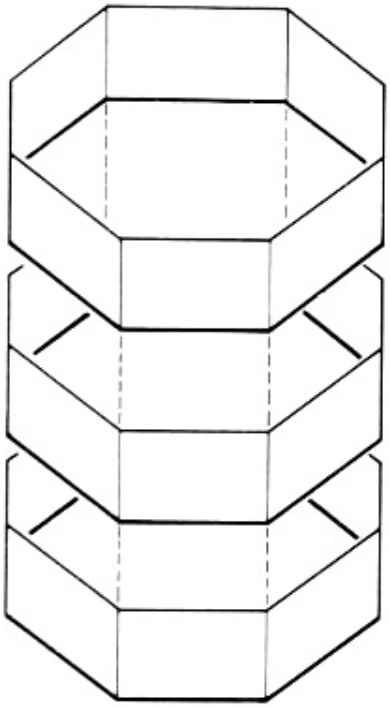


Fig. 8–2. Plan B: Modular construction.

putalker requires 500 to 900 bytes of memory. Computalker's speech can be produced in either of two ways. Speech can be synthesized by subjecting a human speech pattern to computer analysis, thereby generating the required Computalker inputs. Speech can also be produced by software, which generates Computalker data from standard speech phonemes. These phonemes can be linked in many combinations to form words and sentences. This method requires a minimum of memory. Unfortunately, the quality of the speech produced by the phoneme method is poor. All things considered, I will probably use Computalker's human speech analysis method of speech generation.

Two other systems that I am going to add to Mike are a video terminal and a keyboard. The terminal and keyboard will enable me to

load programs into Mike's computer-brain in assembly language. The computer can convert the assembly language into machine code by using an assembler program. The terminal and keyboard will enable me to run programs in BASIC or FOCAL on the Kim–1. With the keyboard and the terminal it will be possible to develop a special control language specifically suited to Mike's operation. Commands could relate directly to Mike's functions. Therefore, responses to various sensory stimuli would be easily programmed. For example, a command such as "F 1 5 L 1 5" would tell Mike to move forward at slow speed and turn left at the slightest turning angle for 5 s. The command "V 1 7 8 2" would tell Mike to recite a sentence using words 1, 7, 8, and 2. A control language, as made possible by the terminal and keyboard, would greatly simplify response programming for Mike. A function that the video terminal will perform later is that of enabling Mike to display a picture of his environment as he "sees" it.

Another feature that Mike is going to have is one or two arms. I will probably mount the arm(s) on top of Mike, or else on his chest. The advantage of its (their) being on top is that the arm(s) could reach in all directions, without Mike's having to turn.

Depending on its complexity, an arm can contain between two and 20 reversible motors. The arm that I have designed contains seven (Fig. 8–3). The shoulder joint uses a gimbal (Fig. 8–4). A gimbal contains one joint within another. If one joint of the shoulder is to be moved, motor 1 is activated. The position of the joint is recorded by pot 1. If the

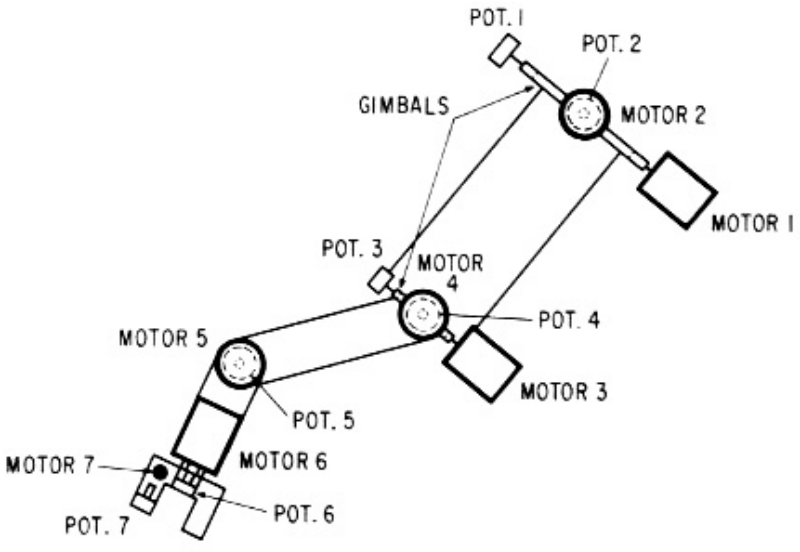


Fig. 8–3. Projected arm design.

shoulder's other joint is to be moved, motor 2 is activated. The second joint's position is recorded by pot 2. By turning each joint at different degrees, the arm can be moved to almost any position that your shoulder can move your arm. The elbow joint uses a second gimbal. The gimbal's inner joint is moved by a reversible motor (motor 3). The motor is attached to the upper arm and moves the lower arm left and right. A potentiometer is also attached to the upper arm. It informs the micro of the position of the inner joint. A second motor (motor 4) moves the outer joint and causes the lower arm to move up and down. Using a gimbal at the elbow joint allows Mike's arm to have freedom of movement even

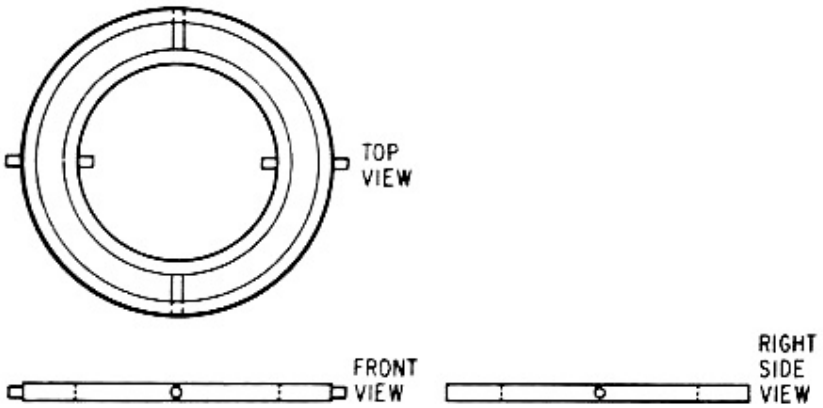


Fig. 8–4. A gimbal (as used in Mike's arm).

greater than that of a human arm. The wrist joint contains two motors. The first (motor 5) allows the wrist to move up and down. The vertical position of the wrist joint is monitored by a pot. The second motor (motor 6) allows the claw to rotate. The claw's rotational position is monitored by a nine-turn pot. The last motor (motor 7) is smaller than the others. The motor is used to open and close the claw. I have not yet determined precisely how the motor will operate the hand. However, I do know that the claw's position will be monitored by a pot. The claw may contain a force sensor to detect the amount of pressure necessary to pick up an individual object. The force sensors are presently in the design state and have not yet been perfected.

Another system that I plan to add to Mike is an image sensor camera, a device that will refine his "sight" capability. An image sensor camera can actually produce a black and white picture of Mike's environment. The camera must be able to move freely up and down Mike's body so that he can see objects at all levels. The camera uses 1024 image sensors, which are simply photo diodes. The photo diodes are all charged to a predetermined voltage and are then allowed to discharge.

Each image sensor discharges at a rate proportional to the amount of light that hits it. A picture can be produced in 16 shades of gray, so that 512 bytes can store the entire picture.

An interesting application of image sensor "sight" is object recognition. Once Mike can "see" his environment, he can be taught to pick out and examine certain objects that interest him. He will hold each object directly in front of his image-sensing camera. He will store a picture of each object on cassette tape. Some time later he will show you a picture of each object and ask you to type in the name of the object. Next, you will say the object's name. Mike will analyze the name you say with his voice analysis circuitry. He will store the spoken name, along with the typed name, immediately before the picture of the object, on the cassette tape. As he "learns" more and more objects, Mike will begin to recognize those that he has seen before.

By using a control language, Mike can be made to seek out and find certain objects. "FIND [control word] BALL [object]" causes Mike to search for and, it is hoped, find a ball. After finding the object, he can respond, using his voice. In addition to direct commands, Mike could learn to recognize various words and phrases used as "catch words." When Mike was being addressed, he would recognize these words within the conversation. With a sufficient amount of memory, Mike could learn enough words so that he could actually carry on a fluent conversation.

What comes next? I can't truthfully say that I know. Mike can be as complex as you are imaginative.

Some people say that I am a dreamer. But that's what they said when I started.

appendix

Motorized Wheels—Cat.# TM21K460
and
Directional Control Motor (6–12 VDC Motor Geared Reducer)—Cat.# TM20K407
and
Controlflex Ribbon Switches—Cat.# T5–012
Available from:
Herbach & Rademan, Inc.
401 East Erie Ave.
Philadelphia, PA 19134

Joystick with 5 kΩ Pots
Available from:
James Electronics
1021–A Howard Ave.
San Carlos, CA 94070

Kim–1 Microcomputer
Available from:
MOS Technology
950 Rittenhouse Rd.
Norristown, PA 19401

LM 1812 Ultrasonic Transceiver
Available from:
Tri–Tek Inc
7808 N. 27th Avenue
Phoenix, AR 85021

23-kHz Ultrasonic Transducer—Cat.# SP–110
Available from:
Meshna
19 Allerton St.
East Lynn, MA 01904

Econoram II®
Available from:
Bill Godbout Electronics
Box 2355
Oakland Airport, CA 94614

index

A/D circuits, 2, 30ff
  parts list for, 29
Arms, 126f

Battery cage, 14ff
  anchorage of, 16
Battery charger, 22
Body design of robot, 123

Cable, four-conductor, 34
Camera, image sensor, 5, 127f
Checklist of completed stages, 7
Claw, 127
Computalker, 125
Control language, special, 126, 128
Cyborg, 3

Directional control
  assembly, 16ff
  circuit, 2, 20, 26ff
  gear ratio, 16
  motor, 15–18, 26
  offset, 68
  pot, 32, 34f, 38, 48, 50
  routine, 38
Disable switch, 26

"Ears," 5
Elbow joints, 127
Enconoram II, 88ff

Feelers, 4, 65
Filters, 87
Force sensor, 127
Forward-reverse control, 23
Framework, basic, 8ff

Gear ratio for directional control, 16

Heatsinks, 22, 23, 25, 26

Image sensor camera, 5, 127f
Impact sensors, 4, 56ff, 85
  circuitry, 65
  control routine, 68
  detector, 65
  outer frame for, 57f
  parts list for, 56
  selector, 66
Infrared detection, 76
Interrupt capability, 88
  mask, 70
  selection circuit, 94
  software, 88
Interval timer, 95
Inverter circuit, 2, 30f, 32
  parts list for, 30

Joystick, 2, 3, 30, 32, 34, 48, 50
  control program, 52, 85

Keyboard, 6, 125f
Keypad, 36, 42
Kim-1 microcomputer, 2, 32, 36, 88

Logic circuitry, 21

Memory board, 88
Micro, *see* Microprocessor
Microcomputer, *see* Microprocessor
Microphone, 5, 87, 88
Microprocessor, 2, 21, 23, 30, 36f
Mobility of the robot, 2

Object recognition, 128
"On-off" switch, 22

Phonemes, 124f
Power supply, 2, 14, 19
  circuit, 2, 21ff
Programming, 36
  see also Scheduler program, Self-direction program, Software
Proximity detection, 76
Pulse code modulation, 124

Response programming, 126
Ribbon switches, 57, 62ff

S-100 compatible board, 88
"S Byte," 68
"S Count," 52
Scheduled tasks, 94, 97
Scheduler program, master, 94
Self-direction program, 49, 52
Self-speed routine, 50
Self-turning routine, 49
Sensory systems, advanced, 5
  see also Feelers, Impact sensors, "Ultrasonics"
Shoulder joints, 126f
Software, 30, 37, 39, 42, 68, 79f, 88, 108
Speech, computer-generated, 124
Speed command pot, 32, 39
Speed control
  circuitry, 2, 20, 23ff
  routine, 39
Speed offset, 70
Stages of the robot's development, 1ff

"T Count," 50
Templates, *see* Word templates
Timed tasks, 94, 97
Timer, 30
  see also Interval timer

ULCNT, 80
Ultrasonic detection, 76
  circuitry, 76ff
  tuning, 83
Ultrasonic "sight," 3
Ultrasonic sound, 75ff
Ultrasonic transducer, 3, 78

Video terminal, 6, 125f
Voice commands, 5
Voice recognition, 84ff, 94ff
  circuitry, 85ff, 130
  parts list for, 84
  program, 97

Wheels, 9, 23ff, 26, 34, 80
  steerable, 11
Whistling, 5, 108
Word recognition, 108
Word templates, 5, 96f, 108
Wrist joints, 127

Zero crossing analysis, 85

1-8
75

HOW TO BUILD A COMPUTER-CONTROLLED ROBOT

Tod Loofbourrow

The dream of robotics, to create an intelligence other than human, is fast becoming a reality with the availability of microprocessors. This book combines the dream and the reality, by providing both hands-on experience with robotics and an application of a microprocessor. You can learn the fundamentals of robotics while utilizing the ultimate in current hobby computer technology.

This book details the step-by-step directions for building a computer-controlled robot, named "Mike," controlled by a KIM-1 microprocessor. Every step of the construction of "Mike" is explained, with the complete control programs clearly written out. Photographs, diagrams, and tables help to direct you in the construction. You may use the directions exactly as they are set forth in the book or as a basis for developing your own design.

So, get ready to experience the thrill of creating an intelligence other than human. Open the book and begin seeing your dream come true.

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

ISBN 0-8104-5681-8