BUILD A MICROCOMPUTER-CONTROLLED ROBOT

AND OTHER COMPUTER CONTROL PROJECTS

BY B. C. TAYLOR

Budget Robotics & Computing
PO Box 430
Lakeside, AZ 85929

First Edition
Second Printing

Printed in the United States of America

Taylor, B.C.
Build a microcomputer-controlled robot and other computer control projects.

CONTENTS

THIS PAGE INTENTIONALLY BLANK

Introduction

This book will show the robot tinkerer, the high school science fair project builder and anyone else interested in understanding computer control, how to build and demonstrate this control, simply and inexpensively. How inexpensively? For $30 to $50 you can have a very capable computer. For an equal amount you can have eight lines of input and output control. This compares to commercially available single board control computers which cost hundreds of dollars.

The book lays out simple application projects with detailed examples and explanations. These projects range from simple computer input and output to the electrical and electronics circuits of a self contained mobile robot. Although it is a practical "how to build it" book, it also demonstrates the concepts of computer control possibilities which enables the reader to "learn by doing". All components for the projects are readily available, including some commercially available bare board printed circuits. All that is required by the reader is to put them together. The electronics assembly can be accomplished in the home on the kitchen table.

The text explains circuits which are illustrated with schematics, photos and drawings. The basic thrust of the applications is the adaptation of simple existing circuits and integrated circuit packages to work with the inexpensive computers. Software in the form of easily understandable BASIC language and very efficient machine code is included where appropriate and explained with each hardware application.

After piquing your imagination as to the variety of yet unexplored applications for computer control in the home you are guided to selecting the most suitable computer from the standpoint of both cost and capability for the projects described in the book. Computer expansion using a commercially available bare board circuit is detailed and illustrated. Then a full description of two inexpensive input/output circuits is provided, one of which is available commercially in bare board version.

Chapter 5 is the heart of the book illustrating computer control application in the form of the electric and electronic circuits and computer control of a mobile self contained robot. Most of the projects that follow have direct application to the robot development. The next project applies the same principles as used to control the robot to control appliances and lights in the home.

The remaining projects include a simple but potenitally powerful stepper motor control circuit; a simple optical encoder project which can be applied to numerous control applications; an inexpensive computer controlled digital voice system; a good demonstration of voice recognition using simple hardware and an intriguing machine code software technique; and, the implementation of an ultrasonic ranging sensor. The text concludes with a number of hardware and software tips.

Also included is a list of reference publications for further study and a comprehensive list of parts sources for all projects in the book.

The approach of the book is unique among "how to build it" books because it neither requires the reader to buy an expensive kit nor does it require the reader to build all circuits from scratch. Instead, it provides anyone who wants to build the projects and in the process learn about computer control, the option of inexpensive components and in several cases, bare board printed circuits available commercially. It uses technology readily available to the consumer to explore the rapidly growing computer control and robotics field.

Chapter 1
GETTING CONTROL

Although one of the most visible manifestations of computer control is robotics, the computer can be adapted to control virtually anything. I will present some practical computer control applications in this book, including actual circuits, parts sources and software. But first, let's explore some "ideas" that you might pursue. The aim in this book is to take existing components, wherever possible, and adapt them to your use or purpose.

The approach in this book is not to build a computer from scratch! You are probably neither willing nor feel capable of that type of project. I will instead, describe projects that adapt existing components to accomplish computer control with nowhere near the construction complexities of building a computer from scratch.

I will assume that you are either now a tinkerer, or I will pique your interest enough with this book to become a tinkerer. The objective of all projects in this book is to take something good, inexpensive and simple, and make it into something even more useful than it was originally.

WHAT CAN BE CONTROLLED?

My answer to that question is that virtually anything and everything can be computer controlled. Look around you, where ever you are right now, and let your imagination run wild. Believe me, it probably can be computer controlled and the result should be a more efficient use or operation.

Computers in the form of limited function microprocessors and digital logic circuits are appearing in more and more applications in the home and work place. One example in the home is the energy saving heating/cooling thermostat controls. Another is the video text terminal and home computer information service terminal. Electronic communications devices bring a wealth of information to the finger tips of the home user. But why stop there? Why not tie the two together to perform even smarter home control activities? If the temperature outside your home is 60 degrees but is predicted to rise to 75 in a few hours, why not use this information for a smarter thermostat control operation? The furnace activity could be cut back, taking advantage of the warming trend.

Or, why not a computer controlled home security system that performs dual duty as an energy saving system? A security system which monitors room activity for intrusion detection, could contine to monitor activity when not doing security duty. The information could be used to turn off lights and cut back on heating/air conditioning in unoccupied rooms. The possibilities are limitless. But what of the control possibilities involving a home or personal robot? Can a personal robot accomplish any useful tasks around the home?

WHAT SHOULD A ROBOT DO?

In the words of the Isaac Asimov, author of science fiction tales of robotics, robots are simply mobile computers. More specifically, personal robots are mobile platforms controlled by computers. Without a doubt, personal robots are the most intriguing of the computer control applications. The answers to not only what a robot is, but also what a robot should do, were discussed at length during the first International Personal Robot Congress (IPRC) & Exposition 1984, held in Albuquerque, New Mexico 13-15 April 1984 (Figure 1-1). Isaac Asimov, the man who coined the term "robotics" in 1941, delivered the keynote speech to open the IPRC. Three other featured robotics speakers, David L. Heiserman, Joseph F. Engleberger and Noland K. Bushnell along with Mr. Asimov provided much insight on the personal robot subject. The theme of the Congress, "Be Part of the Beginning", was carried forward in the enthusiasm of these robotics experts. Let's explore their thoughts on the subject.

Mr. Asimov's three laws of robotics must be reviewed as a part of any discussion of robot functions. The three laws are

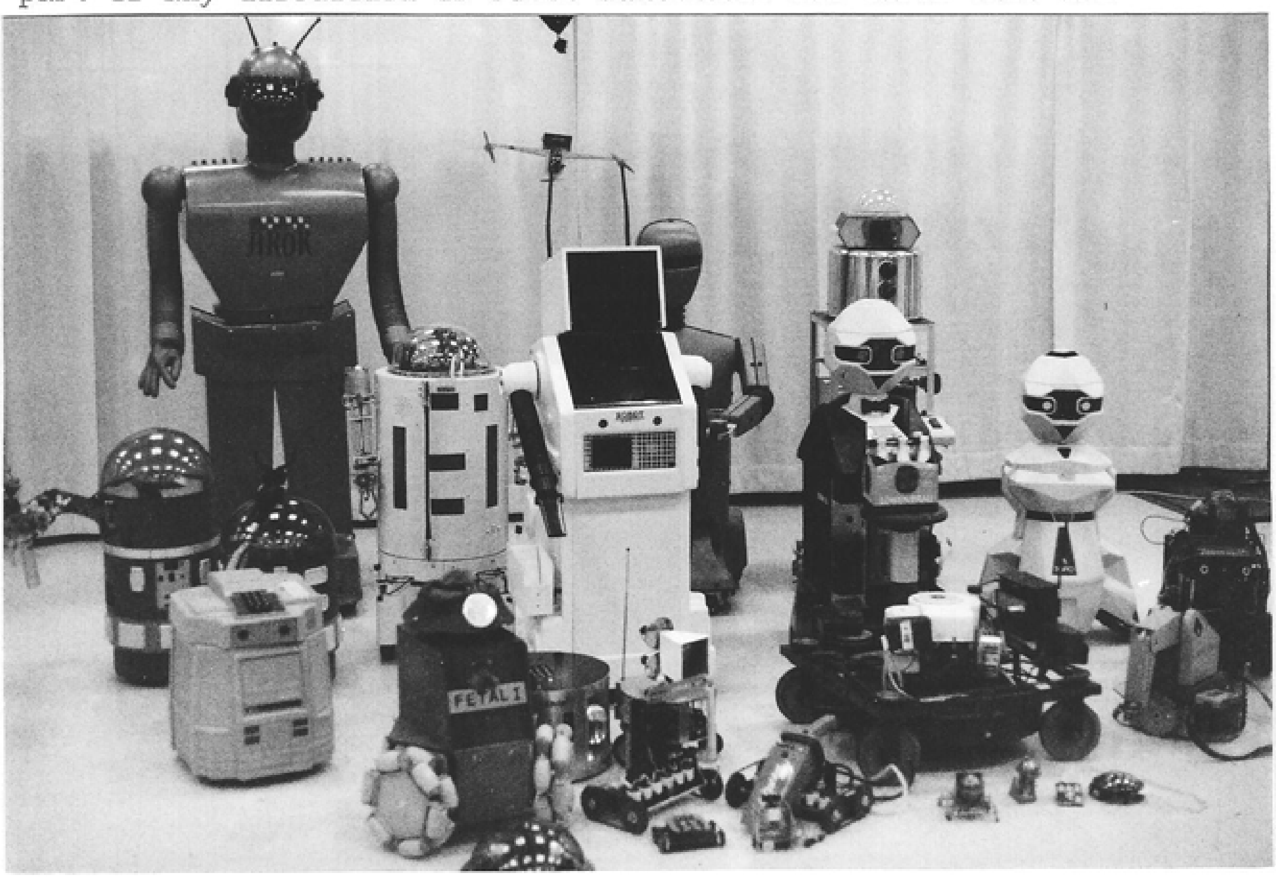


Figure 1-1. H.E.N.R.Y. and the gang at the first International Personal Robot Congress & Exposition 1984. Group included both individually and commercially developed machines.

briefly, (1) a robot must not injure or cause injury to others, (2) a robot must obey orders except when it would violate the first law, and (3) a robot must protect its existence except where it violates the first two laws. Mr. Asimov feels that industrial robots are currently too simple and do not yet have the intelligence to incorporate the three laws. He feels that robots and computers do not now and never will have the same abilities as humans. His point is that it is a waste of time to imitate human actions, because computers and robots just can't do as good a job at many functions as a human. His premise is that computers and robots will get better and better at doing what they can do. On the other hand, humans will also get better and better at what they do. Together they will accomplish a symbiotic effect and do a better job together than either could do independently.

Robot Builder Views

Mr. Heiserman was probably the first to write a book describing how to build a personal robot. He has written several books on the subject and his ideas have been an inspiration to my work in the subject. I totally agree with Mr. Heiserman's point that there are so many things we can do with robots, so why aren't we doing them? His build your own robot book describing "Buster" published in 1976 was helped immeasureably by the release six months later of the movie "Star Wars". His books embody an evolution of ideas in robots progressing through microprocessor control to a self programming robot concept. Mr. Heiserman's latest interest is how a machine (the robot) percieves the universe or how it copes with the enviornment?

Mr. Engleberger is generally acknowledged to be the founder of industrial robots and robotics. Based on his extensive background in industrial robots, he has definate ideas concerning what a robot is and what it should do. He found it difficult to pin down a precise definition of an industrial robot and even harder to come up with a definition of a personal robot. He does state though that, "I know one when I see one". You can form a very definate picture of his idea of what a robot is, by studying his opinion of what a robot must and can do.

Mr. Engleberger has strong feelings that a personal robot must be able to do something useful and must sense and interact with the enviornment. He also feels that it has to think in world coordinates; have computer control; possess dead reckoning; inertial navigation; and, scene analysis. Mr. Engleberger evaluates a robot without an arm as a mere computer rolling around on a cart.

Mr. Engleberger has a long list of possible applications for mobile robots. They include garbage collection and fast food preparation (intentionally grouped together); gas station attendent (safety and convenience advantages); hospital aides; and, household servant (including fire & intruder emergency service). A major point was that he doesn't think that personal robots have to be cheap. This position is in apparent conflict with the opinions of Mr. Bushnell, to be discussed next.

Some Differing Views

Mr. Bushnell, the man who brought us Atari video games and Chuck E. Cheese's Pizza Time Theater, has some different views on what personal robots should be and do. He insists that personal robots must be cheap and fun. He feels that "fun sells" and personality is cheap. He has been amazed at how people will assign personality to random number generators in video games. Speech in a robot also adds personality.

He also feels that it is possible for a personal robot to do meaningful tasks in the structured enviornment of the household. Mr. Bushnell outlined his idea of the evolution of the personal robot over the next ten years. In the next three years, the robot that tells the best jokes is going to be the winner. In four to six years, the one that guards the house will be sold. And, in eight to ten years, he sees the robot picking up, manipulating, washing a window, bringing a beer, and performing as a surrogate child.

Enough Philosophy

The robot projects described in this book have much in common with the ideas of these experts. My ideas concerning what a personal robot can or should do around the house are still being formed. These ideas are evolutionary and had humble beginnings a few years ago when H.E.N.R.Y., my personal robot was born. Others don't always see what you see, as evidenced by an early view of H.E.N.R.Y. and I by a family friend which can be seen in Figure 1-2.

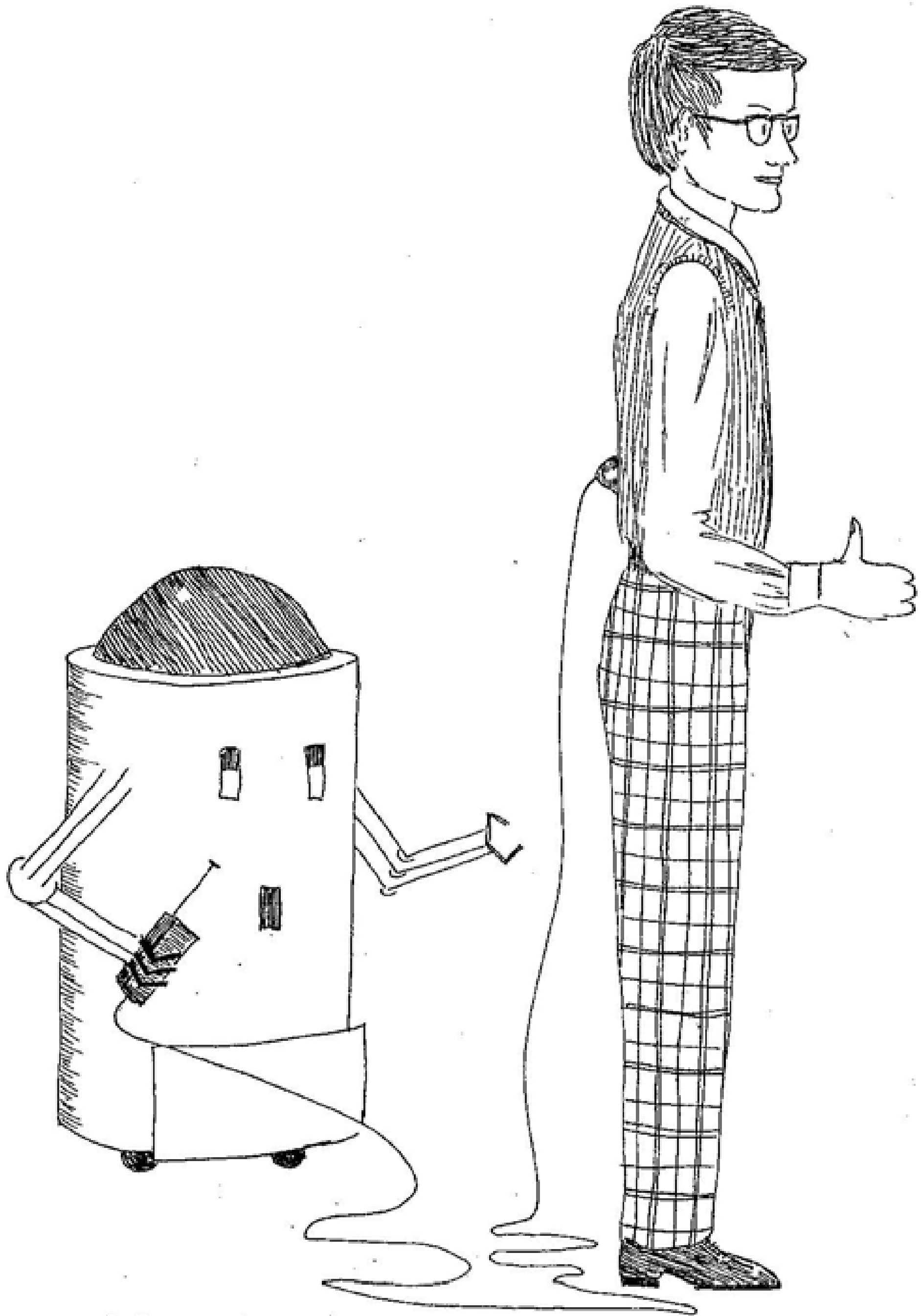
The humor seen in my robot was not lost on the judges at the first International Personal Robot Congress & Exposition. H.E.N.R.Y., whose construction, circuits and software are described throughout this book, won the Golden Droid Award for Most Entertaining at the IPRC (Figure 1-3).

Although the second and third IPRCs (1985 and 1986) were originally scheduled to be held in Albuquerque, the second was held in San Francisco in September 1985. If you consider yourself a real robot enthusiast, you should plan to attend an IPRC in the future. Why don't you go and learn the latest as to what the commercial personal robot manufacturers and the tinkerers are up to? See you there!

SELECTING THE COMPUTER

The selection of the proper computer is key to inexpensive robot control applications. Remember, the theme of the projects in this book is "keep it inexpensive". Although the control circuits described can be adapted to any popular Zilog Z80 Central Processor Unit (CPU) based microprocessor (i.e. early Radio Shack TRS-80 models, Heath H-89, etc.), I chose the Sinclair ZX81 computer (Figure 1-4) for several reasons. This computer has also been marketed in the U.S. as the Timex/Sinclair 1000 and the Timex/Sinclair 1500. Although these computers are not currently being manufactured for sale in the U.S. by either Sinclair or Timex, several million have been sold world-wide and are still being sold both in the U.S. and by Sinclair outside the U.S. Hundreds of thousands of these computers bought in the U.S. as first computers are now gathering dust in closets. Those that can be still found for

Figure 1-2. H.E.N.R.Y. and I when H.E.N.R.Y. was young (Drawn by H.E. Kilp)



"IT'S A B.R.U.C.E. IT REALLY DOESN'T DO ANYTHING USEFUL AT ALL, BUT I FINALLY GOT IT TO STOP BUMPING INTO WALLS."

HEK

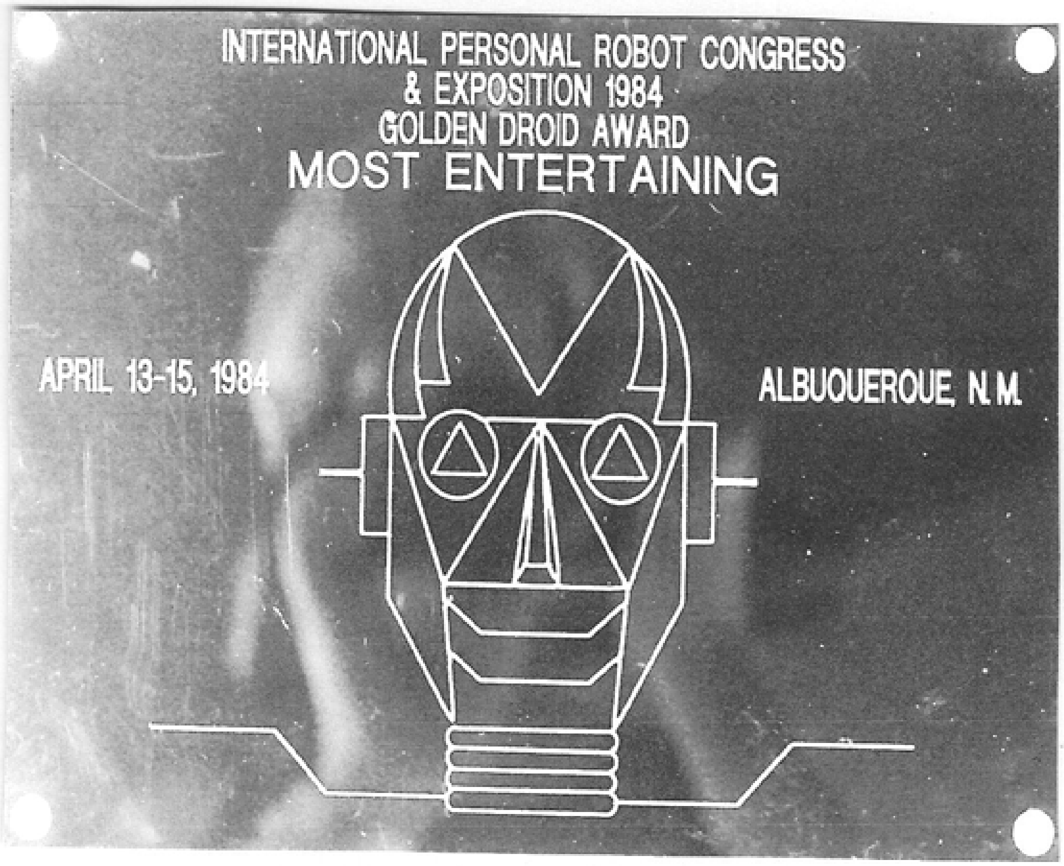


Figure 1-3. H.E.N.R.Y.'s Golden Droid Award.

Figure 1-4. TS1000 & ZX81 computers.

can be still found for sale, new or used, can be bought for approximately $30 to $40 or less.

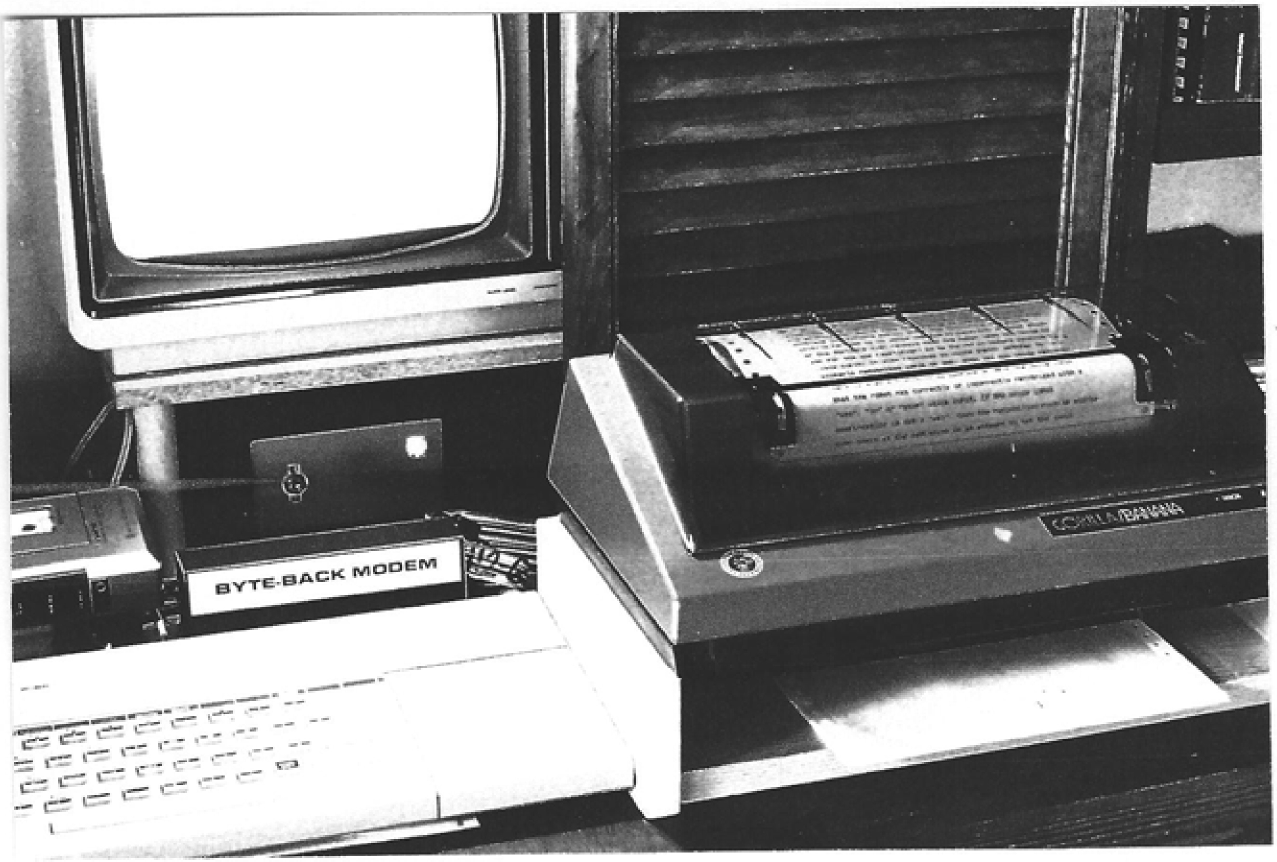
It is unfortunate that these computers are considered by many as "toys" and not respected for thier potential capability. It is educational to explore the apparent failure of the Timex/Sinclair computer venture in the U.S., while Sinclair computers continue to sell well in most of the rest of the world. You can't lay all the blame on Timex. Computer stores won't touch computers selling for less than a couple of hundred dollars because there is not enough profit. They can make much more by selling a $2,000 computer. You can't blame the computer stores, they are in business to make a profit.

I guess the major blame for the apparent failure of the Sinclair market in the U.S. belongs with Sinclair itself. They made the machine too good and sold it at too low a price! Although the ZX81/TS1000 machines have shortfalls, they have some features not even found in the "big" machines. One such feature is the single key BASIC language entry, using only one byte of memory per instruction and performing syntax error check immediately upon entry of BASIC statements.

So why am I into supporting the Sinclair machines? Because I got into computing for the fun of it (fun can be cheap). With the Sinclair machines, you can have fun and learn a lot, for only a little money. The same fate may strike the new Sinclair QL computer in this country that befell the other Sinclair computers here, because they don't cost enough. The Sinclair QL is a fantastic machine, which for less than $300 will run rings around many machines selling for several times as much. But what computer store will handle it and settle for three to four times less profit per sale? I predict that if the Sinclair QL is not marketed under some new and creative scheme, it will be a failure in the U.S., even though it is a superior and inexpensive machine.

I must devote a few words to another fine Sinclair product, even though it is no longer manufactured for sale in the U.S. by Timex either. The Timex/Sinclair 2068 Personal Color Computer, a version of the Sinclair Spectrum being marketed worldwide, was sold in the U.S. by Timex from the fall of 1983 into 1984. Again, the only thing I can see that is wrong with this computer is that it also was too inexpesive. It is in many ways superior to the very popular Commodore 64 and originally sold for about the same price. In fact, several of the projects described in this book also include instructions for use with the TS2068. Working with this machine has truly been a pleasure, and on a budget too. This book was written using only the setup shown in Figure 1-5. It consists of a TS2068 computer, a Byte-Back RS-232 interface, a Gorilla-Banana (Leading-Edge) printer and Tasword Two word processor software (from Tasman Software, U.K.). And what did this budget operation cost to set up? I am dying to tell you. It cost less than $500! I have used $10,000 word processors and this setup has nearly the same capability at a fraction of the cost. When BRC acquired the rights to publish this manuscript, after TAB Books owned the rights for over a year and didn't publish, the only equipment added was a Tandy DMP-130 printer. It is a fine dot matrix printer which produced the near letter quality camera ready pages you are reading.

Figure 1-5. Low cost word processor setup used to write book.



Why The ZX81/TS1000?

Although I have already mentioned some of the advantages of the ZX81 and TS1000 computers, the most relevant advantages are discussed below. The computer comes with a complete single key entry BASIC language in ROM (Read Only Memory). You will see other single board computers advertised with a tiny BASIC but it is just that, and not a complete BASIC language. Simple machine code programming of sophisticated control routines is easy to implement using the ZX81/TS1000. Several of the projects described in this book include machine code routines and an explanation of how to implement them. Also, Chapter 12 expands and explains the machine code implementation on the Sinclair computers. Other tools like FORTH language are avialable from the cottage industry suppliers. Although these suppliers shrank somewhat when Timex left the U.S. market, they are still around, in the U.S., in Canada and of course in the U.K.

The overriding justification for choosing the Sinclair computers for demonstration and control applications is the low cost or "throw away" nature of the computers. This is important in overcoming the natural reluctance to wiring up attachments and fearing the worst might happen, damage to your personal computer. Yes, I damaged the ZX81 once with an improperly wired peripheral and once with a static electricity discharge. In both cases the repair required the replacement of IC1, the controller chip, at a cost of $12 each. These replacement parts can be obtained from Sinclair Research, Nashua, NH for the ZX81 or from Timex Computer, Little Rock, AR for the TS1000 or TS1500. Also, a chip from either source will work in either computer.

A prime consideration in selecting the Sinclair computers for a project like a self contained robot is the small size of the computer. Ounce for ounce and cubic inch for cubic inch, it can't be beat.

Will Other Computers Work?

If you insist on using some other computer in conjunction with these projects either because you haven't got a ZX81, TS1000 or TS1500 or you can't find one at a flea market or in someone's closet, as mentioned earlier, it can be done. However, it will require that you have extensive knowledge of the particular computer. Basically any computer with a Z80 could be adapted to use with many of the circuits described. These include the TRS-80 Model 4, Heath H-89, Spectravideo SV-318, Mattel Electronics Aquarius, Toshiba America HX-20, Cromemco C-10 and Osborne I to name a few. For example, the optical encoder circuit described in Chapter 7 was adapted from a circuit originally designed for use with the TRS-80 Models I & III. However, in order to adapt these circuits to other computers you will require some knowledge and information not provided in this book. The best way to pursue this adaptation is to obtain schematics and as much other information on both the Sinclair computer and whatever computer you are adapting the circuits to work with.

Well, this is the end of the preliminaries and small talk about computers and computer control. The rest of the book is devoted to construction, wiring circuits, writing software and making things happen. Learn by doing and do it inexpensively. Good luck on your computer control projects.

THIS PAGE INTENTIONALLY BLANK

Chapter 2
COMPUTER EXPANSION

Although one or two peripherials can be added to your computer without requiring an expansion board, a fully buffered expansion board is a requirement for the serious control experimentor. An expansion board is a must for projects such as a personal robot, whose electronics are described in Chapter 5.

An expansion board can be built from scratch with the information provided in this chapter. However, a much easier and more foolproof way to build the board is to buy the bareboard version of the circuit from a commercial source. The board fully described here was originally designed and hundreds sold by Computer Continuum of San Francisco who no longer make or sell the board. An approved copy of this board is now available from Budget Robotics and Computing of Tucson, Arizona. This circuit is actually a combination of the old and new versions made by Computer Continuum. It combines the pinout of the output connections on the old version with the improved decoding circuitry of the new version. This pinout design duplicates the pin out on the rear of the computer enabling any device designed for connection to the back of the computer to be plugged directly onto the expansion board. For example, the RX81 I/O board described in the next chapter can be plugged directly onto the expansion board. In this case, the most in the way of an adapter that would be required for any device would be a piece of printed circuit board with fingers extending the complete length of the board (see Figure 2-1).

An important aspect of an expansion board is the availablity of a larger +5 volt power supply than the one that is built into the computer. This is important as you keep adding

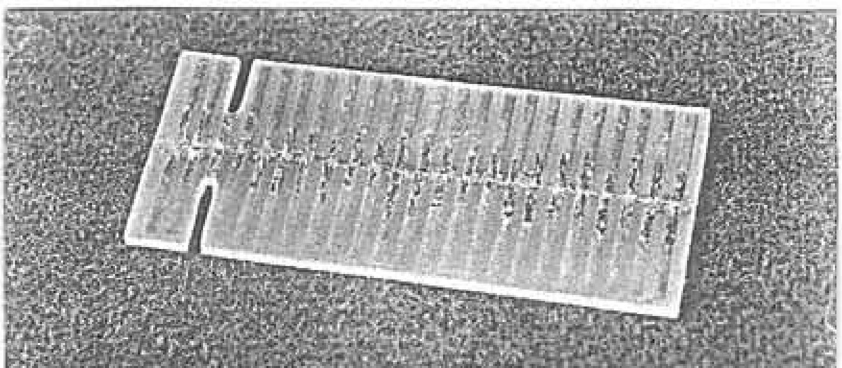


Figure 2-1. Double sided 23 x 2 finger extender for use with Expansion Board.

more peripherials, as there is no need to add 5 volt regulators for each project with 3 amps of power is available directly from the expansion bus. Another important aspect of the expansion board power supply is that you can supply the +5 volts required by the computer, from the expansion board, bypassing the computer's 5 volt regulator. Those who have experienced computer overheating know what an important advance in computer reliability this is.

THEORY OF OPERATION

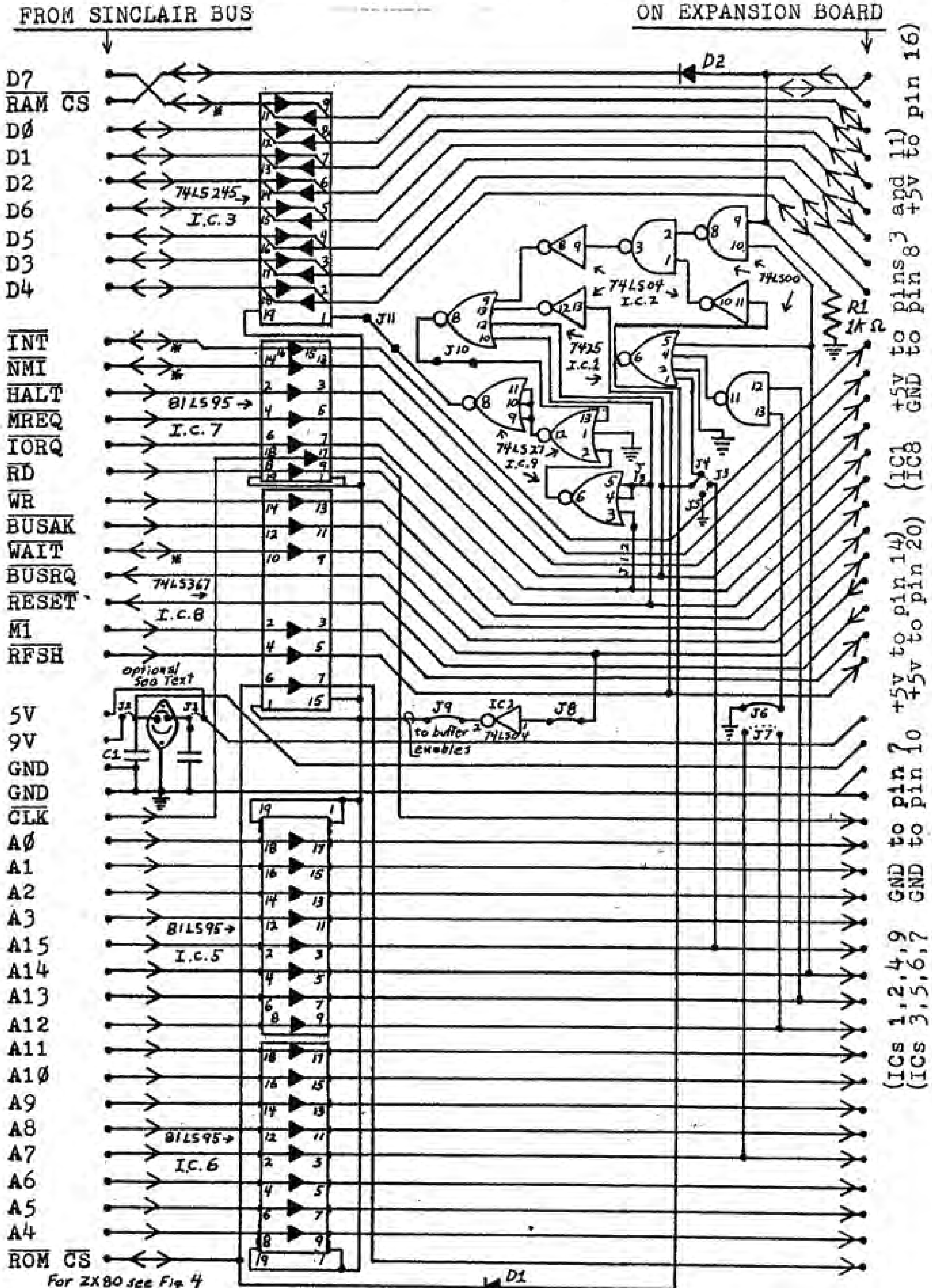
You can refer to the schematic (Figure 2-2) when tracing the operation of the logic described here. The buffers used are the 74LS type (Low power Schottky) and can each drive up to 10 LS loads. All buffers in address and control bus are directing logic away from the Z-80 Central Processor Unit (CPU) in the computer. Two control lines, RESET and BUSRQ are left unbuffered and are used purely as inputs to the CPU. INT, NMI, and WAIT are CPU inputs but are already used by the ZX81/TS1000 structure. It is dangerous to impose any data on these lines, for that reason the buffers are in the write direction only. For those who wish to impose logic on any of these inputs, you may easily unbuffer a line by pulling the buffer's output pin and making a solder bridge across the appropriate adjacent buffer's pads.

The circuit contains two kinds of input/output mapped I/O: I/O mapped I/O and Memory mapped I/O. The computer keyboard, ZX and TS printer and RX81 I/O board are I/O mapped. Therefore the optional circuit at IC9 is required to make these peripherials work by sensing READ and IORQ which goes active low when an I/O port is called to input data into the CPU. The IC9 circuit imposes a high at the direction pin of the data bus buffer to input to the computer during that read cycle. I/O mapping uses the lower 8 bits of the address bus thus the hardware needed to fully decode a peripherial is simpler. However, memory mapped I/O is easily programmed from Sinclair BASIC whereas I/O mapping requires a small machine language program. The next chapter will detail both applications with the RX81 using I/O mapping, with machine code examples and the 8255 PPI chip using memory mapping with decoding fully explained.

BUSRQ (busrequest) feature. If an active low signal is placed on the BUSRQ line on the Expansion Board, the buffers will all be automatically sent to their high impedance (off) state. This is useful for advanced projects such as with Direct Memory Access (DMA). See the schematic, and note the inverter between BUSRQ and the buffer enables.

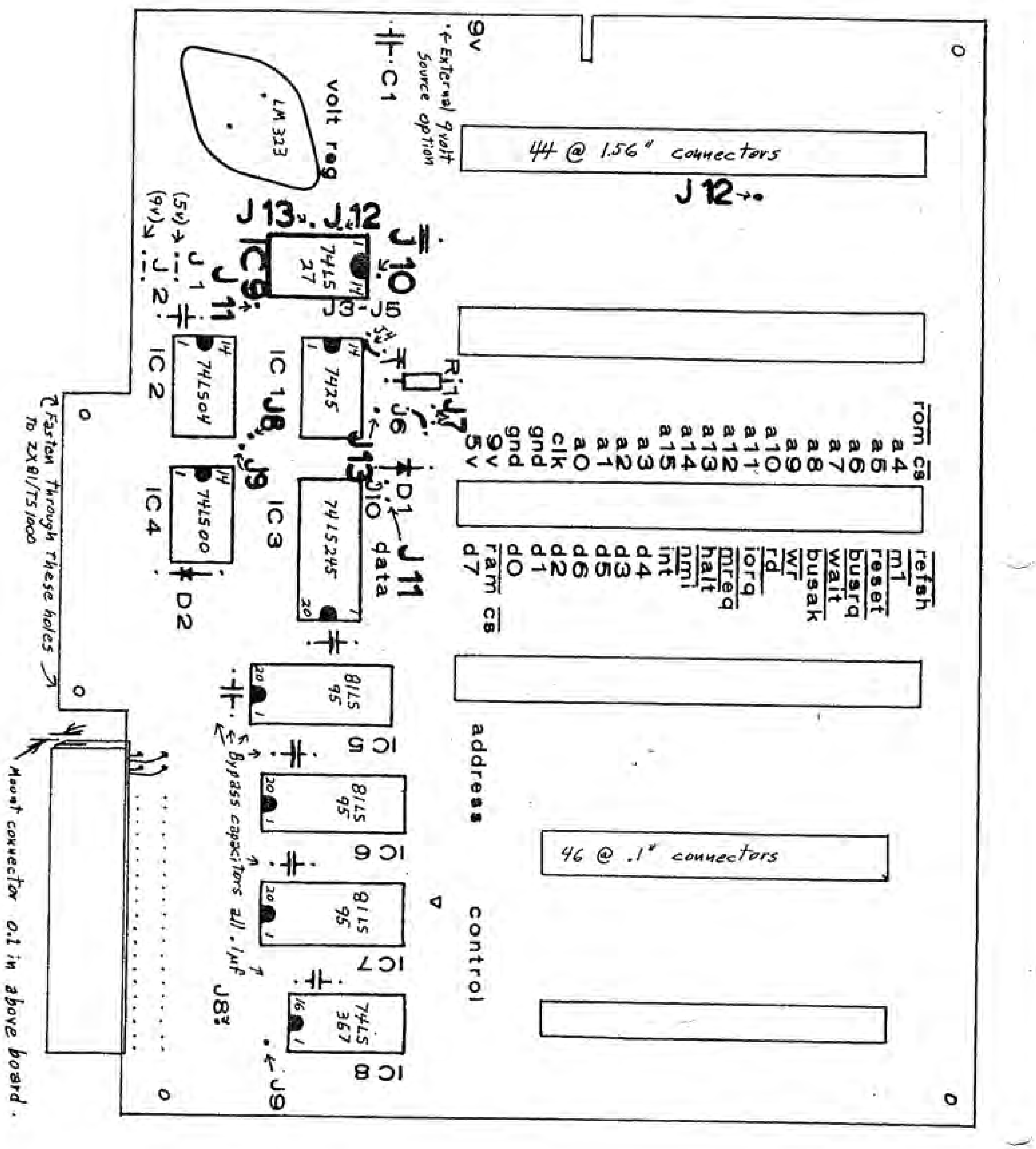
The data bus buffer is normally writing in the outward direction. The decoder switches the direction inward only when the necessary conditions are met. The decoder can be analysed in two sections: (1) decoding for RAM read, and (2) decoding for memory mapped 'I/O space' read. The reason for this separation (which barred further simplification) is that the signal to the ROM CS had to be sent in response to a read from memory locations in 'I/O space'. For those desiring further details on the decoding of the data buss, a complete explanation of the equation for the decoder comes in the documentation accompanying the Expansion Board.

Figure 2-2. Schematic diagram of buffered buss Expansion Board.



* Signal directions not accomodated by buffers. If needed, unbuffer with solder bridge and pull output pin on chip.

Figure 2-3. Expansion Board legend.



LAYOUT AND CONSTRUCTION

Refer to the bare board Legend (Figure 2-3), the schematic (Figure 2-2) and the Parts List (Table 2-1) to follow these instructions. (Note: An optional method of wiring this board is included in the current board documentation from BRC for use with Paul Hunter's non-volatile memory boards. This optional wiring is also superior for the control projects in this book.)

Table 2-1
EXPANSION BOARD PARTS LIST

(1) 7425 (IC1)
(1) 74LS04 (IC2)
(1) 74LS245 (IC3)
(1) 74LS00 (IC4)
(3) 81LS95 (IC5-7)
(1) 74LS367 (IC8)
(1) 74LS27 (IC9)
(1) LM323
(4) 14 pin DIP sockets
(1) 16 pin DIP socket
(4) 20 pin DIP sockets
(4) 22/44 @ .156" edge card sockets (Digi-Key # C1-22)
(1) 50/100 @ .1" edge card socket (Digi-Key # C5-50)
(1) 1000 ohm 1/4 watt resistor
(2) 1N914 switching diodes
(9) 0.1 uF 10 volt disc capacitors
(1) 1 uF 20 volt capacitor
(1) TO3 Heat sink
(1) Expansion Bus bare board with edge connector (Budget Robotics & Computing)

Mount the IC sockets, diodes, resistor and capacitors. Note that each IC is accompanied by a pair of holes for a bypass capacitor. These cut down the noise which can travel over the power supply. Use any disc capacitor within an order of magnitude of 0.1 uF. All of these may not be necessary. If the 9 volts is supplied via the expansion board, C1 at 1 uF is advised.

Orient the computer connector (provided with the board), computer and Expansion Board to each other (Figure 2-4). Find the two pins that correspond to the keyway and clip their solder tails. Push a keyway insert into the proper contacts. On the ZX81/TS1000 you must mount the connector approximately 0.1 inch above the surface of the board. You can arrange a perfect fit by assembling the connector, expansion board and computer, then soldering the connector in place. You may want to fasten the board to the computer with two 1/4 x 4 screws or other suitable fastener after the system is completely assembled and checked out.

The 100 pin @ .1" edge connector will have to be cut into two 45 pin (22 pin x 2 plus empty slot) connectors with a hack saw or Dremel tool grinder/saw. Solder all edge connectors into place.

The LM323 5 volt, 3 amp voltage regulator is highly recommended and previously explained. Install after all IC

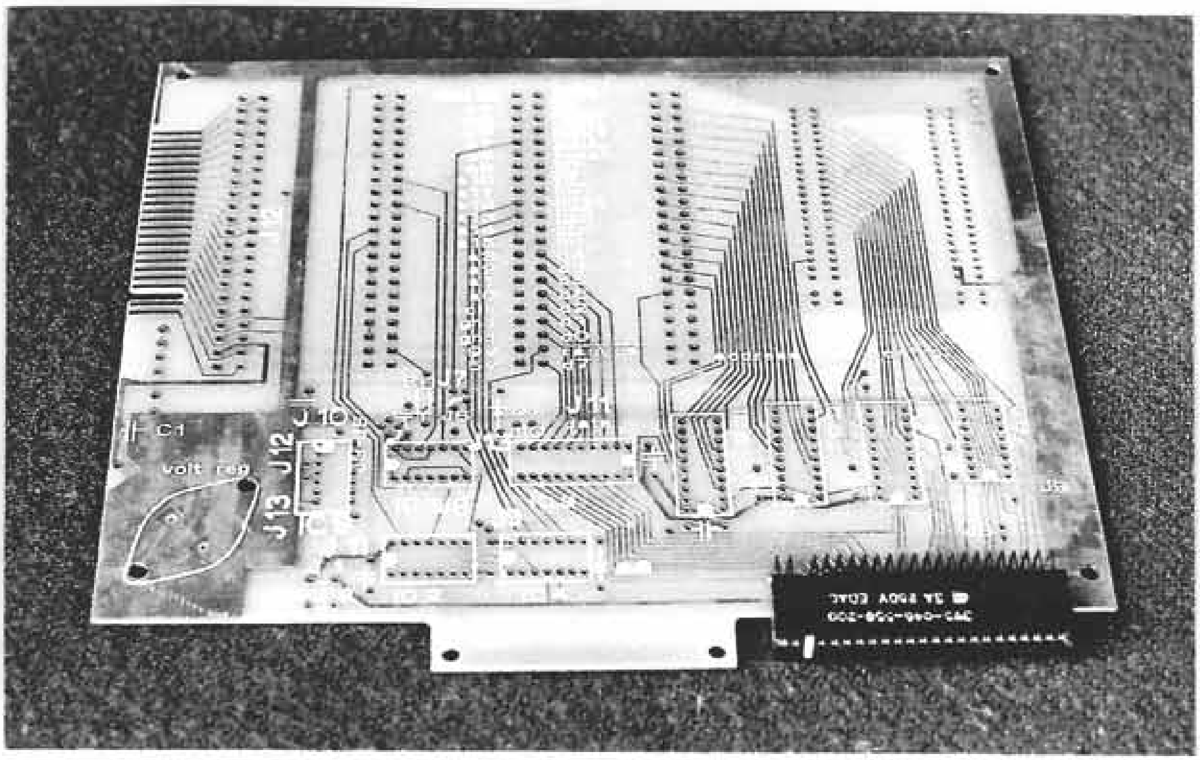


Figure 2-4. Expansion bare board with 90 degree connector.

sockets and jumper wires have been installed. Fasten it with the heat sink with bolts before soldering.

Jumper Wires

Jumpers J1 and J2. For +5 volts supplied from the computer's regulator to the Expansion Board, use J1 and J2. For +5 volts supplied from the Expansion Board to the computer, use J1 (do not use J2) and supply the 9 volts to the regulator at the point on the Board marked '9 volts'. For power supplied from the computer's power pack via the computer to the 3 amp regulator, use J2 (Do not use J1). For power supplied from external 9 volt source to the 3 amp regulator and the computer's regulator, use J2 (Do not use J1).

Jumpers J4 and J6. To keep the ROM from repeating at addresses 8192 to 16383 use J4 & J6. This will give you 8192 (= 8K) of free addresses for memory mapped peripherals or more memory. Either J4 & J6 or J3 & J7 must be used.

Jumpers J3 and J7. To keep the ROM from repeating at addresses 12288 to 16383 use J3 & J7. This gives you 4K of free addresses.

Jumpers J8 and J9. Prepare two wires which will hang on the solder side of the board from IC2 to J8 and J9. Connect one end of J8 to pin 1 of IC2 and the other to the BUSRQ end of J8. Then connect one end of J9 to pin 2 of IC2 and the other to the buffer enable side of J9. BUSRQ will remain active low (0 volts). If you chose not to use this recommendation, then the end of J9 connected to the buffer enables must be connected to ground.

Jumpers J10, J11, J12 and J13. This is a data bus buffered external decoder option. It is required for operation of such peripherials as the Sinclair or TS printer and the RX81 I/O board. Prepare four wires which will hang on the solder side of the board from IC9 to various points. Connect J10 from pin 8 of

IC1 to pin 13 of IC9. Connect J11 from pin 1 of IC3 to pin 8 of IC9. Connect J12 from buffered IORQ to pin 3 of IC9. Connect J13 from pin 10 of IC1/buffered RD to pin 4/5 of IC9. If you would like to experiment with another decoder option and not connect IC9 into to circuit, the following must be accomplished: wire one jumper connecting the pin 8, IC1 side of J10 to the pin 1, IC3 side of J11.

Mounting the computer

There are many options for mounting the expansion board and computer (Figure 2-5) depending on the application. For the Home Control project (Chapter 6) you may wish to build a box to protect the boards you add to the expansion board. Be sure to somehow insulate the bottom of the expansion board from accidentially coming into contact with metal objects to prevent short circuits. In the robot project (Chapters 4 & 5) you may want to substitute a flexible wire connector for the 90 degree connector so that the computer can be mounted underneath the computer on the same mounting board.

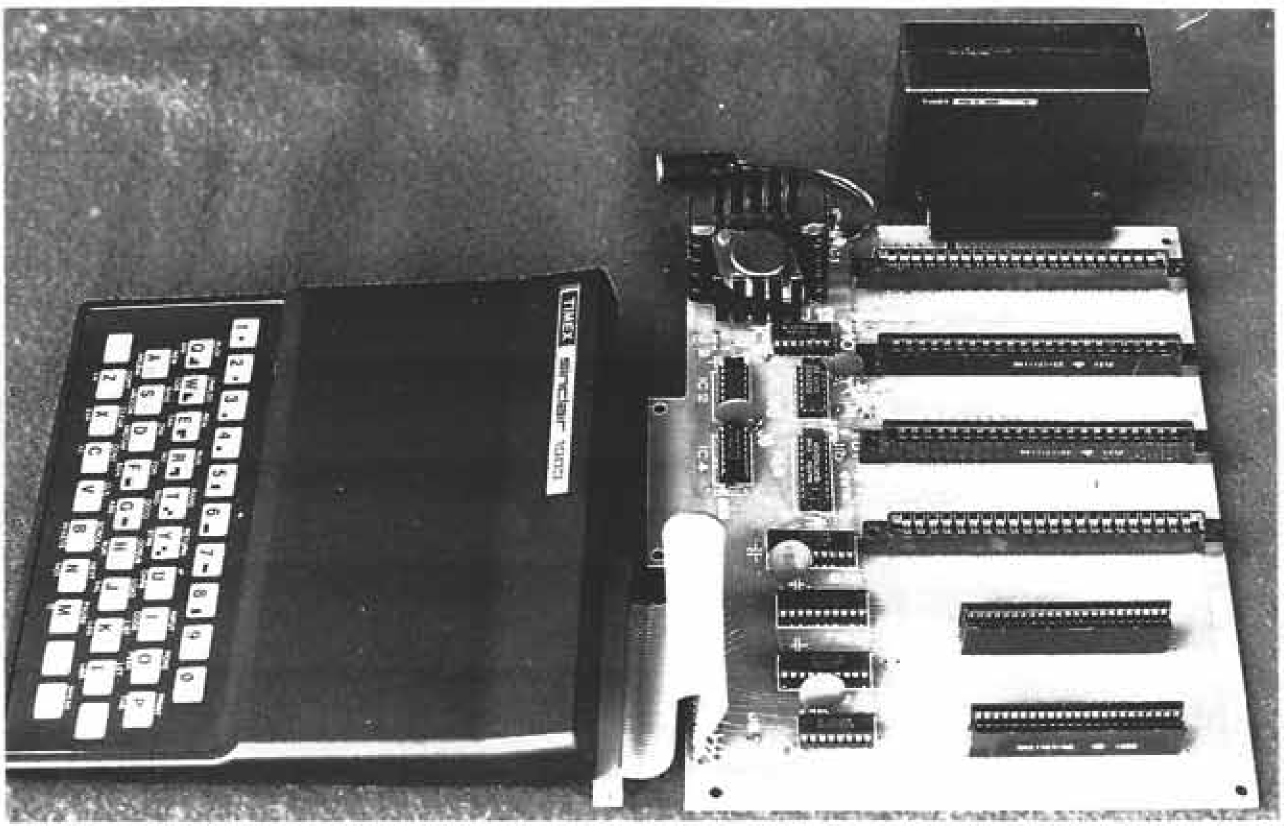


Figure 2-5. Built up expansion board with 16K RAM and computer connected with flexible connector.

Chapter 3
INPUT/OUTPUT CIRCUITS

I will describe two different input/output (I/O) circuits, one available commercially in bare board form and one based on the 8255 PPI I/O chip. Both circuits are inexpensive. The commercially available bare board is the RX81 which uses four common integrated circuit chips and can be programmed through very fast machine code. The basis for the other I/O circuit is the Intel Corporation 8255 Programmable Peripherial Interface I/O chip which was designed for the 8000 microprocessor family but is easily adapted for use with the Zilog Z80, the CPU microprocessor in the Sinclair and T/S computers. Both I/O circuits are adapted to applications in this book and can both be used together when the expansion board described in the previous chapter is used. These I/O circuit adaptations are the key link between the computer and the peripherial, making it possible to output to or input from virtually any device. The remaining chapters contain a wide range of such application possibilities.

RX81 INPUT/OUTPUT BOARD

An improved version of the RX81, formerly available from ZODEX, is now available in bare board form, from Budget Robotics & Computing, Tucson, AZ. The RX81 is a tiny circuit which, when attached to the Sinclair/Timex or Sinclair computers, enables on to turn on/off eight LED displays built-in and read the conditions of eight switches via commands written in either machine code or a combination of machine code and BASIC language. The eight switches and LEDs represent the many possible applications of this input/output building block.

An output RESET button is provided to allow manual shutdown of all output devices. All outputs can be automatically reset on system power up.

The RX81 power loading effect on your computer is negligible. One LS load on all LEDs out equals approximately 20 milliamps.

Although the circuit was originally designed as a learning tool, with improved software it now may used with discretion in many control applications. If you wish to extend the capabilities of the basic unit, this system provides you easy access to all input and output points with circuit suggestions for interfacing in this chapter and specific projects described in later chapters.

The software uses machine code stored in one simple REM statement for the ZX81/TS1000/TS1500 and in a DATA array for the

TS2068. Turning on an output takes a minimum of two instructions and reading an input takes a minumum of one, in BASIC language.

The following is a circuit description following the schematic in Figure 4-3, from right to left. The LEDs are powered through 1.2K ohm limiting resistors. IC4 is given an eight bit word by the databus D0 through D7. That word is transferred accross the latch to light the LEDs, when OUT 7 is sent out in machine code.

The DIP switches are attached to the inputs of IC2 with 4.7K ohm pull up resistors. When a switch is closed, the voltage is pulled down to zero. When IN1 is sent in machine code, the ones and zeros are placed on the data bus D0-D7 and read as an 8-bit word. When you are communicating with either device you are working with a base 2 number (a full example is included later in the software description).

IC1 on the schematic will turn on the latches only if the computer is reading (RD) or writing (WR); an I/O is requested (IOREQ);and, the A7 (address 7) line is low. It generates four INs and four OUTs, only two of which are used (one IN and one OUT) per board, at a time. You can modify any board to change the IN or OUT lines being used.

The diode and capacitor with the momentary switch provide a reset capability. The circuitry minus the switch provides automatic reset on power up.

Extra filtering on the 9 volt supply is provided with the 47uF capacitor.

Component Layout

Assemble the parts listed in Table 3-1. Arrange the IC sockets as shown in Figure 3-2. Prepare the connector by hacksawing off the extra pins to make up a 46 pin mate for your computer edge connector (this will also work on TS2068). Glue the connector to the circuit board's edge, trimming off the board on either side of it. Bend down the connector tabs and solder them to the fingers on the board. Install connector key (third slot). Remember, if you are using the RX81 with the expansion board the connector would go on the expansion board, not the RX81. The RX81 would then be plugged directly onto the expansion board.

1. Tie in your power (5 volts) and ground to all components.

2. Continue to solder the connections per schematic.

3. Verify all connections with an ohmeter or beeper.

4. Check that 5 volts and 9 volts are not shorted together or to ground.

5. Connect optional reset circuitry, if desired.

6. Insert the appropriate ICs into their sockets (pins as shown).

7. As always, with power off on the computer, plug the RX81 onto the edge connector, aligning the slot with the key on the connector.

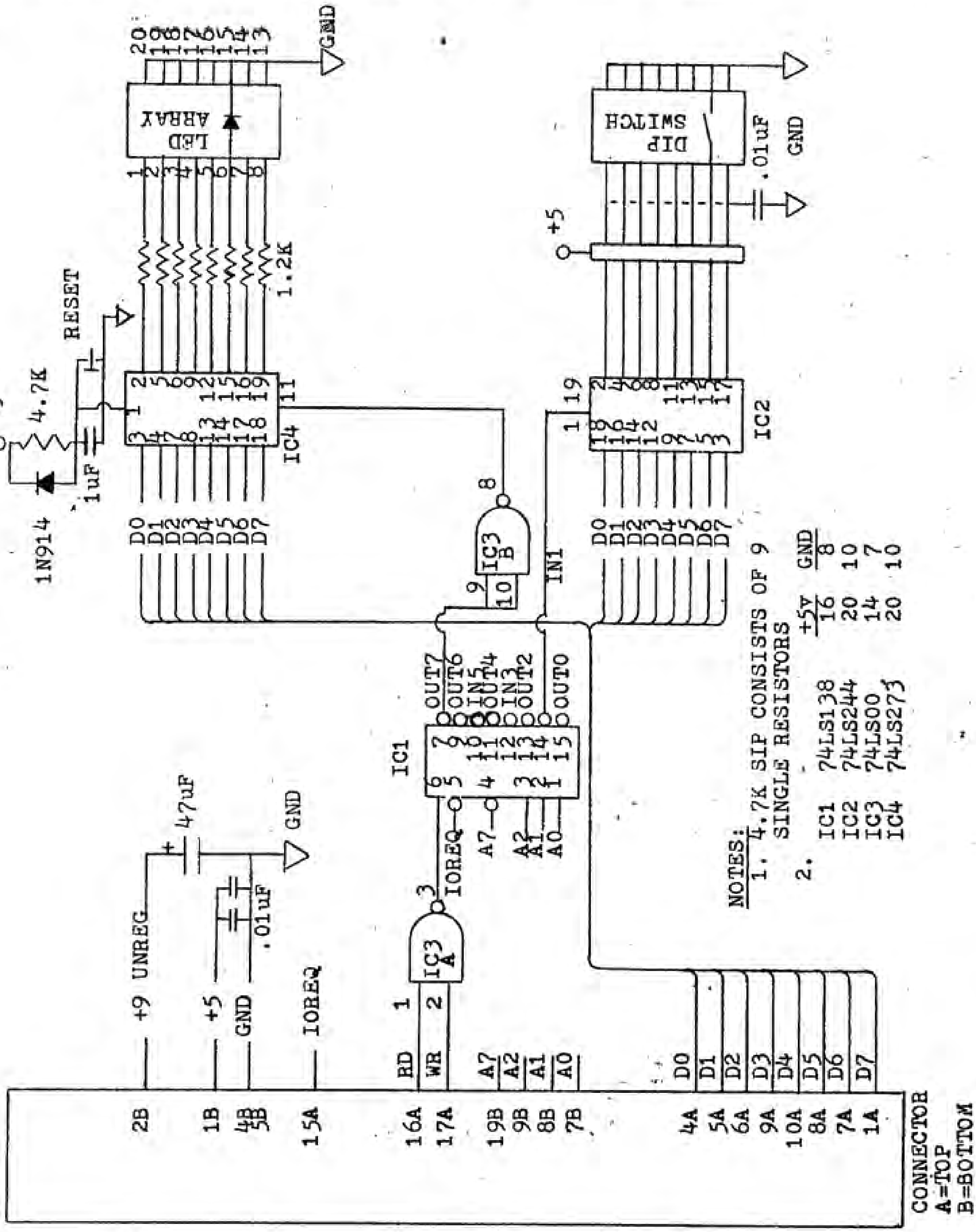
8. Power up. Your computer should act normally and the LEDs should not be lit.

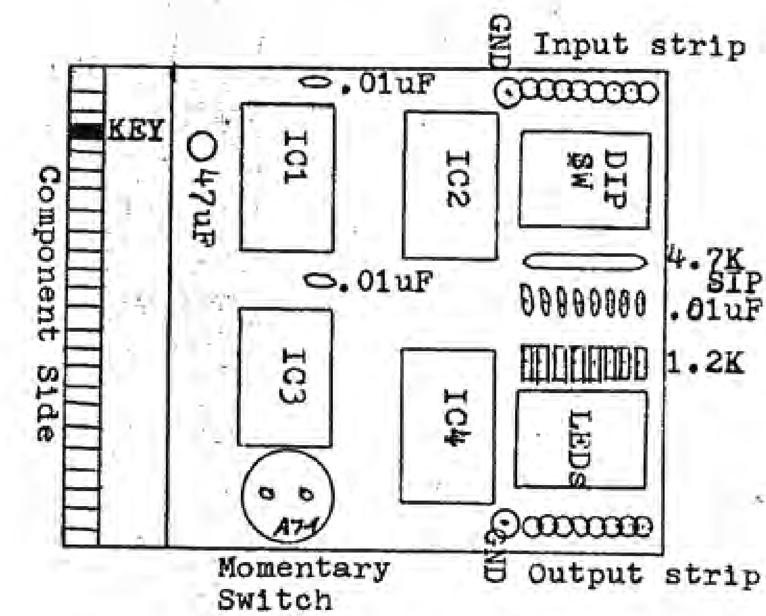
9. You are now ready for the software to get you going.

Operation Precautions

Always disconnect the power from your computer before either connecting or disconnecting the RX81. With the RX81

Figure 3-1. RX81 input/output board schematic diagram.





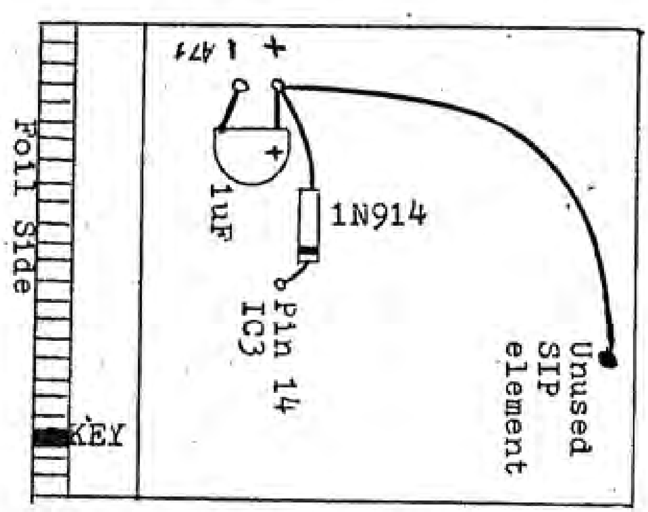


Figure 3-2. RX81 I/O board component layout.

Table 3-1
RX81 PARTS LIST

(1) RX81 bare board (Budget Robotics & Computing)
(1) 74LS273 (IC4)
(1) 74LS244 (IC2)
(1) 74LS138 (IC1)
(1) 74LS00 (IC3)
(1) Edge connector, 25/50 pin .1" x .2" (Digi-Key # C5-25)
(1) DIP switch (8 switches)
(1) Momentary pushbutton switch
(2) 16 pin IC sockets
(3) 20 pin IC sockets
(1) 14 pin socket
(8) 1.2K ohm 1/4 watt resistors
(1) 4.7K ohm reisitor network SIP, 9 element (Digi-Key # Q9472)
(1) 1N914 diode
(1) 1uF 10v capacitor
(1) 47uF 10v capacitor
(1) LED array (RS 276-081)
(10) .01uF Cer or Mica capacitors

connected, the KURSOR should appear as usual, on power up, then load the software.

Always protect your computer and peripherials from static sources found in carpet and on television screens.

Entering Machine Code

For RX81/TS1000/TS1500, first enter a "1 REM" statement containing at least 24 spaces. Then enter the rest of the program in Table 3-2. Now enter "RUN" and as each address location is displayed, enter the appropriate code from Table 3-3. Then LIST the program and check the 1 REM statement by comparing it with the listing in Figure 3-3. If a symbol is incorrect, just correct it by POKEing the address directly. If it is ok you can delete lines 10 through 50 and procede with the sample BASIC software. This machine code, now stored in the 1 REM statement, can be SAVEd along with the BASIC listing. Complete documentation for the machine code routine is contained in Table 3-4.

Table 3-2
MACHINE CODE ENTRY PROGRAM

```
 1 REM
10 FOR N = 16514 TO 16537
20 PRINT AT 10,10;N
30 INPUT I
40 POKE N,I
50 NEXT N
```

Table 3-3
CODE TABLE

ZX81/TS2068 address	Code	Address	Code
16514/65268	219	16526/65280	47
16515/65269	1	16527/65281	185
16516/65270	47	16528/65282	32
16517/65271	79	16529/65283	240
16518/65272	219	16530/65284	6
16519/65273	1	16531/65285	0
16520/65274	47	16532/65286	201
16521/65275	185	16533/65287	62
16522/65276	32	16534/65288	1
16523/65277	246	16535/65289	211
16524/65278	219	16536/65290	7
16525/65279	1	16537/65291	201

Figure 3-3. 1 REM statement with machine code characters.

Machine code for the TS2068

Begin all TS2068 control programs with the following three BASIC statements listed in Table 3-5 and they will automatically load the machine code into the computer when the program is RUN.

BASIC Demonstration Programs

Now complete the RX81 control software by entering the appropriate portions of Table 3-6. After entry is complete, RUN

Table 3-5
TS2068 MACHINE CODE SUBROUTINE

```
1 CLEAR 65267: FOR n = 65268 TO 65291
2 READ d: POKE n,d: NEXT n
3 DATA 219,1,47,79,219,1,47,185,32,246,219,1,47,185,32,240,
6,0,201,62,1,211,7,201
```

TABLE 3-6
INITIAL DEMONSTRATION PROGRAM

```
 5 SLOW                     (omit for TS2068)
10 FOR N = 0 TO 135
20 LET IN = USR 16514          (65268 for the TS2068)*
30 PRINT IN; "space";                                *
40 POKE 16534,IN               (65288 for the TS2068)*
50 LET OUT = USR 16533         (65287 for the TS2068)*
60 NEXT N
```

* See note in text

Table 3-4
RX81 MACHINE CODE DOCUMENTATION

ZX81/TS2068 address	Code	Hex	Z80 Assembler	
(IN routine)				
16514/65268	219	DB	in a,N	
16515/65269	1	01		(select I/O board) (i.e. IN 1)
16516/65270	47	2F	cpl	
16517/65271	79	4F	ld c,a	
16518/65272	219	DB	in a,N	
16519/65273	1	01		(select I/O board) (i.e. IN 1)
16520/65274	47	2F	cpl	
16521/65275	185	B9	CP c	
16522/65276	32	20	JR NZ,N	(diff)
16523/65277	246	F6		-10
16524/65278	219	DB	in a,N	
16525/65279	1	01		(select I/O board)
16526/65280	47	2F	cpl	
16527/65281	185	B9	CP c	
16528/65282	32	20	JR NZ,N	(diff)
16529/65283	240	F0		-16
16530/65284	6	06	ld, b,N	
16531/65285	0	00	nop	
16532/65286	201	C9	ret	
(OUT routine)				
16533/65287	62	3E	ld, a,N	
16534/65288	1	01		(output D0 selected) (i.e. 1=D0, 2=D1)
16535/65289	211	D3	out N,a	
16536/65290	7	07		(select I/O board wired as 7)
16537/65291	201	C9	ret	

To activate these input or output routines the following BASIC commands are given:

To read input, "LET IN = USR 16514" or "RAND USR 16514" Subsitute the address "65268" in a TS2068 program.

To activate an output, "LET OUT = USR 16533" or "RAND USR 16533" Again, substitute the address "65287" in a TS2068 program.

After executing the selected machine code routine, the computer will return to the next statement following the BASIC command that activated the machine code routine.

this program. The computer is reading your 8 switches (an 8 bit word) and displaying it on the screen, while at the same time lighting the corresponding LED. Note how changing the switch settings change the binary number. Understanding this simple example by experimenting with it is fundamental to the proper operation of this interface system. The following is for those who are new to binary numbers (base 2). Example: Turn on only the first and third light up from the bottom (marked binary 1 and 4). The solution is in Table 3-7.

So you "POKE 16534,5" and "LET OUT = USR 16533" and the two OUTs are turned on! Naturally to turn off all LEDs you would "POKE 16534,0" and "LET OUT = USR 16533". Or simply push the RESET button (Remember to convert tbe addresses for application to the TS2068). Now erase lines 10-60 and type in the listing from Table 3-8.

This program is a more user-friendly version. RUN it and you will easily see the conditions of your switches displayed on the screen.

To use this program as a subroutine, delete lines 460-480 and add "458 RETURN". Now, if you want to know the status of say the bottom switch (labelled 1) you just, "GOSUB 400", and the 1 or 0 status of switches 1 through 8 is stored in array S(1) through S(8), ready to be used or printed out. Switch 1 is stored in S(1). What this program is actually doing is reading the eight inputs in binary and dividing the result by decreasing powers of two to determine the orginial ones and zeros (opens and closeds).

The next demonstration program is a more user friendly version of the OUTPUT program. It also must be preceeded by the machine code line(s). If you have a 2K or better RAM machine you can use the previous listing along with the one in Table 3-9.

RUN this and the screen will ask you which LED to control. Enter a 3 and LED 3 will light up. Enter a -3 and the third LED will turn off. Enter a zero and all LEDs will turn off.

To use this program as a subroutine, delete lines 500-520, and line 560 and add line "558 RETURN". Now if you want to turn a particular LED (or external device) on or off, you just..... "LET X = (the LED number)", and, "GOSUB 500".

What this demonstration program is actually doing is taking your LED number and raising 2 to its power to determine the binary number to send to the LED latch. If a negative number is input, its binary number is subtracted from the last one to change the LED status. (Note: On the TS2068, do not use "IN" or "OUT" function keys. Instead, type "in" and "out" in lower case letters in program listings.)

Application Notes

Now that you have the ability to sense and respond, the computer can be put to work providing the brains behind a control application or machine automation.

Inputs can include everything from thermostats and limit switches to external keyboards and analog to digital converters. Outputs commonly driven are relays, motors, fans, alarms, digital to analog converters, etc.

Table 3-7
BINARY NUMBER EXAMPLE

Solution:

Light	Condition	Binary Number		
8	OFF	128		
7	OFF	64		
6	OFF	32		
5	OFF	16		
4	OFF	8		
3	ON	4 --	4	
2	OFF	2		Add
1	ON	1 --	1	
			5	

Table 3-8
"INPUT" DEMONSTRATION PROGRAM

```
400 REM "INPUT"
405 DIM S(8)
410 LET IN = USR 16514         (65268 for the TS2068)*
415 FOR N = 8 TO 1 STEP -1
420 LET R = IN -2**(N-1)       (** is symbol shift H on the
425 IF R = 0 THEN GOTO 455      2068)*
430 IF R > 0 THEN GOSUB 440
435 NEXT N
438 GOTO 460
440 LET S(N) = 1
445 LET IN = R
450 RETURN
455 LET S(N) = 1
460 FOR B = 1 TO 8
465 LET C$ = "OPEN"
470 IF S(B) = 1 THEN LET C$ = "CLOSED"
475 PRINT B; "space";C$
480 NEXT B
```

* See note in text

Table 3-9
"OUTPUT" DEMONSTRATION PROGRAM

```
500 REM "OUTPUT"
505 LET M = 0
510 PRINT "LED?"
515 INPUT X
520 PRINT X
525 IF X = 0 THEN LET M = 0
530 IF X = 0 THEN GOTO 550
535 LET N = 2**(ABS X-1)
540 IF SGN X = -1 THEN LET M = M-N
545 IF SGN X = 1 THEN LET M = M+N
550 POKE 16534,M               (65288 for the TS2068)
555 LET OUT = USR 16533        (65287 for the TS2068)*
560 GOTO 510
```

* See note in text

Expansion

The RX81 comes wired to control an LED array with an OUT 7 (see Figure 3-1) but also provides for other OUTs if rewired. Extra boards can be used to select additional outputs if desired. The data to be sent out is POKEd into 16534 (65288 for TS2068), the OUT needed is POKEd into 16536 (65290 for the TS2068), and then USR 16533 (65287) is run. In other words, if boards wired for OUT 6 and OUT 7 are both connected, you select the proper board by POKEing 6 or 7 into 16536 (65290).

Similarly your board reads a DIP switch with an IN 1, but also provides decoding for other INs if rewired.

Driving External Logic

To connect the RX81 to an external board for prototyping purposes, pry the LED array off its socket. Replace it with a 16 pin DIP plug with ribbon cable with the DIP plug inserted into the socket. Pins 1 through 8 are your outputs and 9 through 16 are logic ground. The .01uF suppression capacitors have been added to all outputs. They may be removed if you are feeding the signals to other logic gates. Recommended fanout is one LS load per gate.

Specific examples of input and output devices driven and/or read by the RX81 are found in Chapters 5, 6, 8 and 11.

8255 PPI INPUT/OUTPUT CIRCUIT

As the name implies, the 8522 Programmable Peripherial Interface (PPI) chip is programmable via commands from the computer. These simple POKE commands, which will be fully explained later, turn up to 24 control lines on/off and designate whether a line is ready for input or output of data. The basic difference between this device and the RX81 in the application described is that the 8255 is commanded via BASIC program commands instead of machine code. This means that the 8255 will not be as fast an I/O device as the RX81 but nonetheless it has some advantages of its own. The primary advantage is the large number of I/O lines easily programmable on and off. As you will see in later applications, the speed of machine code I/O is not always required. A good example of the requirement for machine code I/O speed is illustrated in Chapter 8.

Although the 8255 circuit described can be plugged directly into the computer, the application that follows will be constructed on a board that plugs into the Expansion Board. Figure 3-4 depicts the entire circuit, with Figure 3-5 indicating the layout of components. A full parts list is provided in Table 3-10.

Component Layout

IC 4 is located in the center and ICs 1 through 3 are lined up on the right side in Figure 3-5. The capacitors should be installed between +5v and ground on at least every other IC to suppress any voltage spikes. The two extra ICs (7402 and 74LS08) located at the top of the board are part of an output circuit used for two applications described in Chapters 7 & 9. The two empty IC sockets are for DIP jumpers which are also required for these applications. The toggle switch is optional and can be installed between the +5 volt input and supply to the components

Figure 3-4. 8255 input/output port schematic.

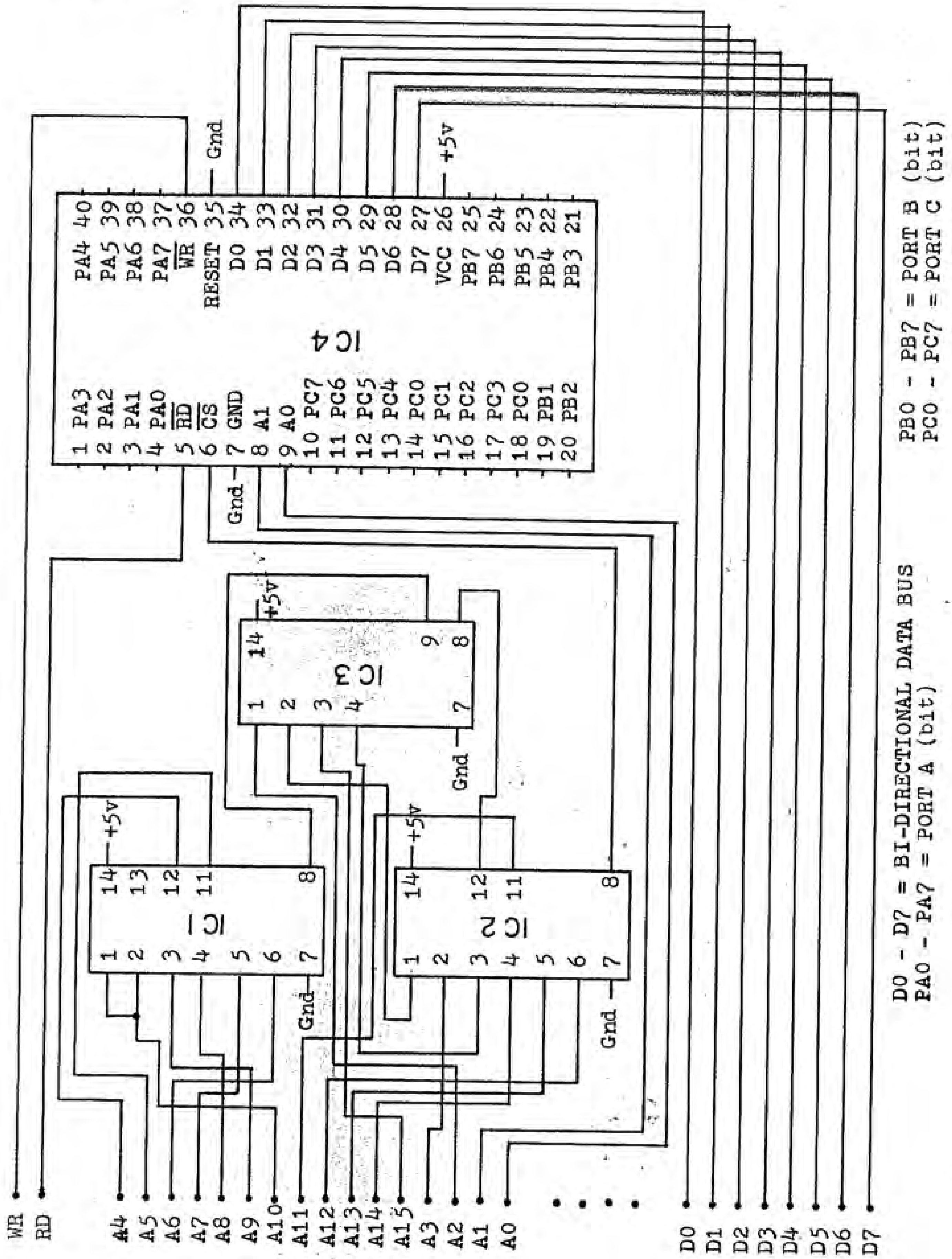


Table 3-10
8255 PARTS LIST

(1) Circuit board (Radio Shack 276-152)
(6) 14 pin IC sockets
(1) 16 pin IC socket
(1) 40 pin IC socket
(1) 8255 PPI (IC4)
(2) 7430 (ICs 1 & 2)
(1) 74LS04 (IC3)
(1) 7402
(1) 74LS08
(4) 0.1uF capacitors
(1) on/off toggle switch
Hookup wire

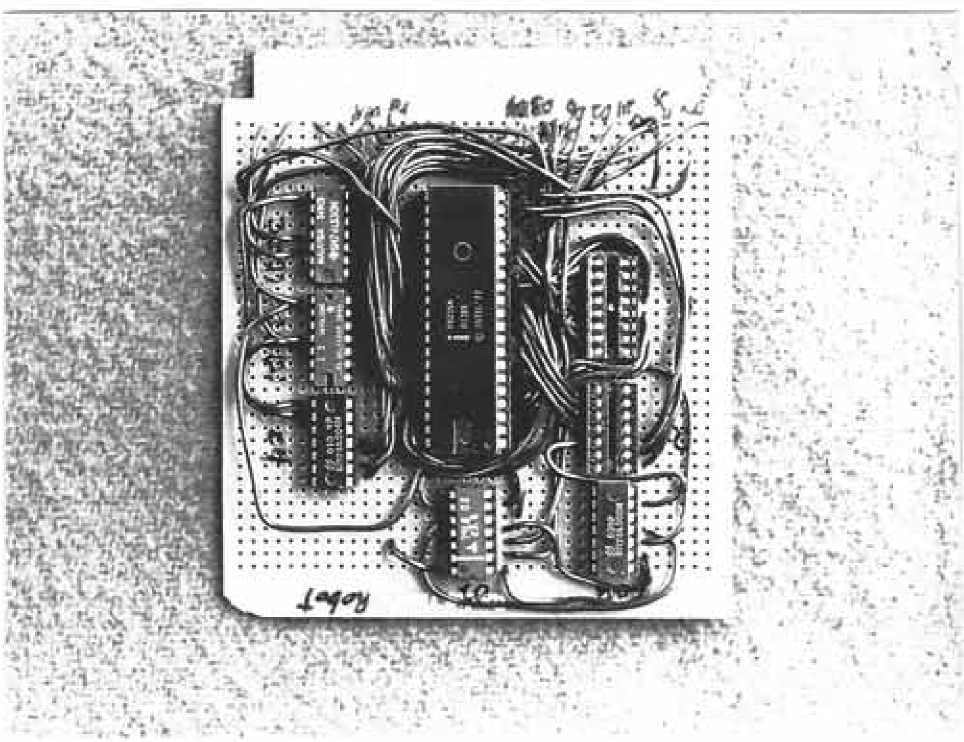


Figure 3-5. 8255 circuit board layout.

on the board. The purpose is to allow the circuit to be deactivated without having to remove the board from the Expansion Board. The utility of this option will become more obvious after an explanation of the address decoding that follows. First solder all IC sockets in place. Then install all other components as indicated with hookup wires as required from board fingers to IC socket points. Figure 3-6 shows both the 8255 and RX81 boards plugged into the Expansion Board. The same operations precautions apply as noted for the RX81. Always disconnect the power from your computer before either connecting or disconnecting any peripherial.

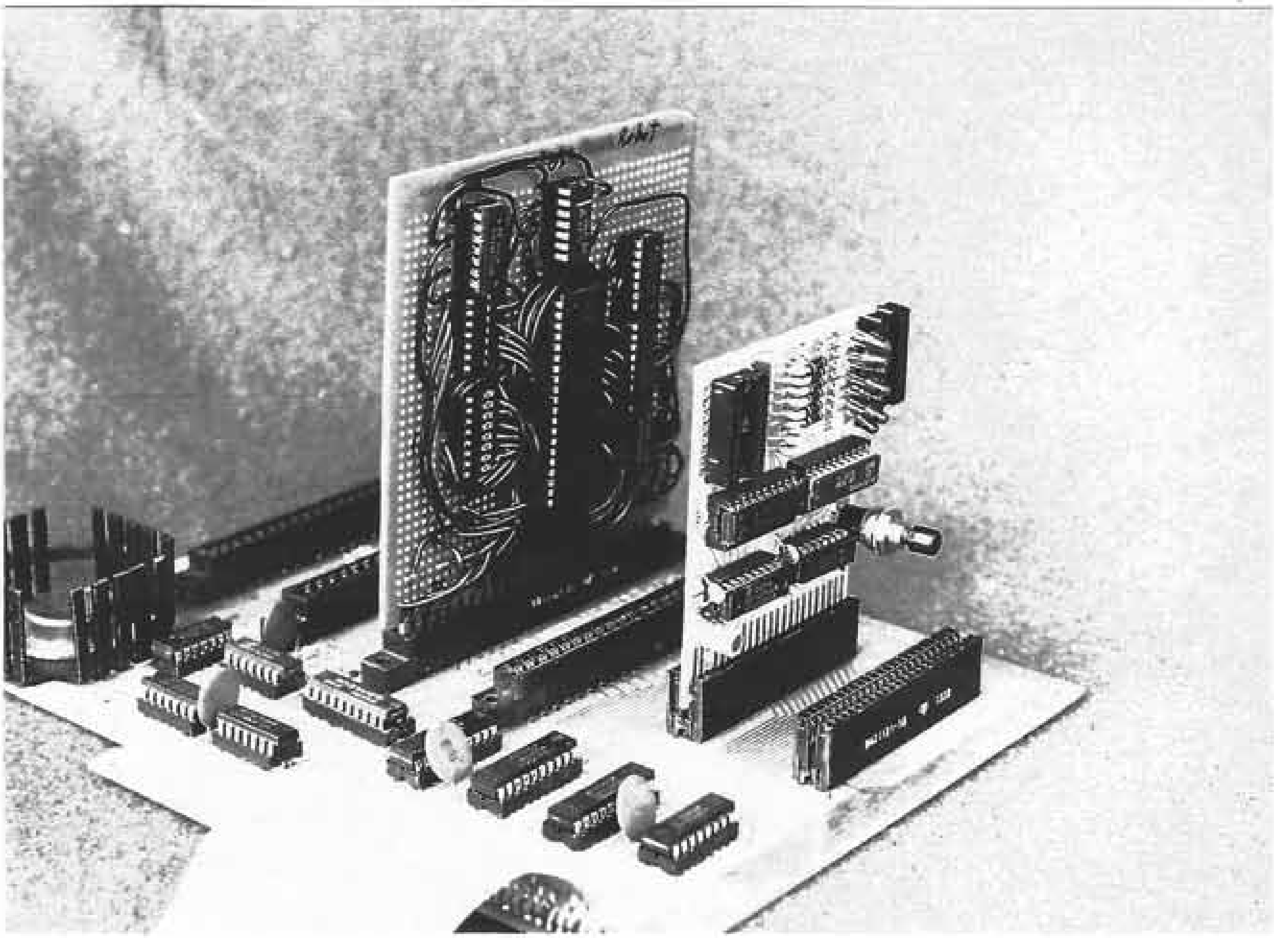
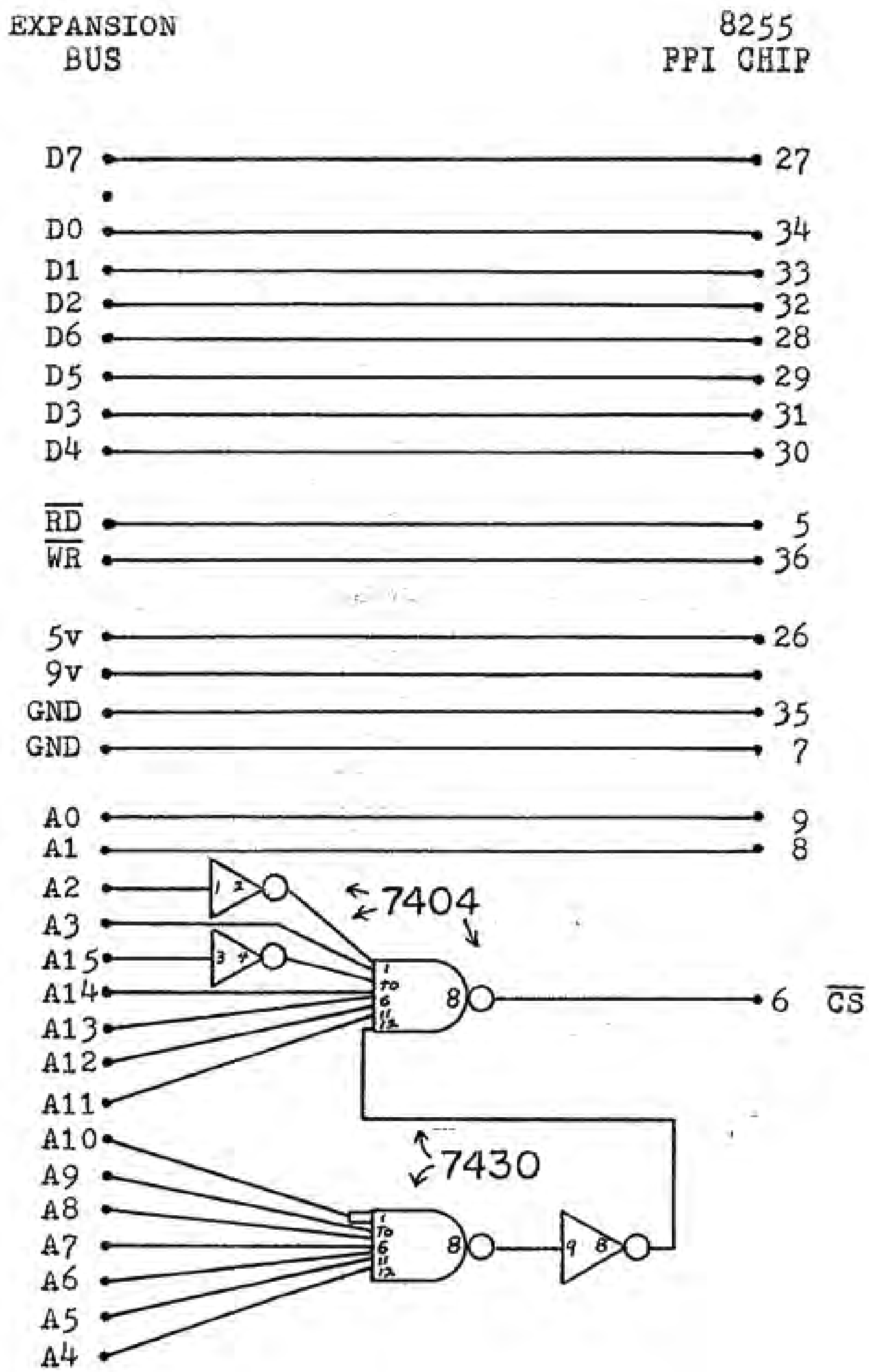


Figure 3-6. RX81 & 8255 boards plugged into Expansion Board.

Address Decoding

Figure 3-7 depicts the address decoding performed by ICs 1 through 3. With this decoding addresses 32760 through 32763 give you complete control of the 8255 chip. Table 3-11 lists all addresses and their function. Because these addresses fall within the useable RAM of your computer (true only if you are using a 16K RAM pack) the RAMTOP must be reset before this circuit can be properly addressed. A full explanation of this requirement is expained in your computer User Manual under "RAMTOP". With a 16K RAM pack installed, the RAMTOP will be automatically set to address 32767. In order to lower the RAMTOP



(pin 7 GND & pin 14 +5v for 7404 & 7430)

Figure 3-7. Address decoding to drive 8255 PPI I/O circuit.

Table 3-11
8255 Addressing

Port	POKE
Port A	32760,(decimal)
Port B	32761,(decimal)
Port C	32762,(decimal)
Control	32763,(decimal)

Control

Bit				1 = Decimal
7	1			128
6	always 0			64
5	always 0			32
4	Port A	0 = output	1 = input	16
3	Port C, upper	0 = output	1 = input	8
2	always 0			4
1	Port B	0 = output	1 = input	2
0	Port C, lower	0 = output	1 = input	1

Examples: To set up all 24 port data lines for output "POKE 32763,128"

To output low (0) to Port A, for all 8 data lines "POKE 32760,0"

To output high (1) to Port A, for all 8 data lines "POKE 32760,255"

below your 8255 addresses, enter the following after you turn on your computer and before you load any programs:

```
POKE 16388,255
POKE 16389,123
NEW
```

The new RAMTOP will now be set at 31743, which is well below the addresses of concern. To check the new RAMTOP setting you can enter the following statement into your computer:

```
PRINT PEEK 16388+256*PEEK 16389
```

If your earlier entry was correct, the answer printed on the screen should be "31743".

Software Control

After giving the proper programming instruction to the 8255 (at address 32763), ports A, B or C can be POKEd (for output) or PEEKed (for input) for the appropriate data line. Remember what you learned about decimal addressing in the previous section explaining the RX81 operation because the same techniques apply for the 8255. Table 3-12 summarizes these commands. For example, to turn port B, data line 7 on, you would POKE 32761,128. Remember that port B must first be turned on for output by POKEing the control address at 32763. To program all port lines for input you would POKE 32763,155. To program all port lines for output you would POKE 32763,128. Study Table 3-11 and you can see that any combination of port input or output can be programmed.

Table 3-12
8255 I/O Addressing

A or B Data Line	POKEd/PEEKed Decimal Address	C Data Line	Value POKEd/PEEKed Low (0)	High (1)
0	1		0	1
1	2		2	3
2	4		4	5
3	8		6	7
4	16		8	9
5	32		10	11
6	64		12	13
7	128		14	15

Note that port C is addressed differently than A & B and may require some study to completely understand. The literature explaining this port address procedure leaves something to be desired. I can only ask that you study Tables 3-12 and 3-13 closely to try to gain an understanding of port C addressing.

Table 3-13
Port C Programming

Output C	POKE 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H
1	L	L	H	H	L	L	H	H	L	L	H	H	L	L	H	H	L	L
2	L	L	L	L	H	H	H	H	L	L	L	L	H	H	H	H	L	L
3	L	L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	L	L
4	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H	H
5																		
6	(and so forth, extending the pattern)																	
7																		

The obvious difference in programming port C is that the upper (lines PC4 through PC7) and lower (lines PC0 through PC3) halves can be programmed independently.

Applications

As already mentioned, specific applications for this I/O circuit are detailed in later chapters. Chapter 7 uses the 8255 PPI circuit to drive a stepper motor. Chapter 9 uses the 8255 PPI to drive the digital voice circuit using the National Semiconductor, Digitalker, integrated circuit chips. The hardware and software described can operate both applications at the same time, in addition to the RX81 I/O circuit. Table 3-14 summarizes commands and pinouts for all peripherals described in later chapters.

Table 3-14
8255 PORT LOCATIONS/PINOUTS

8255 pin	Port Data	POKE Value (output high) (0 = low)	16 pin socket (DIP jumper 1) pin number	14 pin socket (DIP jumper 2) pin number
4	A0	1	1	
3	A1	2	2	
2	A2	4	3	
1	A3	8	4	
40	A4	16	5	
39	A5	32	6	
38	A6	64	7	
37	A7	128	8	
18	B0	1		1
19	B1	2		2
20	B2	4		3
21	B3	8	(not connected)	
22	B4	16	(not connected)	
23	B5	32	(not connected)	
24	B6	64	(not connected)	
25	B7	128	(not connected)	
14	C0	1	9	
15	C1	2	10	
16	C2	4	11	
17	C3	8	12	
13	C4	16		8
12	C5	32		9
11	C6	64		10
10	C7	128		11

NON-8255 PORT FUNCTIONS	16 pin socket	14 pin socket
WR for SPC from 7408 pin 3	13	
+9 volts	14	
Ground	15	
+5 volts	16	
Pulse from 7408 pin 6		12
Ground		13
+5 volts		14

THIS PAGE INTENTIONALLY BLANK

Chapter 4
ROBOTICS ON A BUDGET - CONSTRUCTION

In this chapter I will describe, in some detail, the physical and mechanical design and construction of the robot called H.E.N.R.Y. Chapter 5 will then outline the electrical and computer control details of the robot.

The inspiration for the design of H.E.N.R.Y. came from the Radio-Electronics (RE) magazine reprint series of articles titled, "Build this robot for under $400". Although H.E.N.R.Y. looks similar to the RE Unicorn-1 robot, there are many differences in the design and construction. As complete as the series of articles was, there were many errors and incomplete explanations of the design. Unfortunately, the RE editors were not interested in either acknowledging the errors or printing corrections.

The robot retains the basic "garbage can" design but the diameter of the body was reduced to 16 inches. It was obvious that a wider body, with arms added, would have trouble navigating through doorways and other tight household spots. The drive wheels were placed in the front to pull the robot around and aid in traction. This design feature placed the battery to the rear and aided stability. The base was made round and the same diameter as the body for aesthetic reasons. Although small access doors were located in both the front and back of the base, the required access to the base was achieved by making the entire base plate of the upper body, with turntable, hinge backward to allow full access to the inside of the base.

All illustrations in this chapter will be scale drawings from the robot which I designed and built. Photos of various views of the robot can be found in the next chapter.

The key to inexpensive robotics construction is to adapt materials at hand or available as scraps or surplus. Although I will describe the materials I used, I will also make other suggestions where I am aware substitutes will work. I will also provide cautions against substitute materials and methods which will not work well, and tell you why they are not a good idea to use.

THE ROBOT BASE

A depiction of the entire robot body (minus the arms) can be seen in Figure 4-1. This drawing provides the overall dimensions of the robot and should be referred to throughout the construction. Figure 4-2 provides a view of the base from above with the upper base plate removed, at the top of the drawing.

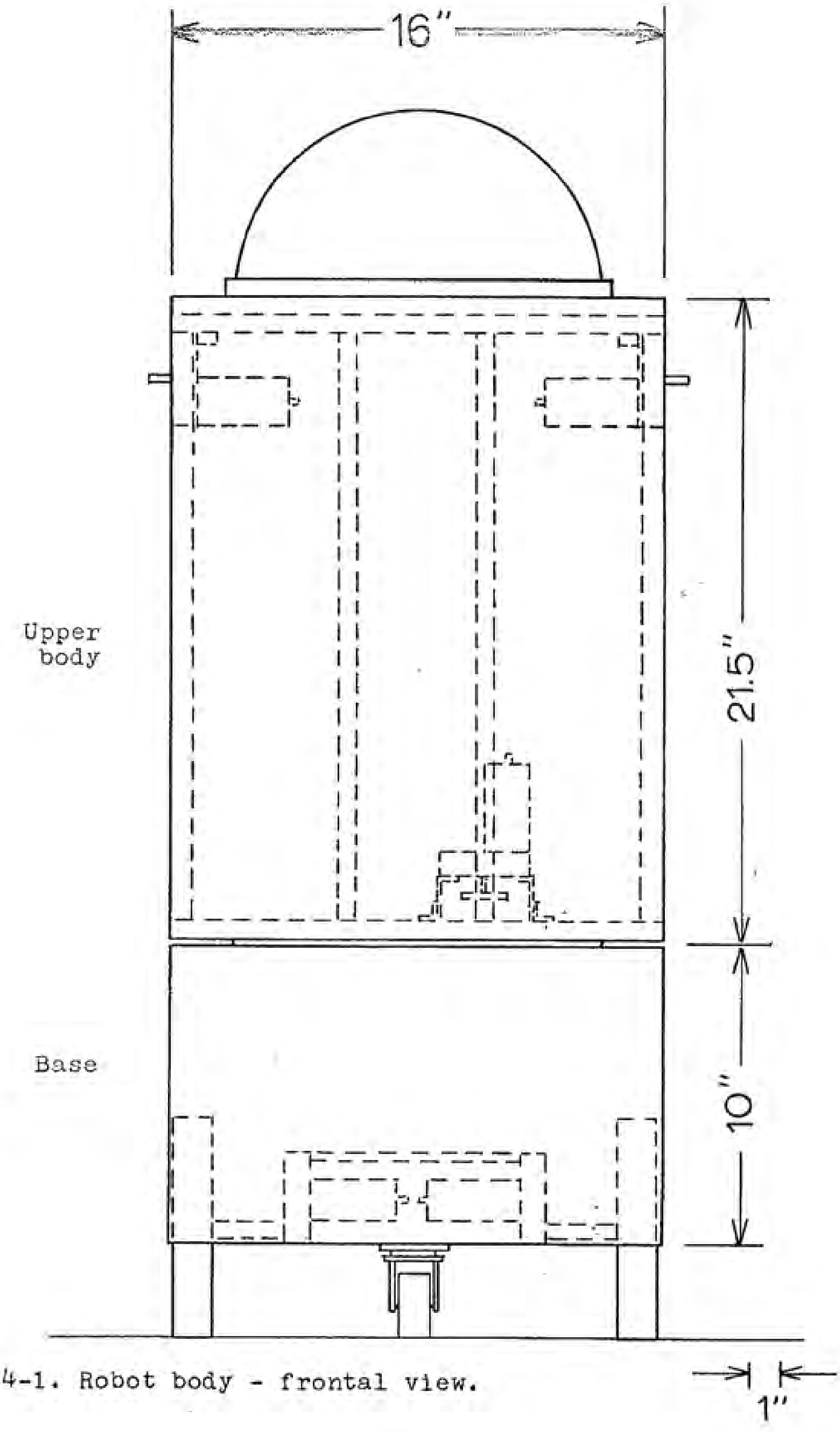


Figure 4-1. Robot body - frontal view.

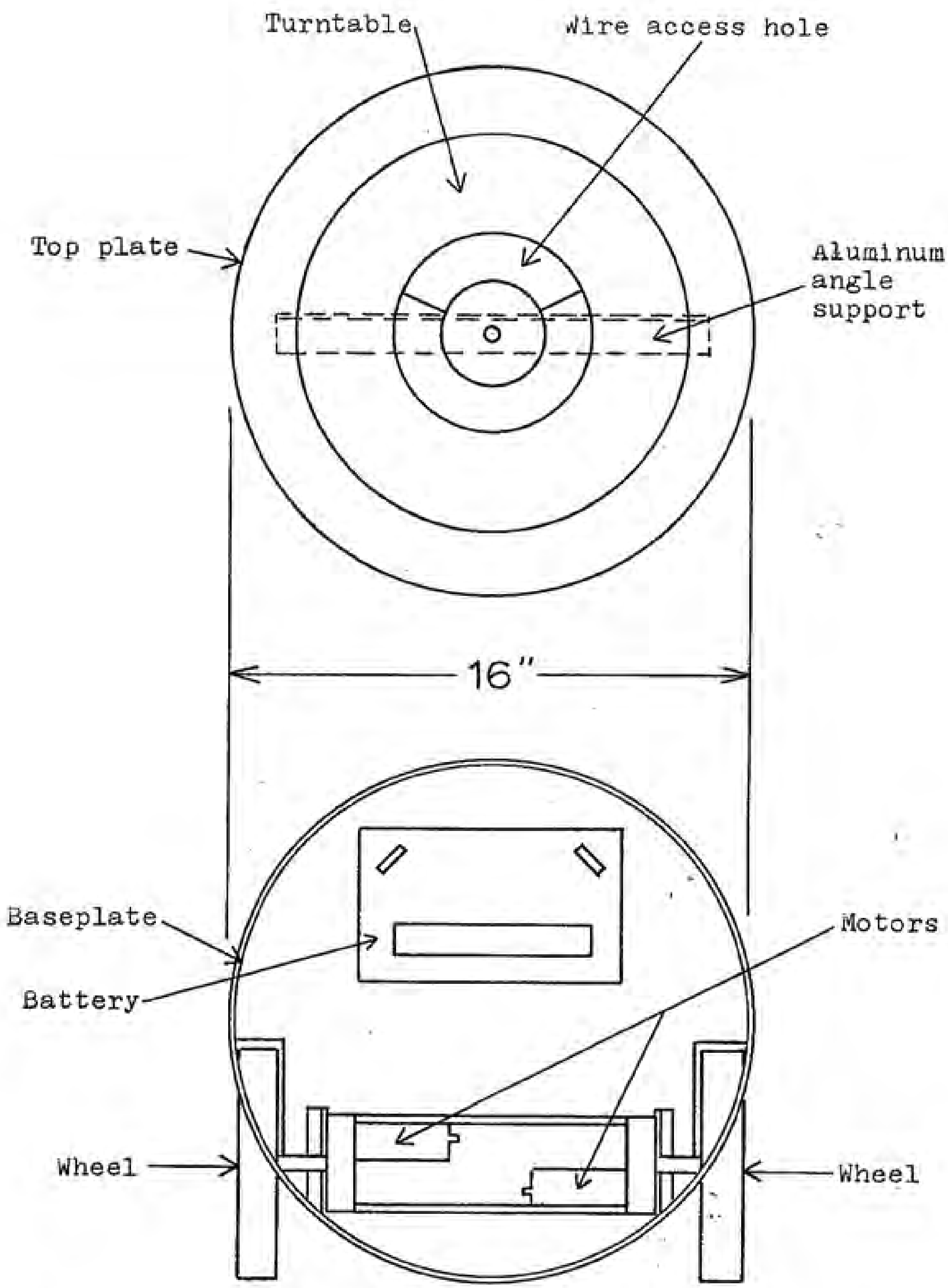


Figure 4-2. Robot base - view from above.

The base top, bottom and sides were made from scrap aluminum sheet which is 3/32 inch thick. The top and bottom plates are both circles, 16 inches in diameter. The top and bottom should be the first pieces cut out to size without cutting the wheel slots, to be cut out later. Use a power saber saw with metal cutting blade for all metal cutting. The sides consist of two pieces as depicted in Figure 4-3. When cutting the side pieces out, do not cut out the access door holes or wheel holes. Metal forming and assembly will be easier if the these holes are cut out after the fitting and assembly is complete. Exact wheel hole size and location will also depend on the wheel size you use and the placement of the drive motors.

The forming or bending of the sides will be easier if you cut a 16 inch diameter circle from a piece of 5/8 inch chip board. This piece will be used later in construction of the upper body. Simply place the chip board circle firmly in a vise and form each aluminum side piece around the circle. Don't worry if the side pieces don't form perfect 8 inch radius circles as this will be taken care of in the final assembly. In fact, the bent pieces should form half circles with a radius slightly larger than 8 inches.

For final assembly you will need to cut 20 aluminum angle supports/fasteners from a long piece of aluminum angle stock as shown in Figure 4-4. More of these fasteners will be used throughout the project. A 5/32 inch hole should be drilled in the center of each side of the fasteners for the 8-32 by 1/2 inch bolts which will be used. These same sized bolts in varying lengths will be used throughout the project. Although available from hardware stores, in quantity they are available at good prices from Digi-Key by mail order.

To assemble, bolt the first angle support to the bottom plate with the remaining drilled face flush with the outside edge of the bottom plate. Then, starting at the front of one of the side pieces, drill and bolt it to the fastener, with the bottom edge of the side piece flush with the bottom side of the bottom plate. Evenly space four more aluminum angle supports along the connecting edges. Repeat the process for the other side, starting at either the front or back, evenly spacing five angle supports and bolting them in place. Now, repeat the entire process fastening the top plate to the sides. However, the bottom plates must be removed first, to allow access to mark the holes, drill and fasten the bolts.

After top and bottom are fitted, the front and rear access door holes can be marked and cut out. Choose your own sizes for these doors but make them large enough to at least get your hand inside.

Next, reassemble the bottom plate to the sides and leave the top plate off for installation of the drive motors and wheels. You may want to install four lengths of aluminum angle on the bottom plate to hold the 12 volt battery at this point. See the next chapter for battery selection.

Drive Motors

I used the Brevel gear motor which is available from Gledhill Electronics or H&R Corporation (see parts list, Table 4-1). It is a relatively inexpensive motor but is very rugged

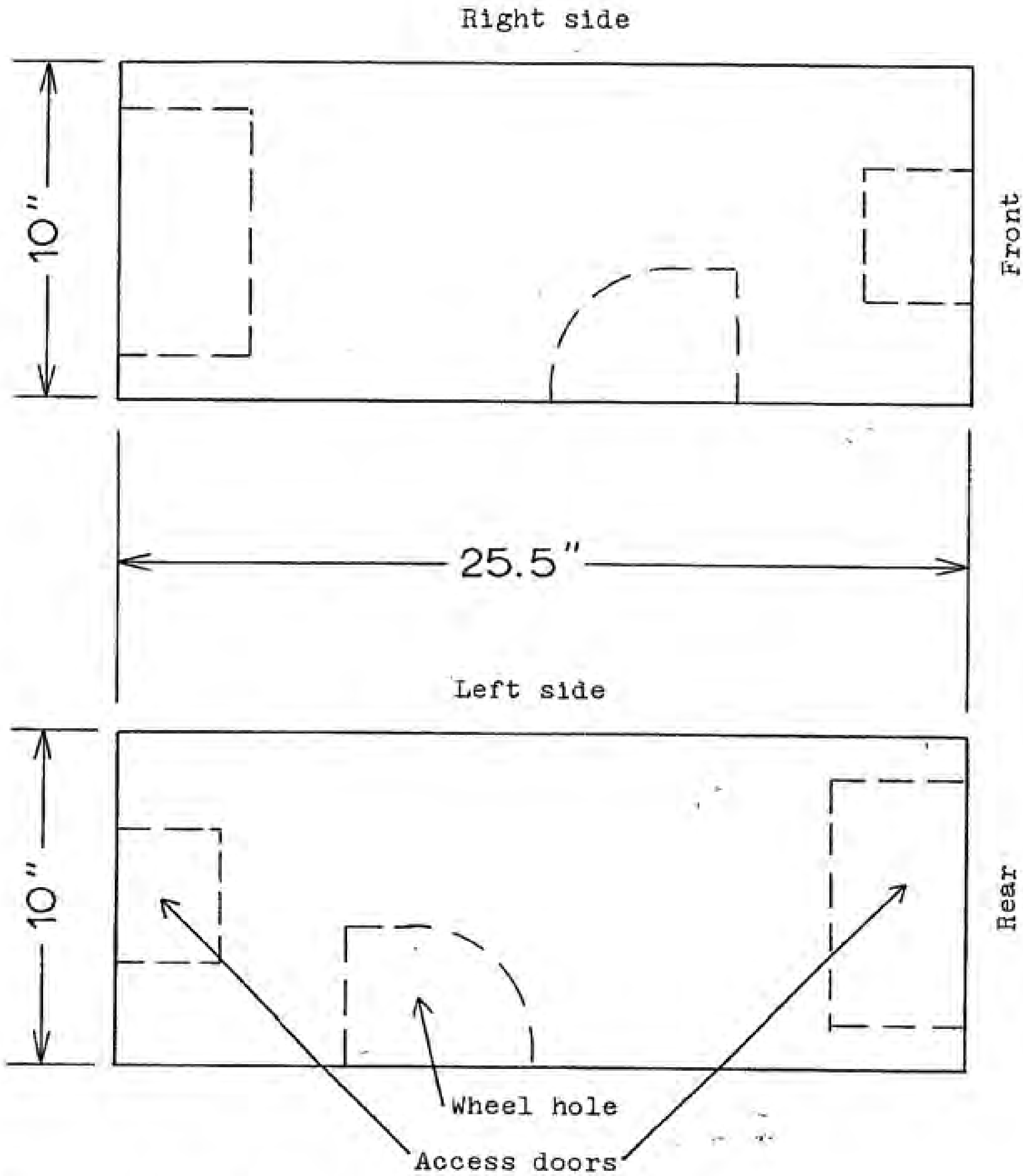


Figure 4-3. Robot base - sides.

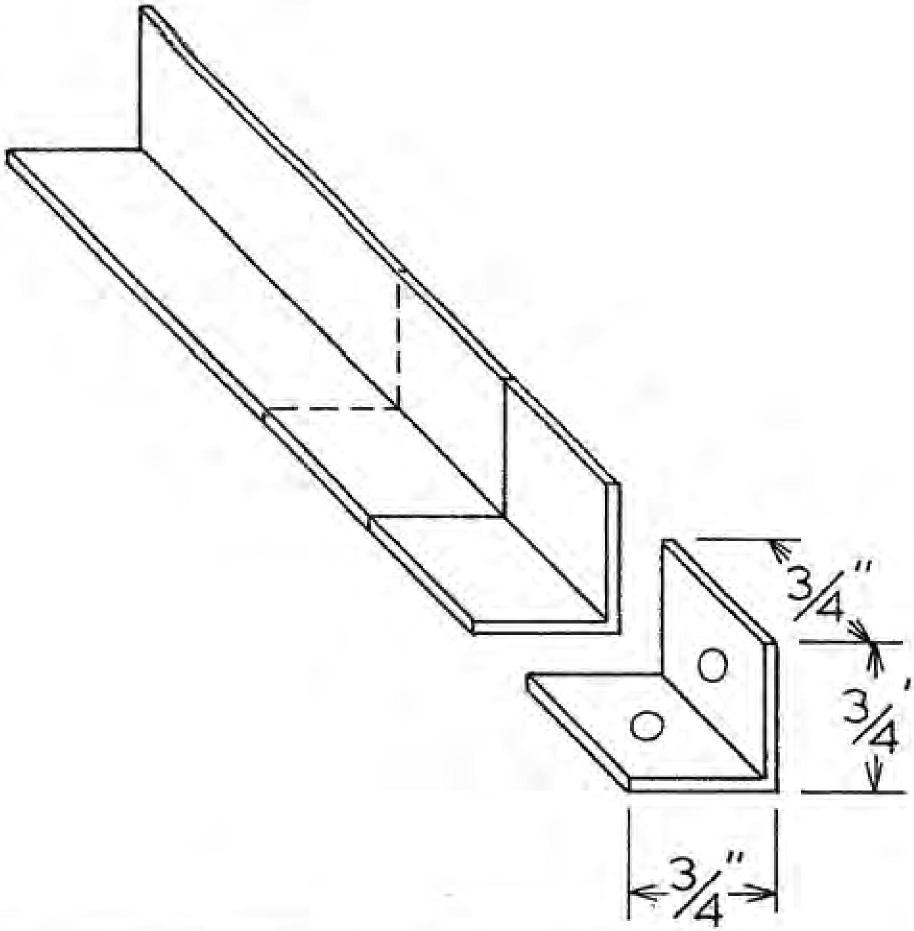


Figure 4-4. Aluminum angle supports/fasteners.

and capable. In fact, there are five of these motors in the robot with one rotating the upper body and one for each shoulder.

The two drive motors can be mounted back to back as a single unit as shown in Figure 4-5. The four 8-32 tap thru holes at each corner of the gear housing are used in the following manner. Two 3/8 inch square by 7 inch long aluminum pieces are end tapped with 8-32 threads. These are fastened to the top edges of the motor gear casings with 8-32 bolts as shown in Figure 4-5. Then two aluminum angles, each 4 inches long, are drilled and bolted to the bottom edges of each gear housing.

This motor assembly is placed as far forward on the base plate as possible so that the front edges of the wheels will be even with the front edge of the base. Drill the angle pieces and bolt them to the base plate. This placement will provide maximum stability.

The exact sizes of the axle couplers will depend on the size of the wheels/axles used. I used seven inch diameter discarded lawn mower wheels. The maximum distance to the outer edges of the wheels should be 15 3/4 inches so that the assembly fits within the base dimensions. After the wheels are sized, the wheel holes can be cut out of the base sides and bottom plate. A look at Figure 5-4 in the next chapter will give you an idea how this should all look when completed.

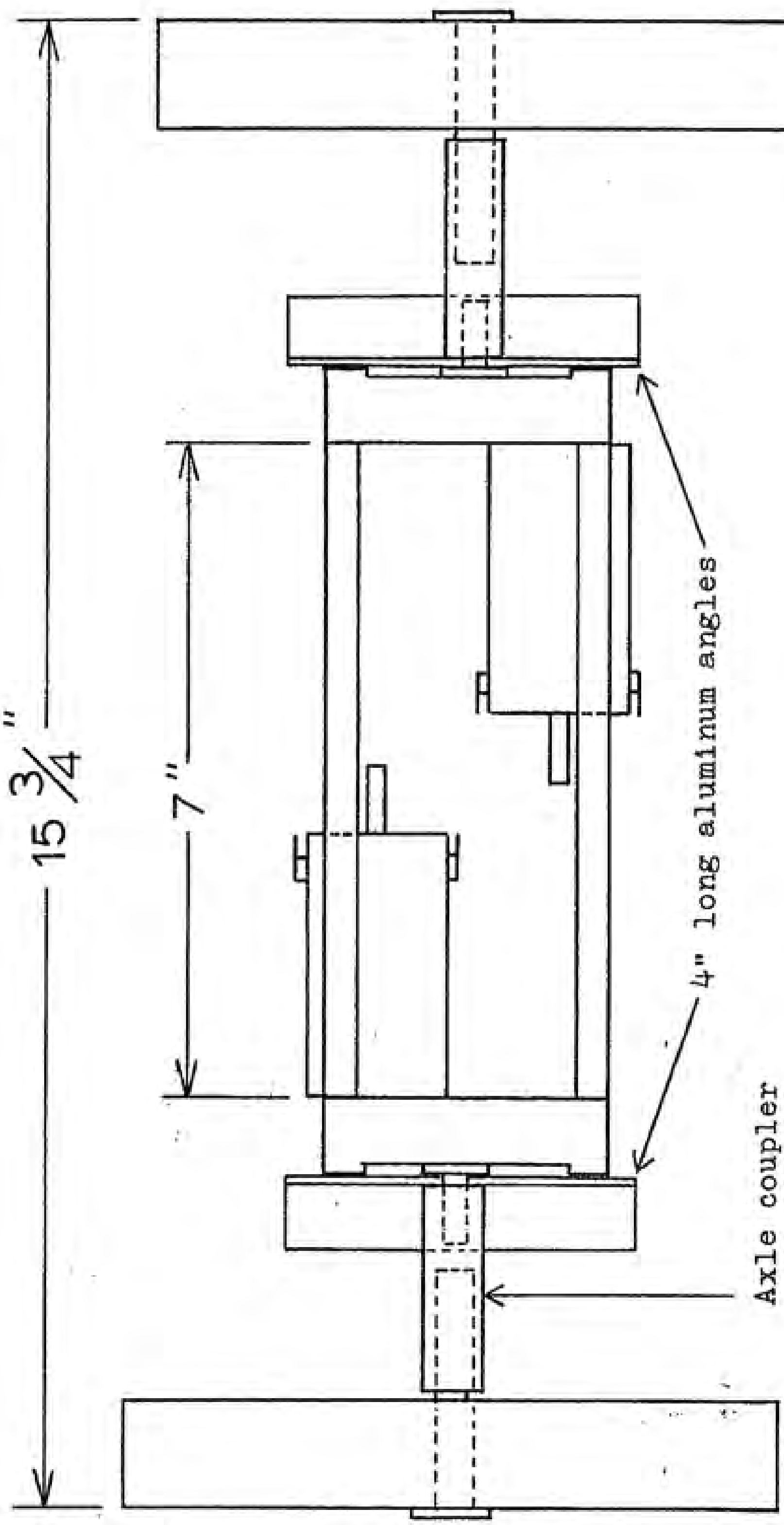


Figure 4-5. Aluminum angle supports/fasteners.

Base Top Plate

Referring back to Figure 4-2 you can now cut the wire access hole in the base top plate. This drawing and Figure 4-7 show the hole as an arc formed by a 1 1/2 inch and a 3 inch radius circle.

Next cut a 14 inch long 1 1/2 inch aluminum angle for bottom support of the base top plate. Bolt it to the underside of the top plate so as not to interfere with the turntable. The 12 inch metal turntable with ball bearings is available at most hardware stores and is also available from Edmund Scientific via mail order. The turntable is bolted to the topside of the base top plate.

Although not detailed in the drawings, you may want to fasten the top plate to the base sides, at the rear, with a normal room door hinge. The installation and its access advantages can be seen in the Chapter 5 photos. If you select this option the only other modification required is to replace the nuts on the bolts which fasten the base sides to the base top plate supports with "blind" nuts. Use 3/16 inch blind nuts which are available at hardware stores, and replace the bolts with 3/16 x 1 inch round head bolts.

Finally, don't forget to select and install the rear caster wheel. Select one based on the distance that the bottom of the base is above the floor. This will depend on the diameter of the drive wheels. I used a 2 inch diameter caster wheel. Mount it as far to the rear as possible for maximum stability. I found that a hard plastic castor wheel is best on carpeting and a hard rubber wheel is best on hard surfaces such as tile, wood or cement.

UPPER BODY

Refer to Figures 4-1 and 4-6 for construction of the upper body. The eight upright (vertical) supports are wooden strips, four 1/2 by 3/4 inch and four 3/4 inch square, all 19 1/2 inches long. The bottom and top pieces along with the lid are all 16 inches in diameter cut from 5/8 inch chip board. The 16 inch circle you used to form the base sides can be used for one of these pieces. Cut the holes in each as shown in Figure 4-6. You may want to wait to cut the hole in the lid for the wires connecting the computer with the expansion board until you decide on the exact placement of these components.

Although you can fasten the uprights to the top and bottom pieces with wood screws, I don't recommend it. Instead, for maximum strength, use one of the 3/4 aluminum angle supports (Figure 4-4) bolted to each end of each upright. That means you will need 16 angles cut and drilled. Again use 8-32 bolts of the appropriate lengths.

The upper body lid (Figure 4-6) should be fastened to the upper body top with a recessed cabinet hinge. The lid is hinged forward, as shown in Figure 5-6 (Chapter 5) using a piece of nylon cord fastened to screw eyes at each end to hold it in place while opened. The clear plastic bubble on the top of the lid is a 12 inch diameter terrarium cover, available from Edmund Scientific.

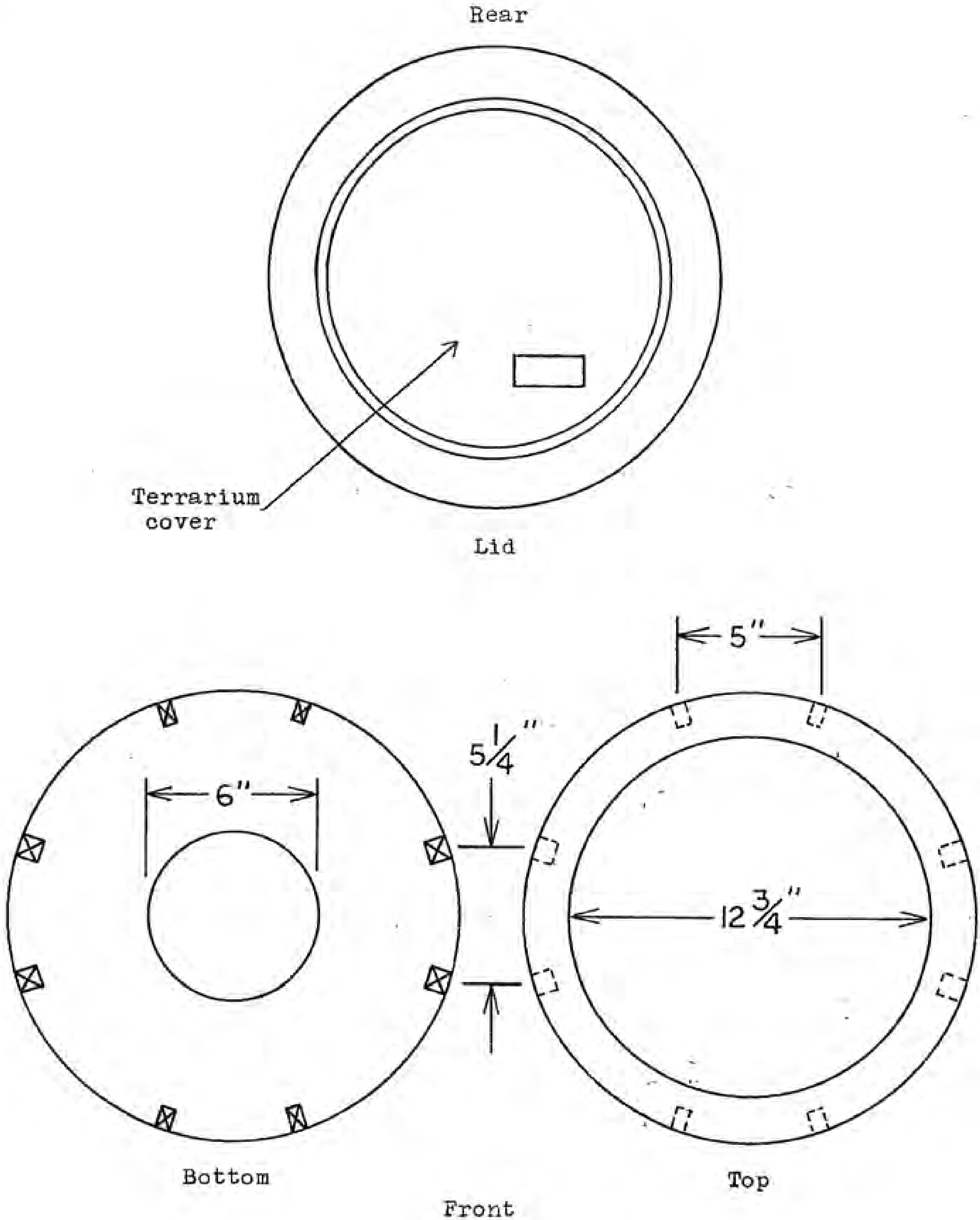


Figure 4-6. Upper body - view from above.

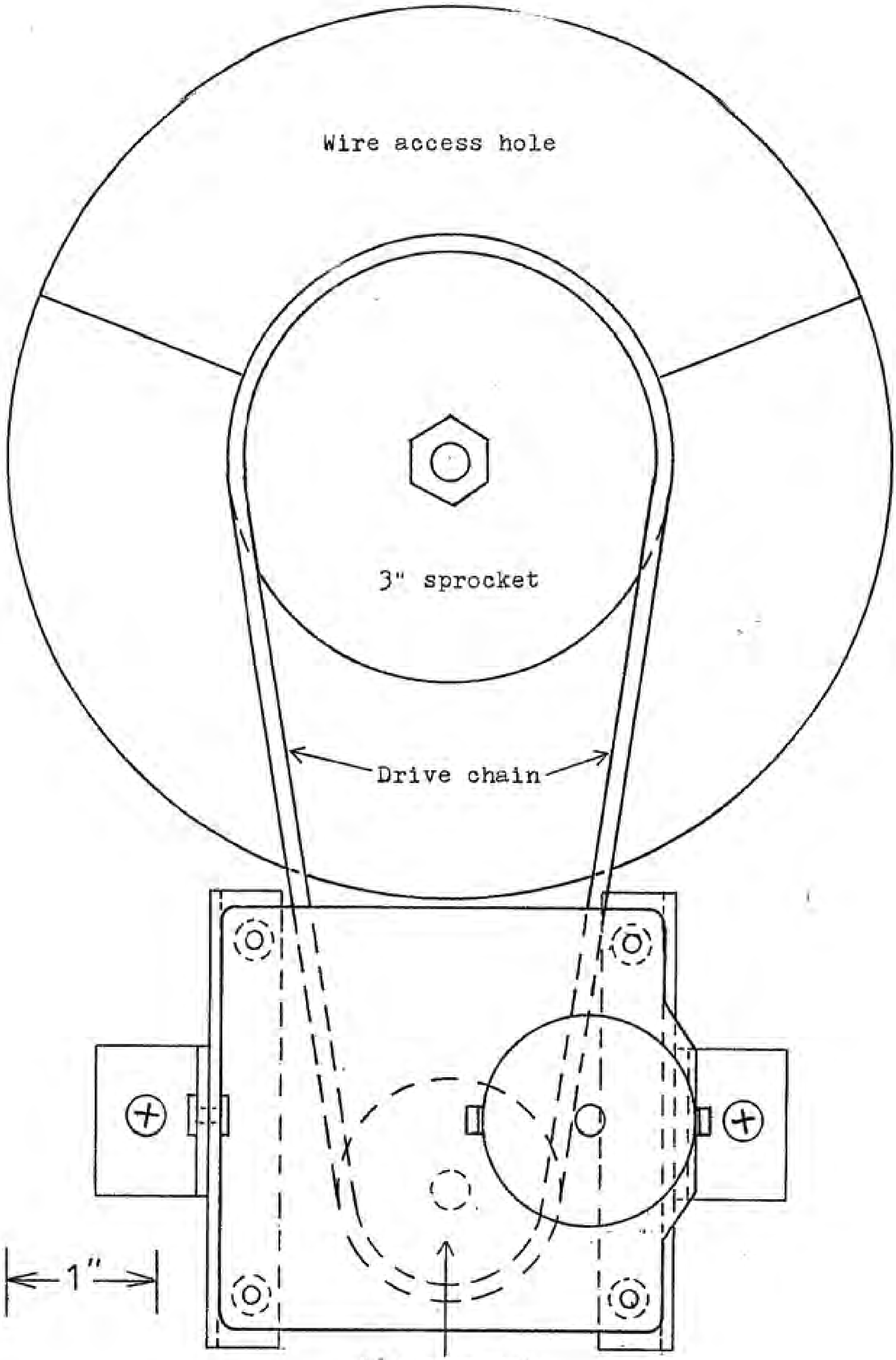


Figure 4-7. Upper body rotation mechanism.

Upper Body Rotation Motor

Before putting the skin on the upper body, the rotation motor should be installed. Again a Brevel motor is used but with sprockets and a chain drive. The motor is mounted with two sets of aluminum angle brackets as shown in Figures 4-1 and 4-7. The 1 1/2 inch sprocket is mounted on the gear motor and the 3 inch sprocket is mounted on a three inch long 3/8 inch bolt fastened to the top plate of the base. Berg part numbers for the two sprockets and the plastic covered drive chain are provided in the parts list. Figure 5-7 (Chapter 5) also provides a view of these parts installed.

Upper Body Skin

Before adding the upper body skin, the shoulder motors should also be installed. Again these are the Brevel gear motors. Mounting is simple (Figure 4-1) with a single piece of aluminum angle or bar stock approximately 4 inches long for each motor. Fasten the bracket to the top edge of the gear case via the two tapped 8-32 holes and the bolt the bracket to the underside of the upper body 5/8 inch chip board top.

The upper body skin can be any stiff thin material such as Formica used on counter tops. I obtained a thin fiberglass material in Germany where I assembled H.E.N.R.Y. The material is the same as that is available in corrugated form in the U.S. for patio roof covers.

Although the overall dimensions of the skin are shown in Figure 4-8, I recommend that you cut the skin about one inch larger in each dimension and trim the excess after it is installed. This procedure will allow for any small inaccuracies you might have introduced while assembling the upper body frame.

Also, do not drill the holes in the skin before installation. Instead, position the skin on the frame and tape it in place. Then mark and drill each vertical row of holes and immediately fasten the skin to the frame uprights with round head screws, using washers if desired. Be sure to pre-drill guide holes in the uprights so you don't split them with the screws.

This design allows you to easily remove either a portion or all the skin covering for easy future access to the upper body for wiring or repair. Figure 5-3 (Chapter 5) demonstrates this capability.

TWO ARM DESIGNS

For variety in design and experimentation with differing solutions to the robot arm design challenge, each of H.E.N.R.Y.'s arms is of a different design. One arm has a telescoping lower arm and the other has a bending elbow like a human arm. The designs of both were kept as simple as possible. Parts required for each are found in the parts list for this chapter. Again, in keeping with the overall theme of the book, as many scrap, surplus or generally easy to find parts as possible are used. With the exception of the steel rod stock used in the bending elbow arm, the elbow motors, the two gears and one collar used in each arm, the parts are generally available at a good hardware store.

As far as required tools, you will need a power drill and several taps and dies for threading parts. Although a regular drill press would be nice to have for the drilling of the steel and aluminum parts, I accomplished all drilling with one of the inexpensive drill press stands available for about $20 which uses a medium sized electric hand drill.

Telescoping Arm

Figure 4-9 provides a guide for building the telescoping arm. Start construction by first building the motor holding frame. Cut two 1 x 3 inch (end pieces) and two 1 1/2 x 3 3/4 inch (side pieces) from 1/4 inch aluminum bar stock. Drill and tap these pieces and fasten them with 4-40 machine screws to form the box shown in Figure 4-9.

The critical fitting parts in this design are the two sets of telescoping outer guide rods forming the outer guide rods. I used 3/8 inch diameter aluminum rods with a 1/4 inch diameter hole bored inside. Sliding inside these guide rods are 1/4 inch diameter iron or soft steel rods, forming the outside rods of the lower part of the arm. The exact diameter of these two telescoping rods is not important as long as the smaller one fits snugly and slides smoothly inside the larger one.

The two inner guide rods are aluminum, 1/4 inch in diameter, threaded at one end for bolt fastening to the upper guide bar. The lower ends of these inner guide rods along with the lower ends of the lower arm outer guide rods are drilled and pinned to the lower arm end piece.

As you may have already noticed, the upper and lower guide bars and the lower end of the motor holding frame are all the same size with most holes also common. The exceptions are that the end of the motor holding frame does not have the inner guide rod holes and the center hole in the upper traveling guide bar is a threaded hole.

This threaded hole in the upper traveling guide bar should have threads to match the threaded rod. I used 1/4-20 threaded rod. As you can see in the figure, a collar is fastened at the lower end of the threaded rod and the larger (48 tooth) gear is fastened to the upper end.

The motor is then installed with the smaller (20 tooth) gear meshed onto the larger gear. The method used to tie down the motor depends on the size of the motor used. I used surplus 12v motors obtained from Poly-Paks. A used windshield wiper motor should also work fine. Duct tape or nylon wire bundle tie downs work well for holding the motor in place.

The shoulder shaft connector is a 3 1/2 inch long piece of 3/8 inch diameter steel rod with a hole drilled out of one end to accept the 1/4 inch diameter shoulder gear box shaft. A set screw (or 8-32 machine screw) hole tapped in this rod is required to hold the arm on the gear box shaft.

Finally, the lower arm end piece is a three inch long piece of 1/2 x 1/4 inch aluminum bar stock with inner and outer guide rod holes drilled. As previously explained, these rods are pinned to the end piece.

Elbow Arm

Unlike the telescoping arm which is constructed primarily from aluminum stock, the elbow is constructed primarily from steel rod. The side rods are made from 1/4 inch diameter steel and the end and cross pieces are made from 3/8 inch diameter

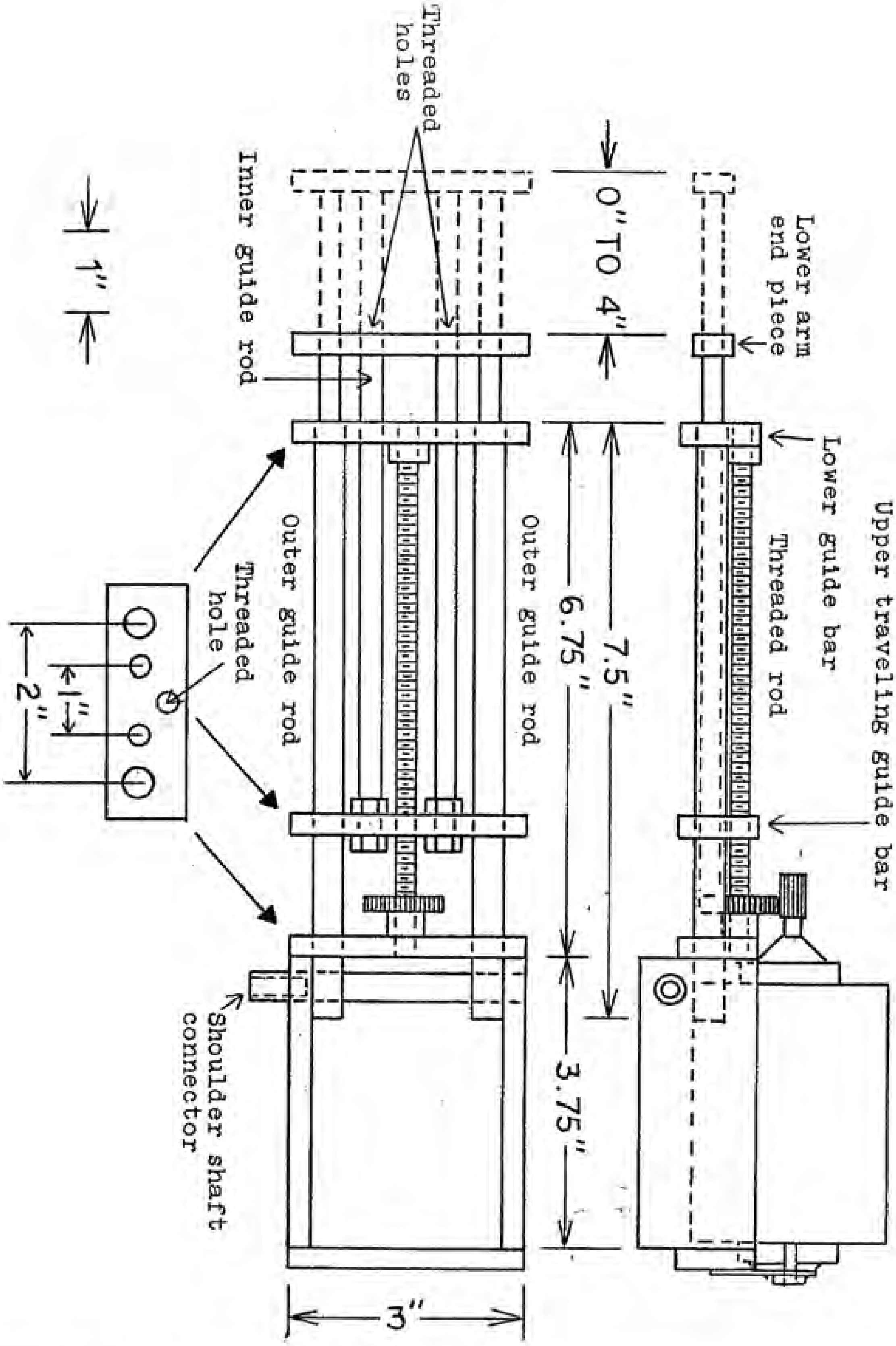


Figure 4-9. Telescoping arm.

rod.

Using Figure 4-10 as your guide, first cut the four side rod pieces from the 1/4 inch stock. Then cut the end and cross pieces from the 3/8 inch stock (a total of five pieces). Now drill the holes (all 1/4 inch holes) for the side rods and threaded rod, as appropriate, in these pieces. Then drill the ends of all pieces for tapping to 8-32 threads. But before you tap the ends, assemble the upper and lower arms and drill through the holes you have drilled to tap, about 1/16 to 1/8 inch deep into the 1/4 inch side rods. This will enable the 8-32 machine screws to set into the side rods for a good hold during final assembly. Now disassemble and tap the holes in the ends of each piece with 8-32 threads.

Cut all the aluminum pieces from 3/8 x 3/8 inch and 3/8 x 3/4 inch bar stock as appropriate. All sizes are indicated in Figure 4-10. These aluminum pieces are: One pivot anchor bar (1 5/8"), two pivot bars (1 1/2"), one traveling guide bar (2 1/2") and one shoulder connector (3"). Drill holes as indicated in the figure, remembering that the hole in the traveling guide bar is a 1/4 inch threaded hole and the rest are 1/4 inch drilled holes. Also, note that the pivot anchor bar and traveling guide bar each require two small angles fastened to their top sides so that the pivot rod can be connected with cotter pins. In addition, note that the pivot anchor bar needs to have 8-32 threaded holes on each end so that the bar can be anchored to the side bars with 8-32 machine screws. In a similar manner, the pivot bars need tapped holes on the top of the lower arm end to hold the bars from pivoting on the lower arm cross piece. The pivoting occurs on the upper arm cross piece.

In the final assembly, note that the large gear (48 tooth) is mounted on the threaded rod with the collar on the opposite side of the adjacent cross piece. The small gear (20 tooth) goes on the motor shaft. Fastening the motor to the arm may require a metal band wrapped around the motor. A strong tape, several nylon wire bundle wraps or a large hose clamp may also do the trick.

The shoulder connector extends at an angle back down the side of the arm in order to give the arm better balance and take some of the load off the shoulder motor. Note that it has 8-32 set screw holes tapped in each end for fastening to the arm and the shoulder motor gear box shaft. On the arm end you may want to use the same technique used in assembling the arm cross pieces to the side rods. That is, drill the tapped hole into the 3/8 inch diameter cross piece about 1/8 inch deep to set the 8-32 machine screw for a stronger connection.

End Effectors

The end effectors are the robots hands. Two designs are presented (Figures 4-11 and 4-12). The hand like effector (Figure 4-11) is used on H.E.N.R.Y.'s telescoping arm. The effector can grip larger objects and does very well with a glass or cup. The end pieces are made of 3/16 inch clear plexiglass. As you can see from the figure, the dimensions are not critical but are dictated by the sizes of the other components used, especially the small gear motor which drives the palm opened and closed. The motor is a plastic encased 12 volt gear motor sold

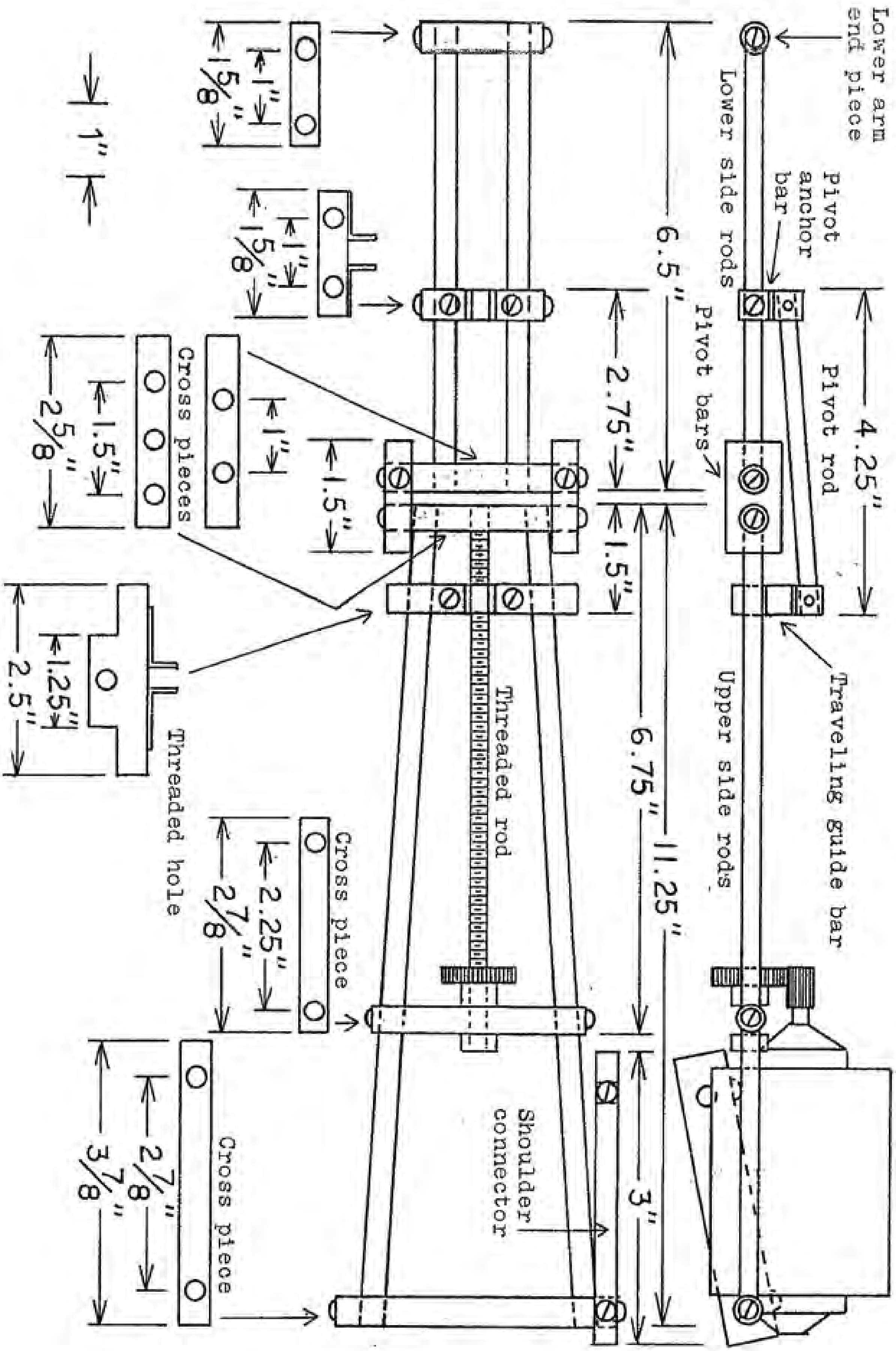


Figure 4-10. Elbow arm.

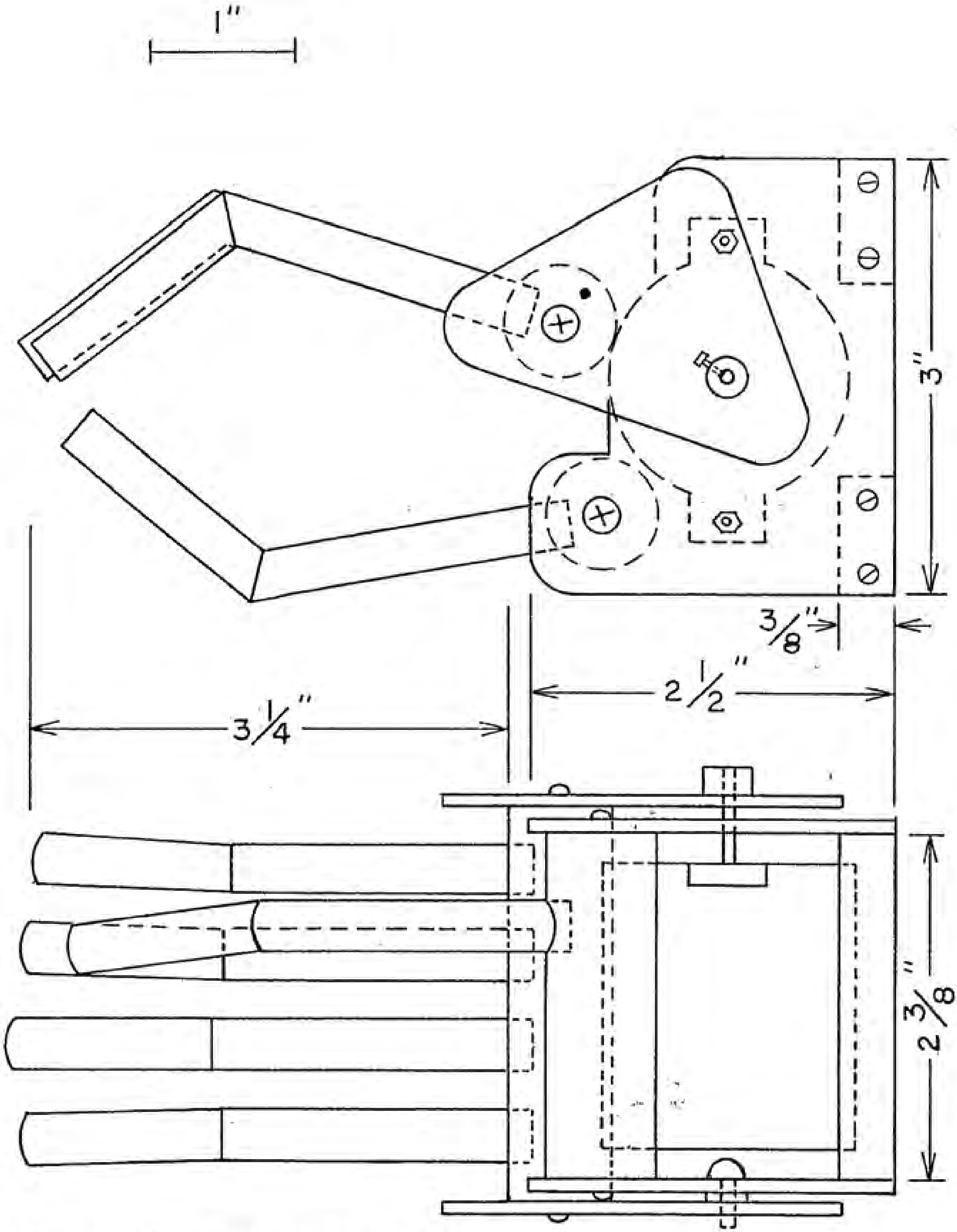


Figure 4-11. Hand like end effector.

by Edmund Scientific. It should be geared down to about 1 to 3 rpm in order to provide just enough grip to hold when power is applied but not enough to crush the object. The collar holding the motor shaft to the finger's end piece is a model airplane bell crank part.

The two palm pieces are 3/4 inch wooden dowel. The stationary palm piece holds the thumb and the moving palm piece holds the four fingers. The thumb and fingers are made from 3/8 inch wooden dowels glued together with carpenter's glue. The wooden hand avoids the metal feeling when the robot is touched by a human hand is as a result is much more friendly than cold metal.

The plexiglass end pieces are screwed to two base pieces consisting of 2 3/8 inch long 3/8 x 3/4 inch wooden strips. The palm dowels are screwed to the end pieces with a single screw on each end and the moving palm is pinned to the plexiglass with a small finish nail to prevent rotation.

The pivot points are the motor shaft on one end and a small bolt on the opposite end. A micro switch can be added to the palm in order to detect an object placed in the hand.

Pinch Type End Effector

This simple 'hand' (Figure 4-12) is adequate for holding pieces of paper, cloth or other thin objects. A hook could also be hung from this effector for picking up objects with handles. The actuator is a small 12 volt solenoid that springs opened when not powered.

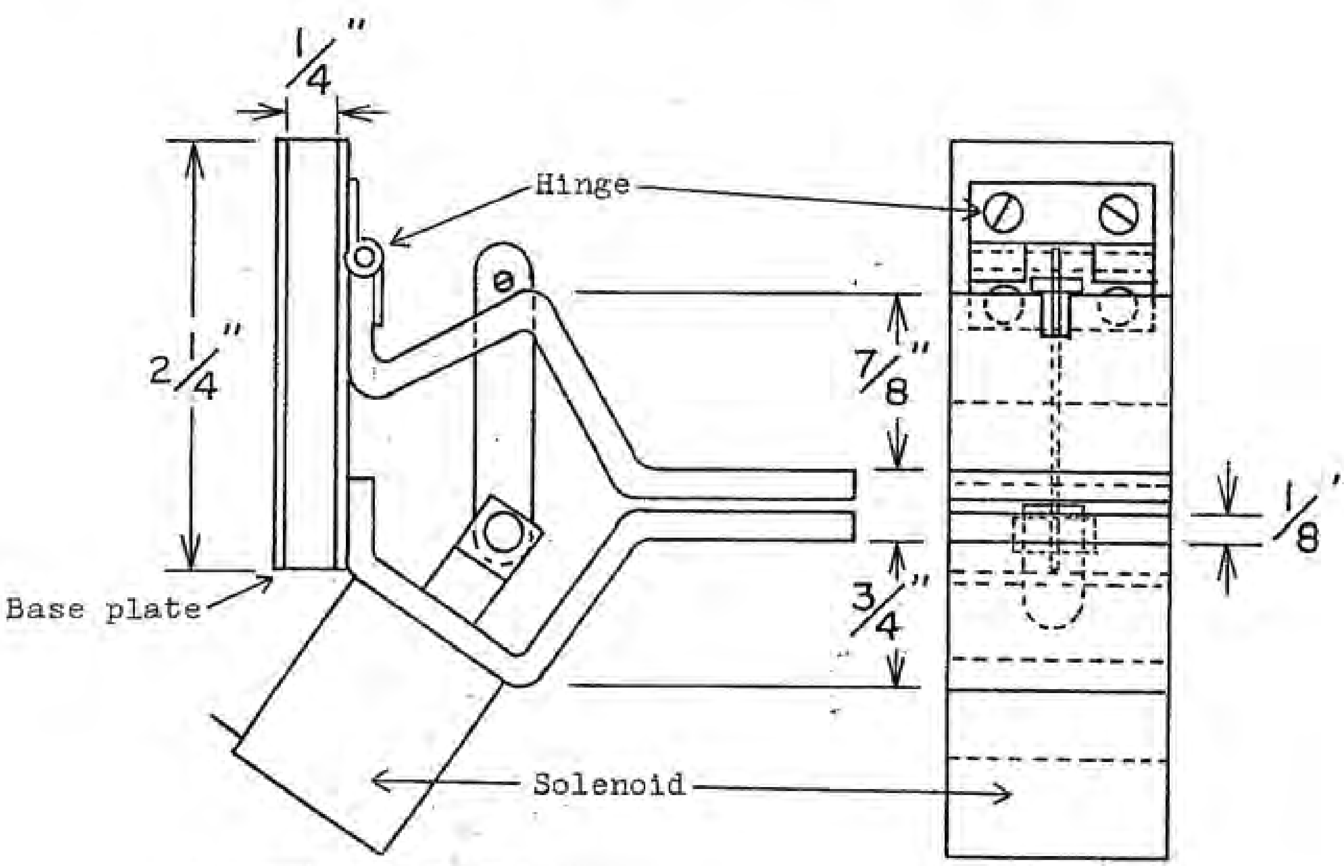


Figure 4-12. Pinch type end effector.

The base plate is made from a short length of aluminum channel used to hold 1/4 inch thick sheets of masonite together, edge to edge. The pincers are bent from 1/8 x 1 1/8 inch aluminum stock. The dimensions are not critical as long as the two opposing pieces are fit together and hinge properly. You will probably have to experiment with the bending characteristics of the aluminum stock. The best approach is to start bending with pieces longer than finally required and the adjust and cut the length to the proper size after bending is completed.

You may want to line the bare metal gripping surfaces with a no slip material in order to provide better holding power.

WIRING AND BUMPER SWITCHES

As you can see in the photos of H.E.N.R.Y. in the next chapter, free use of terminal blocks will aid in wiring and servicing the motors and switches installed in the robot. A large terminal block in the base is handy for making connections to drive motor wiring and bumper switch wiring. Also, smaller terminal blocks are a help near all other motors (shoulder and upper body rotator). Additionally, Molex type connectors are helpful where wires enter the body skin from the arms. This allows you to easily remove the arm(s) both for arm servicing and if the body skin needs to be peeled back for access to the upper body contents.

The bumper switches used for input to the computer described in the next chapter, can be seen in Figure 5-1. They are simple to mount, again using the aluminum angle supports (Figure 4-4). Normally open microswitches are used, with some type of arm which can be extended using stiff wire fastened to the switch arm.

Now that you have seen how the body is built, let's move to the next chapter and learn about the electrical portion and the 'brains', or computer control.

TABLE 4-1
Parts List

ROBOT BASE

(1) 5/32 inch aluminum sheet, 18 x 36 inches
(1) 5/32 inch aluminum sheet, 24 x 36 inches
(1) 3/4 x 3/4 inch aluminum angle, 48 inches long (cut into 3/4 inch lengths)
(1) 1 1/2 x 1 1/2 inch aluminum angle, 14 inches long
(1) 3/8 x 3/8 inch aluminum bar stock, 14 inches long
(2) 12 volt Brevel gear motors (#715-900153) (Gledhill #GM-1)
(2) Axle couplers, with set screws, 1/4" hole in one end for above gear motor shaft. Length and other end hole depends on wheels selected. (Couplers used here were drilled and tapped from Inst. Plate Pillar Posts, Berg #PA1-11.)
(2) 7 inch diameter wheels with fixed axles
(1) 2 inch diameter castor wheel
(1) 3 inch door hinge
Other hinges, quantity and sizes, depend on access door(s) design.
(1) 12 inch, Lazy Suzan, Turntable bearing, Part No.12C (Edmund)
(100) 8-32 x 1/2 inch machine screws and nuts (Digi-Key #H184 and #H224)
(20) 3/16 x 1 inch round head bolts and nuts
(1) Sears Die-Hard utility battery, 12 volt, lead acid, 280 cold cranking amps (Sears #28H9603N)

UPPER BODY

(1) 5/8 inch chip (particle) board, 24 x 48 inch
(4) 1/2 x 3/4 inch x 19 1/2 inch long wood strips
(4) 3/4 x 3/4 inch x 19 1/2 inch long wood strips
(1) 3/4 x 3/4 inch aluminum angle, 18 inches long (cut into 3/4 inch lengths)
(1) 12 inch diameter x 6 inch deep transparent plastic dome (Edmund #080179)
(1) Recessed cabinet hinge
(1) 36 inch length, nylon cord
(3) 12 volt Brevel gear motors, one to rotate upper body, two for shoulders (#715-900153) (Gledhill #GM-1)
(1) Sprocket, pinned hub type "B", 18 tooth (Berg #25EM-B-18)
(1) Sprocket, pinned hub type "B", 36 tooth (Berg #25EM-B-36)
(1) .2500 Pitch-Cable chain, 17.5 inches long (Berg #25CCF-70-E)
(1) 3/8 x 3 inch machine bolt (threaded entire length)
(3) 3/8 nuts
(2) 1/2 x 1 inch aluminum angle, 3 inches long
(100) 8-32 x 1 inch machine screws and nuts (Digi-Key #H196 and #H224)
(1) 55 x 23 inch, 3/64 inch thick Formica or 1/32 inch thick fiberglass sheet
(50) 5/8 x 8 round head sheet metal screws with washers

TABLE 4-1 (Continued)
Parts List

TELESCOPING ARM

(1) 1/4-20 threaded stock, 24 inches long (Berg #T1-7) (Includes enough for elbow arm also)
(1) 3/8 inch diameter steel ground stock, 16 inches long (Berg #S-33) (Save unused portion for elbow arm)
(1) Precision spur gear, #303 stainless steel, 48 pitch, 3/16 face, 1/4 bore, 20 teeth (Berg #P48S28-20)
(1) Precision spur gear, #303 stainless steel, 48 pitch, 3/16 face, 1/4 bore, 48 teeth (Berg #P48S28-48)
(1) Set screw collar, 1/4 inch hole (Berg #CS7)
(2) 1/16 x 5/8 inch dowel pins (Berg #D12-11)
(1) 1/4 inch diameter aluminum rod, 15 inches long
(4) 1/4-20 hex nuts
(1) 1/4 x 1/2 inch aluminum bar stock, 3 inches long
(1) 1/4 x 1 inch aluminum bar stock, 13 inches long
(1) 1/4 x 1 1/2 inch aluminum bar stock, 8 inches long
(1) 3/8 inch diameter, 1/4 inch center bored* aluminum tube, 15 inches long
(1) 1/4 inch diameter* iron or steel rod, 15 inches long
*Note: As long as smaller rod will fit snugly but slide freely within larger tube, dimensions can vary.
(100) 4-40 x 1/2 machine screws (Digi-Key #H146)
(100) 8-32 x 5/8 machine screws (Digi-Key #H188)
(1) 12 volt high torque motor (Poly Paks #7049) or surplus 12 volt windshield wiper motor

ELBOW ARM

(1) 1/4-20 threaded stock, 24 inches (Berg #T1-7) (Enough for both telescoping and elbow arm)
(2) 1/4 inch diameter steel ground stock, 24 inches long (Berg #S1-14)
(1) 3/8 inch diameter steel ground stock, 16 inches long (Berg #S1-33)
(1) Precision spur gear, #303 stainless steel, 48 pitch, 3/16 face, 1/4 bore, 20 teeth (Berg #P48S28-20)
(1) Precision spur gear, #303 stainless steel, 48 pitch, 3/16 face, 1/4 bore, 48 teeth (Berg #P48S28-48)
(1) Set screw collar 1/4 inch hole (Berg #CS7)
(1) 3/8 x 3/8 inch aluminum bar stock, 1 5/8 inches long
(1) 3/8 x 3/4 inch aluminum bar stock, 9 inches long
(1) 1/4 inch diameter bar stock, 5 inches long

8-32 machine screws (Previous listing for robot base, upper body and telescoping arm provides Digi-Key # for 100 each screws. Although you will end up with extra machine screws, the 100 quantity price from Digi-Key is very hard to beat. Hardware stores will nickel and dime you to death on this type of item.)

TABLE 4-1 (Continued)
Parts List

HAND LIKE END EFFECTOR

(1) 3/16 inch thick plexiglass, 6 x 6 inches
(1) 3/4 inch diameter wooden dowel, 6 inches long
(1) 3/8 inch diameter wooden dowel, 20 inches long
(1) Model airplane bell crank collar
(2) 3/8 x 3/4 inch wood strips, 2 3/8 inches long
(1) Small, low torque, 12 volt gear motor (Edmund Scientific)

PINCH TYPE END EFFECTOR

(1) 1/8 x 1 1/4 inch aluminum bar stock, 8 inches long
(1) 2 1/4 inches, aluminum "H" channel, masonite sheet edging
(1) 1 x 1 inch hinge
(1) 12 volt solenoid
(1) 1/16 inch sheet aluminum, 1/4 x 1 3/4 inches
Misc. machine screw fasteners as required for design variations.

NOTE: Supplier addresses are listed in Appendix B. Many items can be bought in hardware stores or scrounged.

Chapter 5
ROBOTICS ON A BUDGET - COMPUTER CONTROL

Three years ago when I decided to start building a robot I had no idea that it would lead to a fully computer controlled project (Figure 5-1). In fact, at that time, I had never even heard of a Sinclair computer. However, as I started reading more about robots and the robotics law, I realized that a true robot had to be a self contained entity able to respond to the enviornment and operate as independently as possible. A few months into the robot project I bought my first ZX81 computer, but even then I did not immediately realize the possibilities of ZX81 control. It was about three months after becoming a ZX81 owner, reading about the expansion possibilities of the ZX81, that I decided to try to combine the robot with the computer.

For those not interested in such a complex computer control project, the hardware, software and interface methods described here can easily be applied to a smaller project such as a robot arm (Figure 5-2).

However, if you decide to build the entire robot described in the previous chapter, you will end up with a robot which looks like H.E.N.R.Y. (Figure 5-3). One feature of the design, the hinged base plate allowing full access to the inside of the base (Figure 5-4), is important to the next phase of the project when wiring is accomplished. Finally, with the computer and expansion board located on the top and underside of the hinged lid for easy access (Figure 5-5 and 5-6), the installation and checkout of the computer control circuitry is simplified.

HARDWARE DESIGN

To save money and learn as much about the input/output circuitry as possible I purchased the Expansion Board and the RX-81 input/output boards in their bare board form and built then up from components purchased separately. All other boards were built up and some etched myself (Figure 5-7). A parts listing is provided in Table 5-1.

The biggest problem on the electronic side was encountered because I chose to run everything from a single 12 volt battery. The DC motors and the controlling relays cause problems because of the voltage and amperage (frequency) spikes which they generate wreak havoc on the computer. The single battery, originally a large motorcycle battery (since replaced with a Sears Die-Hard deep cycle battery) simplified the power management problem. And, the final solution to the spike problem had an added benefit for the ZX81 used as the home computer. If you have never tried to use a small computer in Europe with

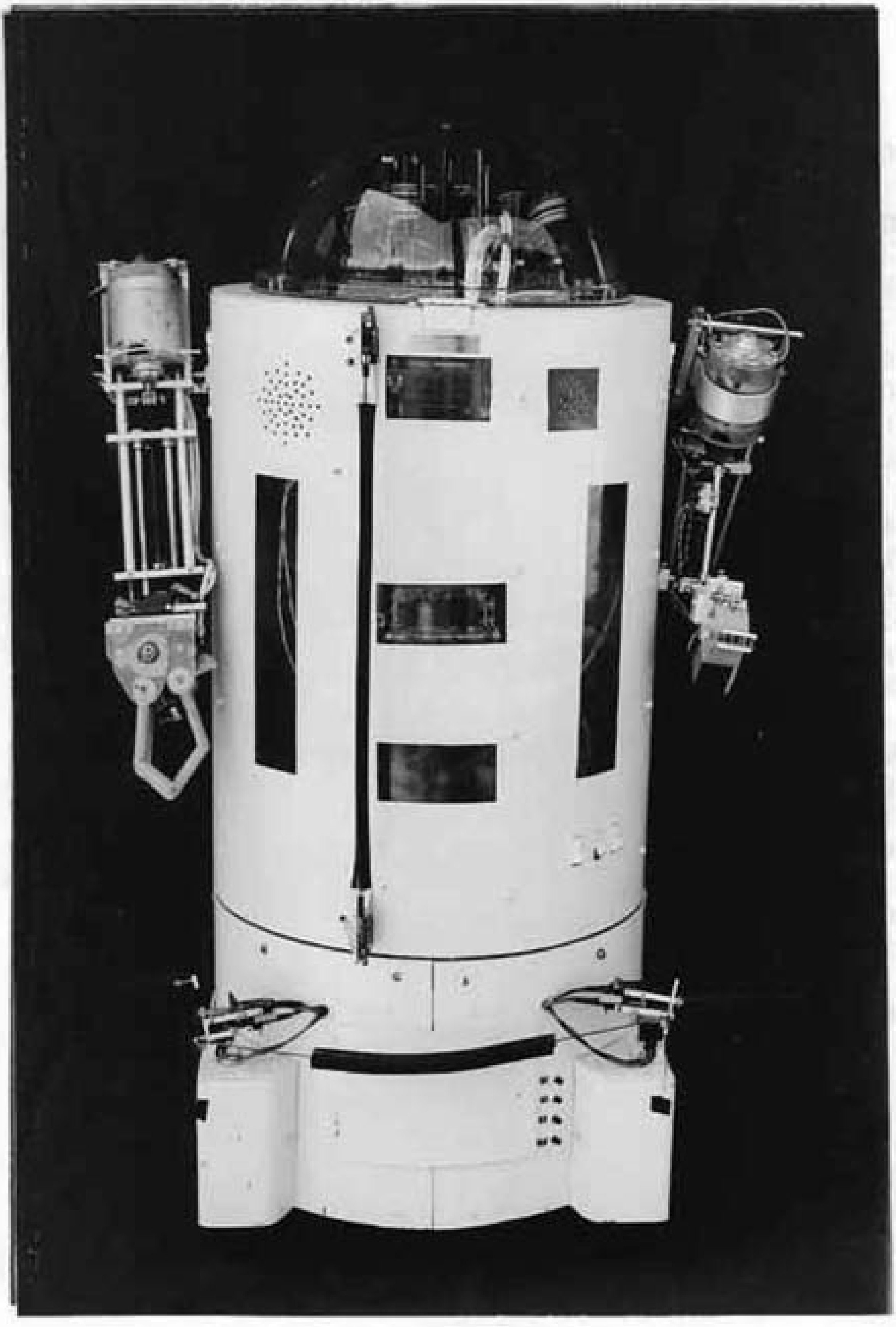


Figure 5-1. H.E.N.R.Y. the robot.

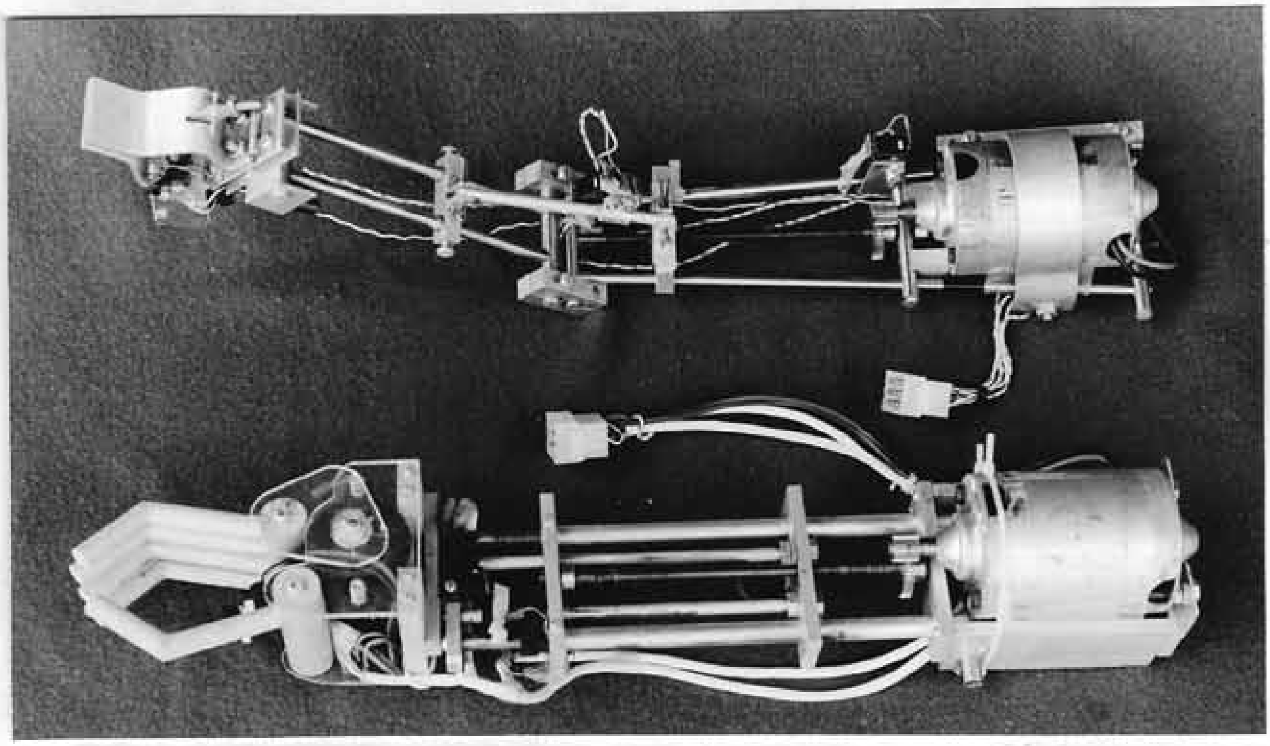


Figure 5-2. Right and left arms removed from the robot.

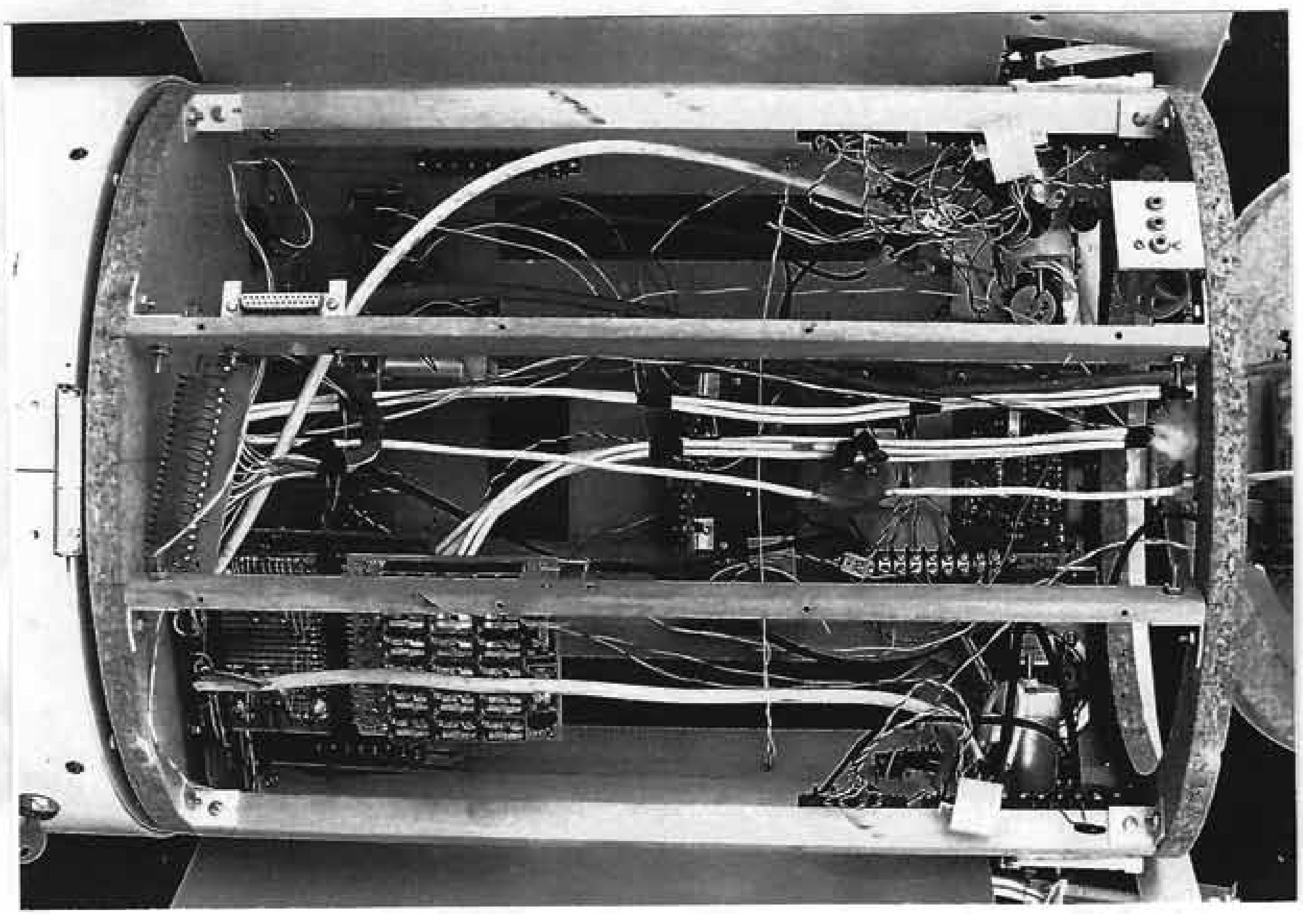


Figure 5-3. Rear view of robot with fiberglass skin peeled back from wooden frame.

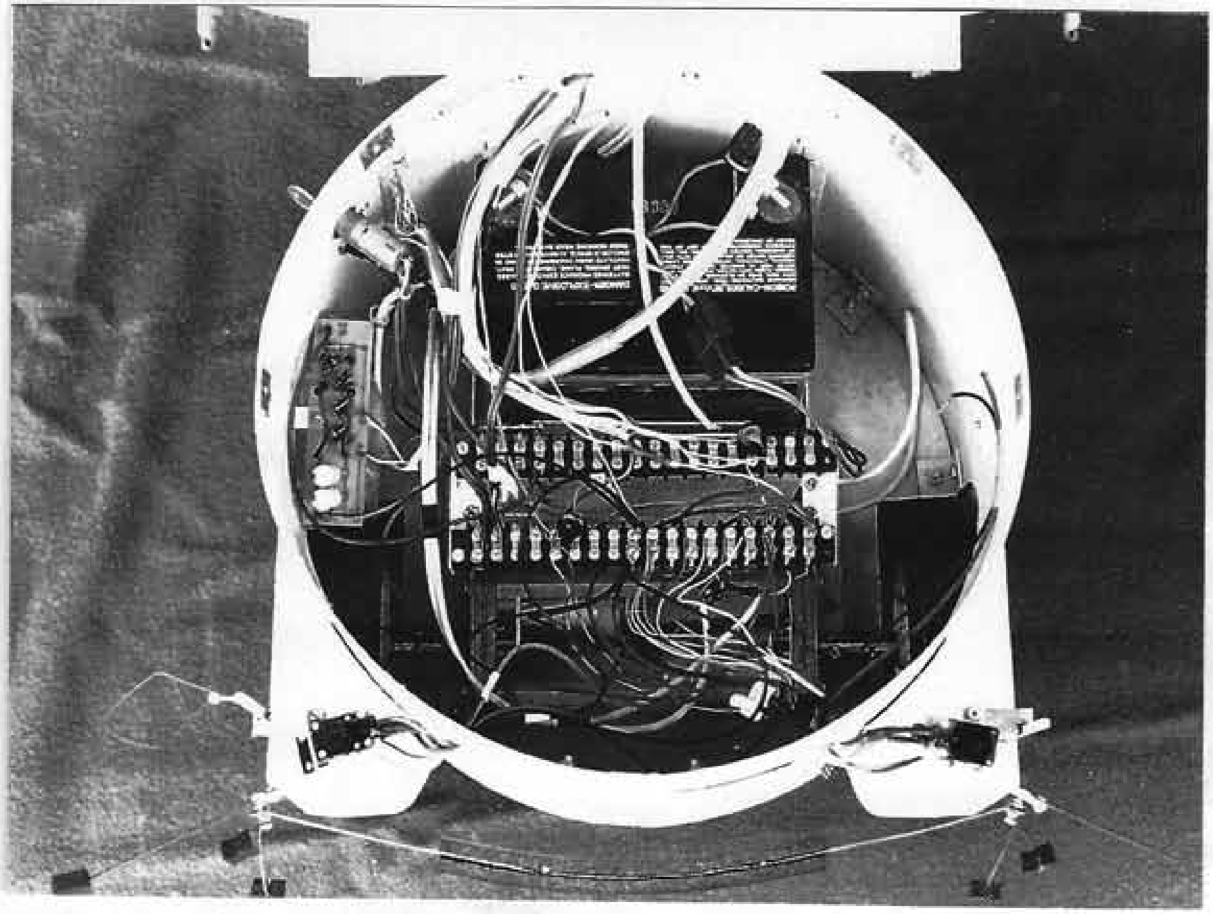


Figure 5-4. Robot upper body hinged back, looking into base.

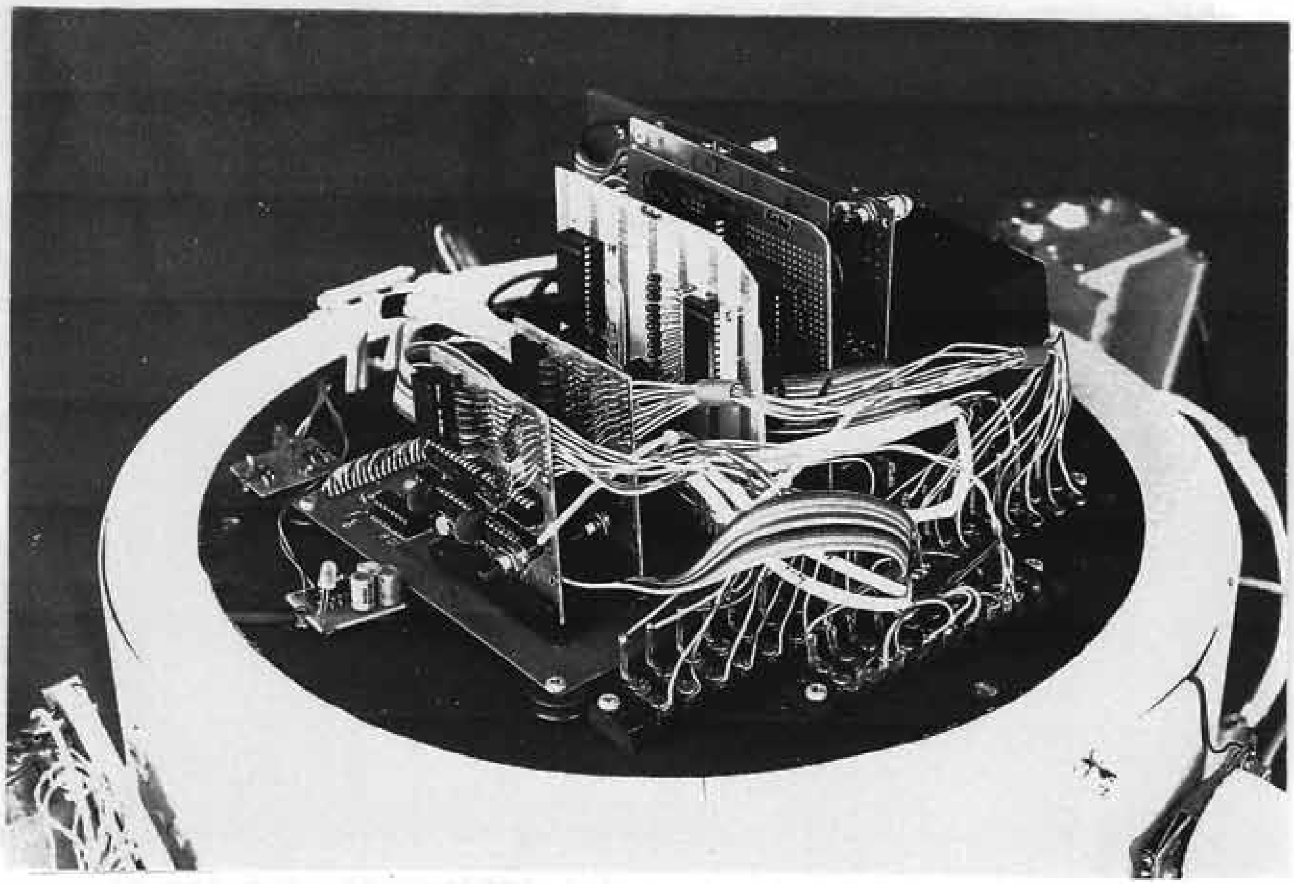


Figure 5-5. Robot top with plastic terrarium bubble removed.

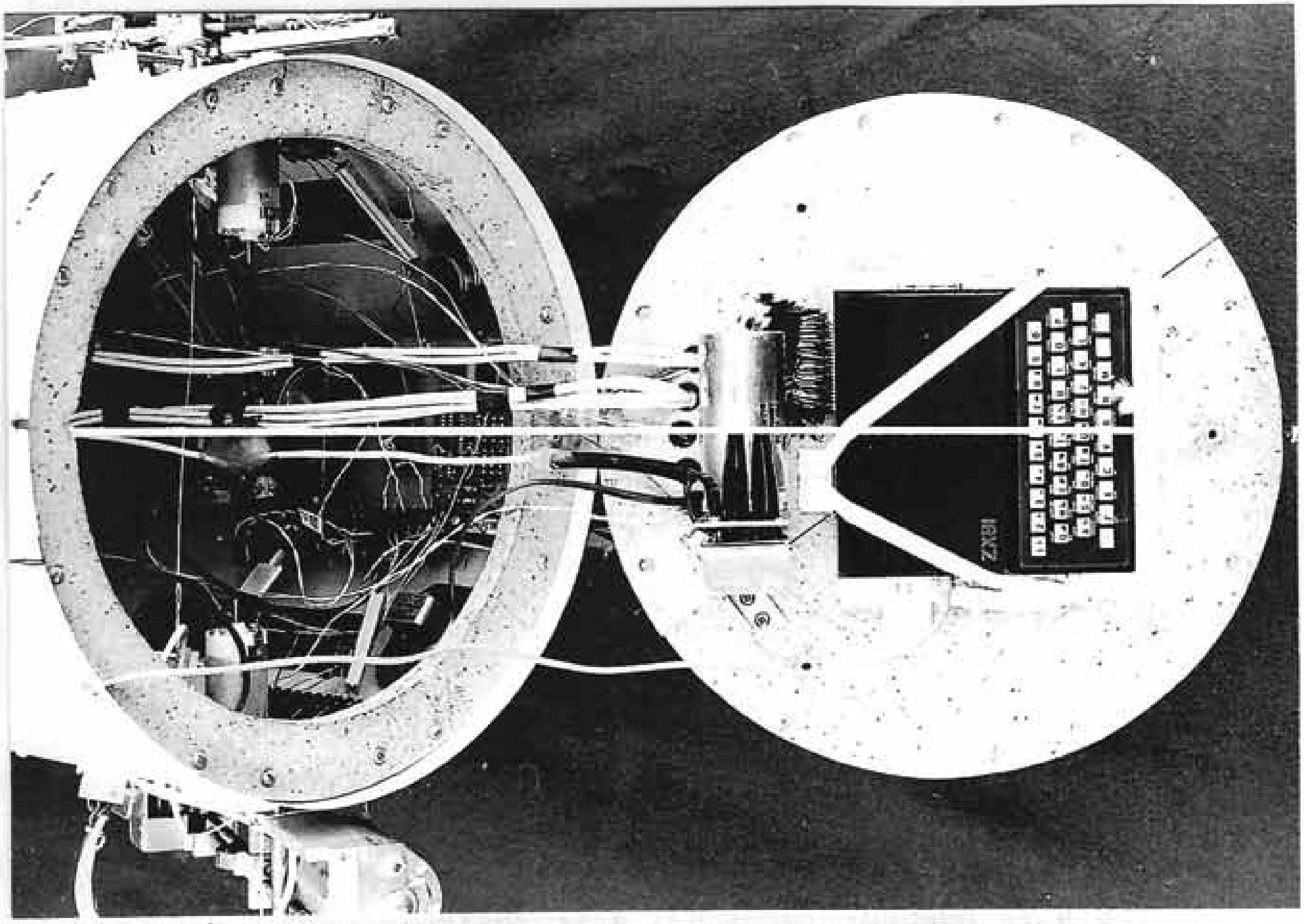


Figure 5-6. Robot top hinged back showing internal control computer and computer power supply.

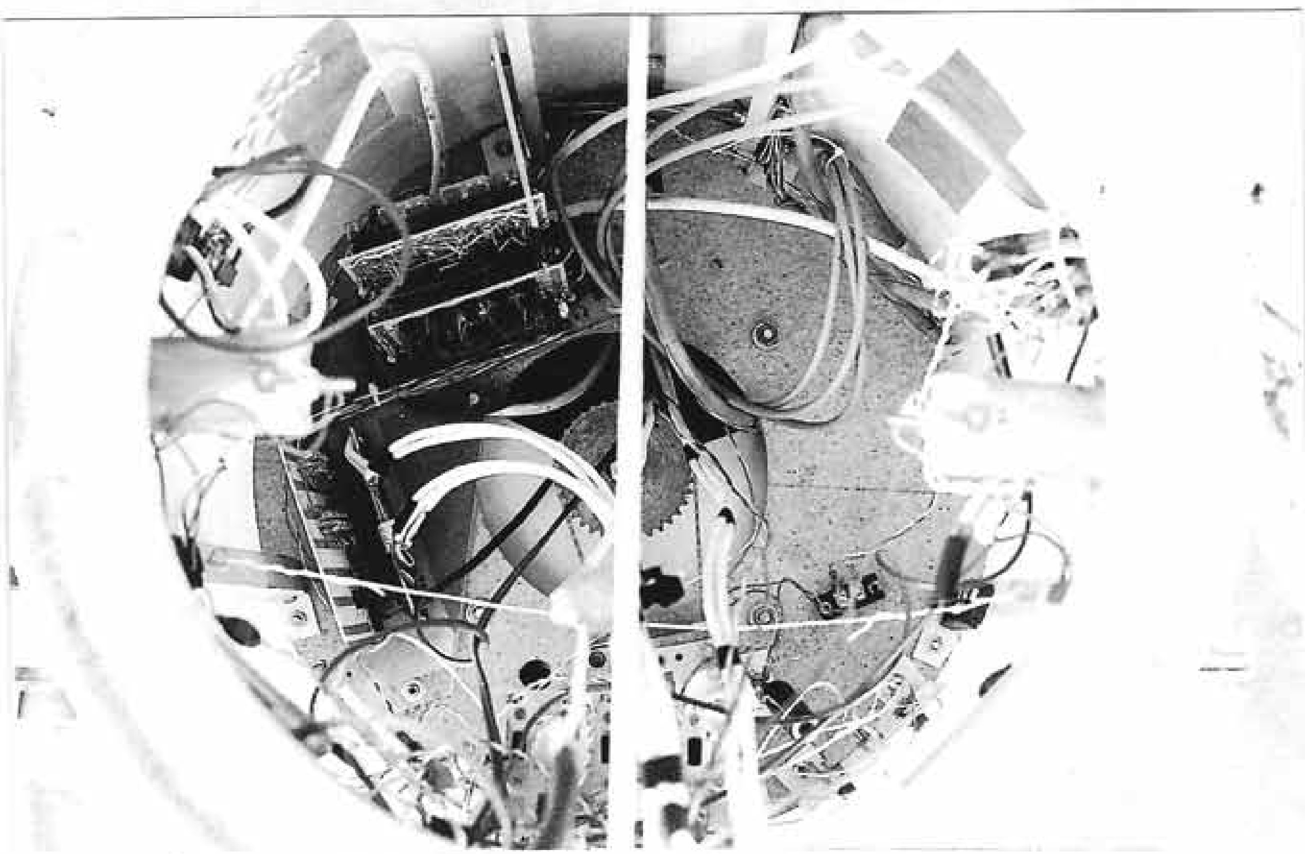


Figure 5-7. View down into upper body of robot.

Table 5-1
ROBOT PARTS LIST

Power Supply
(1) 2300 MFD, 33VDC capacitor (JAMECO # 2300@33V
(1) 5K ohm adjustable potentiometer, 10 turn (MOUSER # 593-830P)
(2) 125 uH, 3.5 amp hash chokes (MOUSER # 542-5252)

DIP Relay Board
(16) 5 volt coil DIP relays (MOUSER # 518-5002105)
(1) PC board, Hobby Board (DIGI-KEY # K160-ND)

5 Volt DPDT Relay Board
(16) DPDT relays, 5 volt coil, 12v/1 amp contacts (ALL ELECTRONICS # FRLY-6
(1) PC board (same as DIP relay board above)

Transistor Switch Board
(1) PC board (same as PC board above)

Circuit Board Holder
(4) Edgeboard connectors (DIGI-KEY # C1-22)

Intermediate Connector (Figure 5-18)
(1) PC board (same as PC board above)
(1) Edgeboard connector (one connector can be cut into two of the size needed) (DIGI-KEY # C5-50)

Buffered Buss Expansion Board
(1) Expansion board (bare board with parts list provided with board) (Budget Robotics & Computing, Expansion Board)

RX81 I/O Board
(2) RX81 I/O boards (bare boards with parts list provided with board) (Budget Robotics & Computing, RX81 I/O boards)

Ultrasonic Ranging System
(1) Ranging design OEM kit (Polaroid # 606783)
(1) RX81 I/O board as above (if this is the third RX81 board used on the Expansion Board then a Budget Robotics 'CONN' board will be needed to adapt it to a 22/44, .156" connector slot)
(1) RANG board (from Budget Robotics to interface above boards)

NOTE: Parts with common part numbers have been omitted from this list, i.e. transistors, ICs and resistors.
Robot mechanical parts (gears, motors, etc.) are those sourced from the R-E reprint series. Also, an additional source of robot parts, such as arm kits, is The Robot Works.

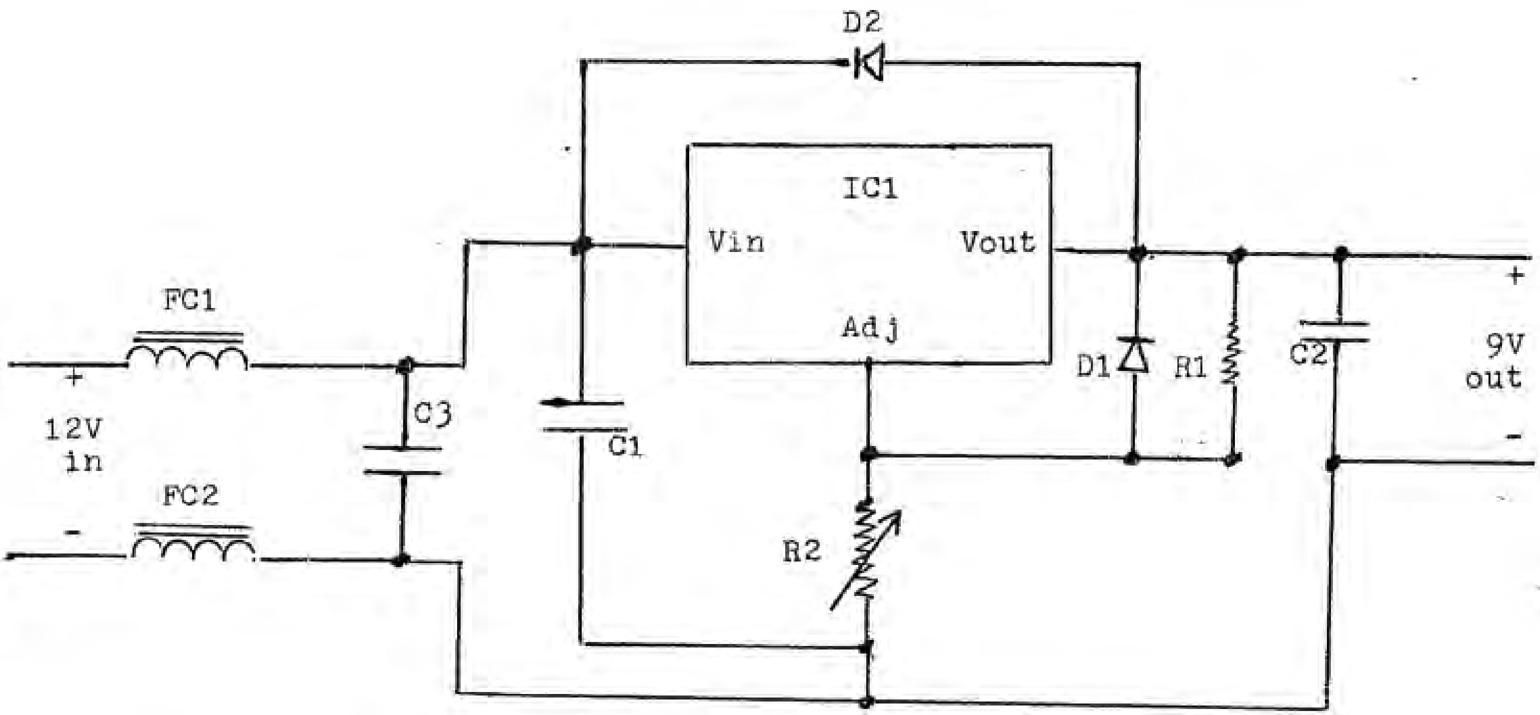
their "spikey mains" as the British refer to their noisey wall power, you can't dream of how many ways a computer can bomb. The problem is the same in West Germany where I was living when "H.E.N.R.Y." the robot was built. The power supply (Figures 5-8 and 5-9) solved all spike problems. For the robot, the 12 volts in, comes from the battery. For a home computer application, the 12 volts in, would come from a rectified 12 volt transformer input. The key to the success of the design is the large can type capacitor (C3) and the hash chokes. The capacitor is available from JAMECO (have been on sale much of the time) and the hash chokes were ordered from MOUSER (also carried by Radio Shack). The power supply worked so well on my wall powered ZX81 that all problems previously causing LOADing and SAVEing glitches have disappeared. Also, with the large can type capacitor, the house lights can momentarily dim with no effect on the computer.

The heart of the robot control is the 5 volt double pole double throw relays driven by the computer output which in turn activate the 12 volt robot motors and solenoids. The functions of these relays are listed in Table 5-2. Although only sixteen are used, the 44 finger circuit boards used are capable of handling up to twenty relays. In my design these relays are double buffered from the computer output drive with both transistor switches and small DIP relays. The 12 volt relays (5 volt coils) are wired similar to the RE reprint layout with some important differences which will be explained later. Table 5-3 gives a complete wiring table from the RX81 output through the 12 volt connections to each motor/solenoid.

I will now quickly walk you through Table 5-3 describing the control of relay #1 and then explain each component in more detail. Relay #1 is controlled by the output of D4 from one of the RX-81 input/output boards. D4 is designated as 1-5 meaning that it is the fifth of eight parallel outputs on the first RX-81 output board. It is connected with a blue with white stripe wire to finger 10 of a short PC board (Figure 5-10) which is plugged into the circuit board holder in Figure 5-11. Pin 10 of this edge connector is connected via etched foil to pin L of the transistor switches PC board edge connector. The output of the transistor switch is connected to finger 10 of its PC board which is plugged into contact with pin 10 of the edge connector. This edge connector pin is in turn connected via jumper wire to pin L of the edge connector for the DIP relay board. This corresponding finger L of the DIP relay PC board is connected to the negative side of the coil at DIP relay #1. The out contact of this DIP relay is connected to finger 10 of the DIP relay board. Pin 10 of the DIP relay board edge connector is jumper wired to pin 20 of the 5 volt relay board edge connector. The 12 volt out contact of relay #1 is connected to finger W of the relay board. The corresponding pin of the edge connector for this board (pin W) is in turn connected to a wire which runs to one of the two body rotate motor leads.

Figure 5-12 is a diagram of how each transistor switch is wired on the transistor switch PC board. The output of the RX-81 provides a latched positive voltage (approx 3.5 volts) through a 220 ohm resistor to the base of the transistor. This activates the transistor providing a ground to the collector which is then

Figure 5-8. Robot computer power supply circuit schematic.



IC1	LM350, 3 Amp adjustable power regulator
D1,D2	1N4002
R1	120 Ohm, $\frac{1}{4}$ watt
R2	5K Ohm adjustable pot (10 turn)
C1	.1 MFD, 25V
C2	1 MFD, 15 V
C3	2300 MFD, 33VDC, 50V surge
FC1,FC2	125 uH, 3.5 Amp hash filter choke

Note: Heat sink IC1 well (3 to 4 sq inches).

Table 5-2
ROBOT CONTROL RELAY FUNCTIONS

RLY	Function (motor drive)
1	Upper body rotate left
2	Upper body rotate right
3	Left drive wheel forward
4	Left drive wheel reverse
5	Right drive wheel forward
6	Right drive wheel reverse
7	(not used)
8	(not used)
9	Left shoulder down
10	Left shoulder up
11	Right shoulder down
12	Right shoulder up
13	(not used)
14	(not used)
15	Left elbow down
16	Left elbow up
17	Right elbow up
18	Right elbow down
19	Right hand close
20	Right hand open

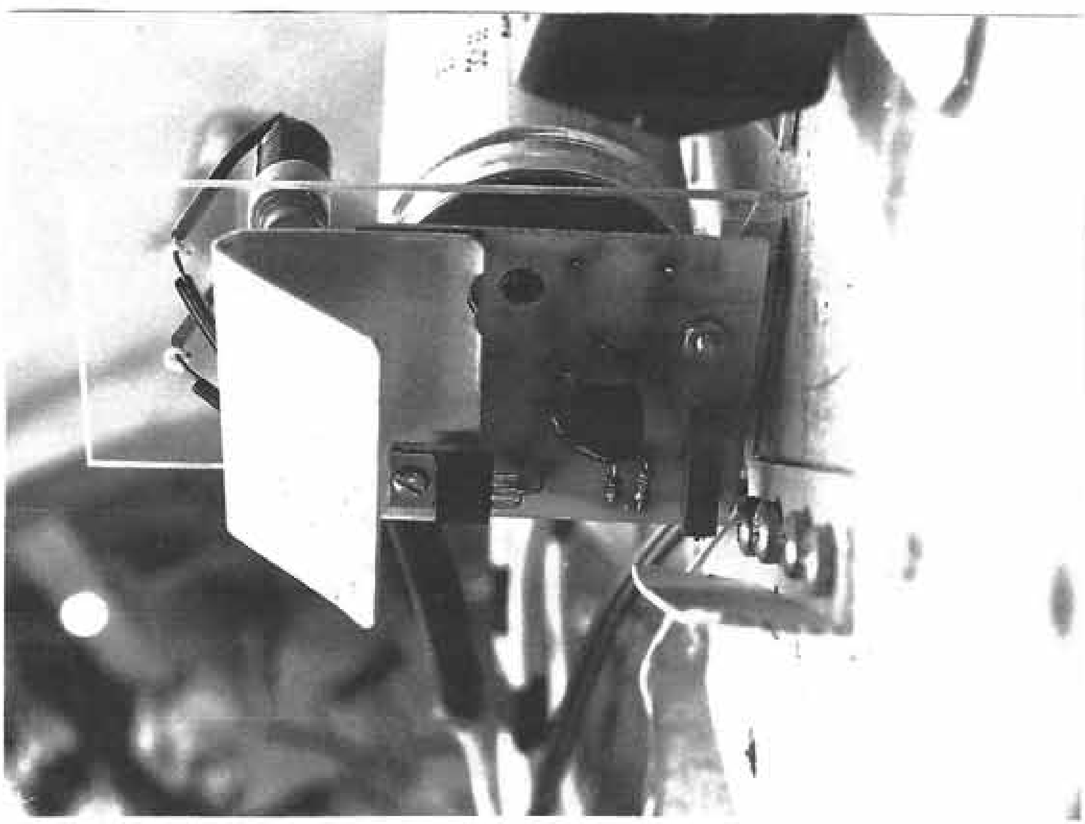


Figure 5-9. Robot computer power supply.

Table 5-3
RX81 WIRING FOR ROBOT CONTROL

	RX-81 Output			Trans. switch		DIP relays		5v relays	
RLY	DO-7	Wire color	PC Conn	Base (in)	Coll (out)	Coil (in) –	Contact (out) +5v	Coil (in) +5v	Contact (out) +12v
1	1-5	blu/wht	10	L	10	L	10	20	W
2	1-6	wht/blu	11	M	11	M	11	17	V
3	1-1	wht/grn	6	F	6	F	6	R	11
4	1-2	grn/wht	7	H	7	H	7	L	12
5	1-3	wht/grn	8	J	8	J	8	6	D
6	1-4	gry/wht	4	D	4	D	5	3	C
7			20	X	20	X	20	21	
8			21	Y	21	Y	21	15	
9	2-1	wht/grn	12	N	12	N	12	N	S
10	2-2	grn/wht	13	P	13	P	13	K	13
11	2-3	wht/gry	14	R	14	R	14	7	E
12	2-4	gry/wht	15	S	15	S	15	2	F
13			22	Z	22	Z	22	19	
14			5	E	5	E	4	P	
15	2-5	blu/wht	16	T	16	T	16	M	U
16	2-6	wht/blu	17	U	17	U	17	8	T
17	2-7	org/wht	18	V	18	V	18	5	J
18	2-8	wht/org	19	W	19	W	19	1	H
19	1-7	org/wht	9	K	9	K	9	18	22
20	1-8	wht/org	3	C	3	C	1	4	10
	Gnd	blu	2	2, B (emitters)		2		A	
+5 volts						3			

RX-81 output board #1 wired as "out 7"
RX-81 output board #2 wired as "out 6"

Example: output D0 from board wired as "out 7" is listed above as 1-1.

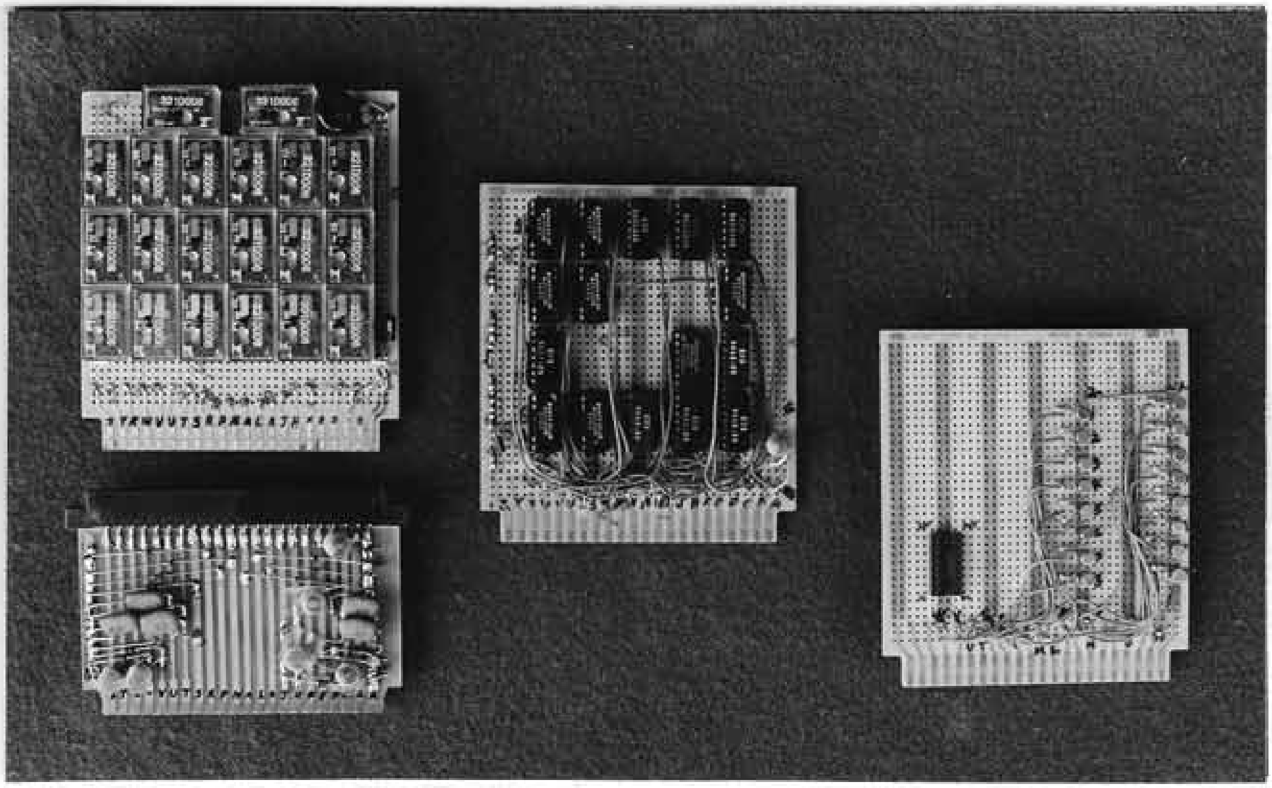


Figure 5-10. The three robot motor driver circuit boards.

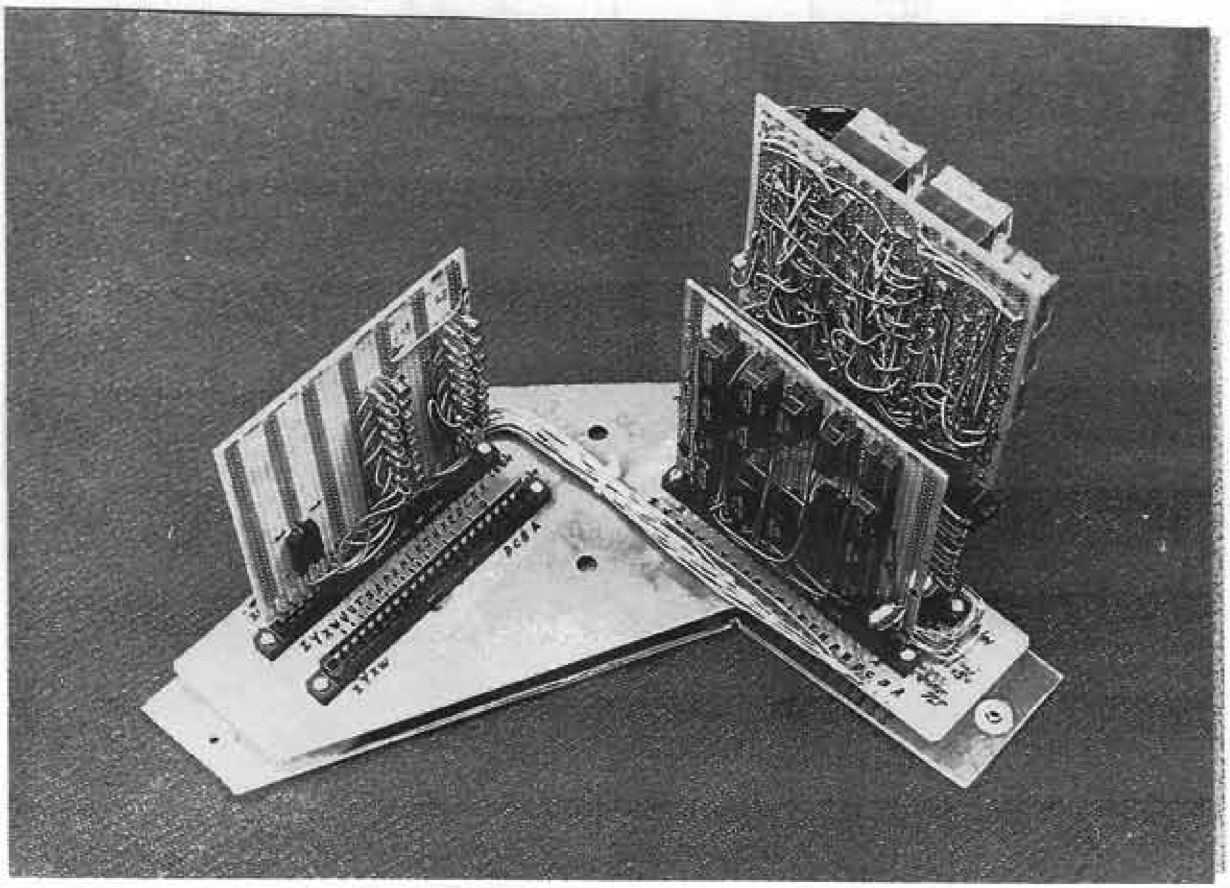
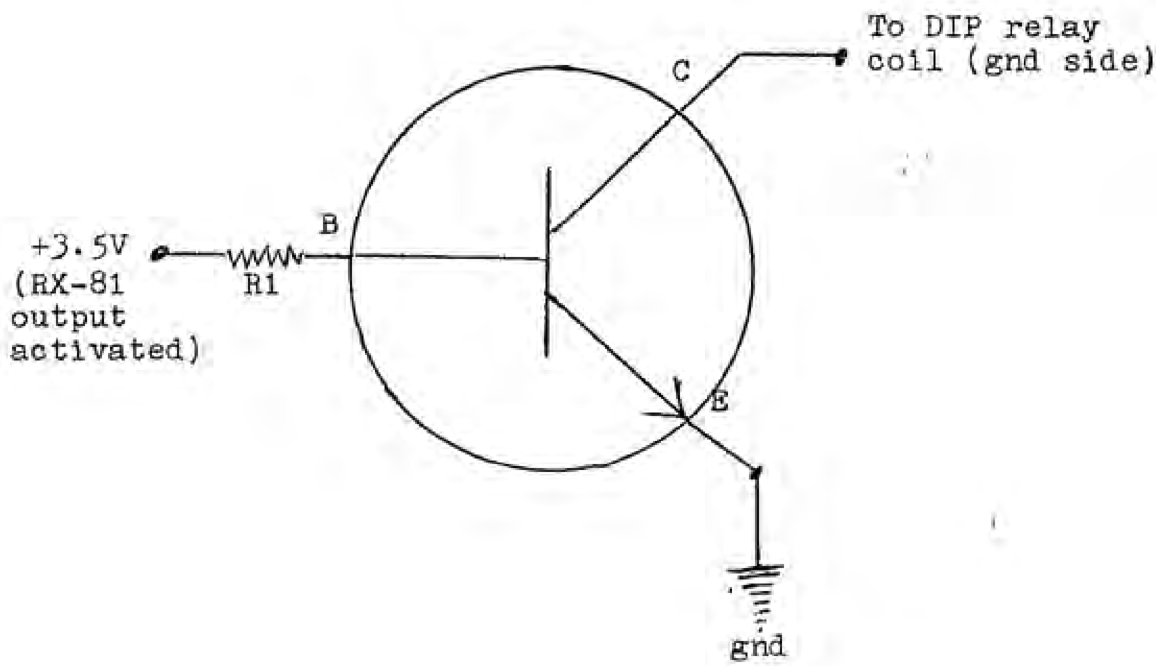


Figure 5-11. Robot driver boards plugged into holder, before installation on base of upper robot body.

Figure 5-12. Robot transistor switch driver circuit schematic.

TRANSISTOR SWITCHES
(Total of 16)

2N2222



R1 220 Ohm ¼watt

sent to the DIP relay.

The DIP relay coil (Fig 5-13) is then activated by completion of the circuit to ground. The closed relay contact then provides 5 volts out of this circuit. Note the diodes and capacitor in this circuit used to suppress spikes. This is very important.

The 5 volts provided by the closed contact on the DIP relay then activates the coil of the appropriate motor control relay (Fig 5-14). These DPDT relay contacts must be able to handle 1 to 3 amp loads, depending on the size of the DC motor or solenoid driven. Large DIP mounted relays of sufficient amperage can be used but they can be expensive ($4 to $5 each). I found some miniature 6 volt DPDT relays rated at 3 amps which were cheaper and worked quite well. One source is ALL ELECTRONICS CORP at only $1.75 each. Sixteen to twenty of these can be packed on a 4 x 5 inch PC board, although the more expensive high amp rated contact DIP relays would not have to be packed so closely.

An example of both the forward and reverse relays are shown in Figure 5-14 because they are wired together to prevent accidental shorting of the output to the motor. Each of the two relays for each motor (one for forward and one for reverse) provide opposite polarity power. If you made no special wiring provision it would be possible to activate both relays at the same time and cause a direct short circuit between +12 volts and ground. Two extra wires between the two relays along with a modification of a direct hookup, protect against this. Looking at Figure 5-14 you see that the coil input (+5 volts) to the righthand relay coil (relay #1) will not cause the coil to be activated unless relay #2 (lefthand relay) is not activated. This is because the coil of relay #1 has no ground to complete the circuit unless relay #2 is in the normally closed position. Notice that the ground for relay #1 coil is supplied through the normally closed contact of relay #2. This way, even if both the reverse and forward circuits are activated at the same time, only one will work.

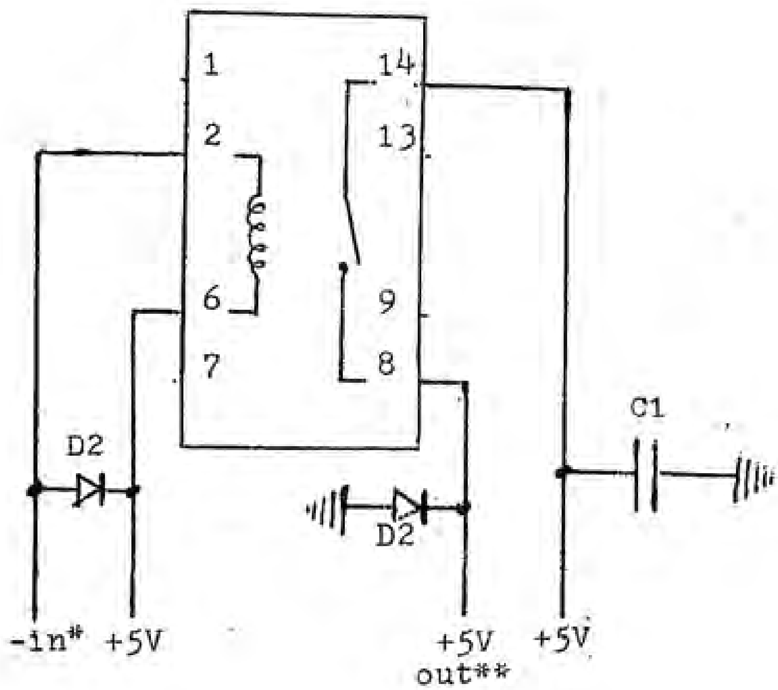
Also note the diode and capacitor protection on these relays. This feature is especially important as the unloading of these relays introduces a lot of unwanted garbage into the electrical circuits.

For some motors in your robot the 12 volt output of the relays in Figure 5-14 will be hooked directly to the motor leads. However, you may wish to install limit switches on the mechanism of some motor drives such as the arm elbow motor. The limit switches are normally closed contact microswitches. A simple but effective limit switch circuit using two diodes and a four post, terminal strip is depicted in Figures 5-15 and 5-16. If you happen to get it wired backwards (50/50 chance) just reverse either the 12 volt input leads or reverse the motor output leads.

Before we turn to software, a few words about input stimuli to the input board. First, the input board is the same board that is used for output, the RX81 I/O board in this case. The RX81 provides eight input lines and a ground. To input a signal to D0, or as I refer to it, 1-1 (input line #1 of RX-81 board #1), you just connect the input line to ground. For my robot I

Figure 5-13. Robot DIP relay circuit schematic.

DIP RELAYS
(16 total)



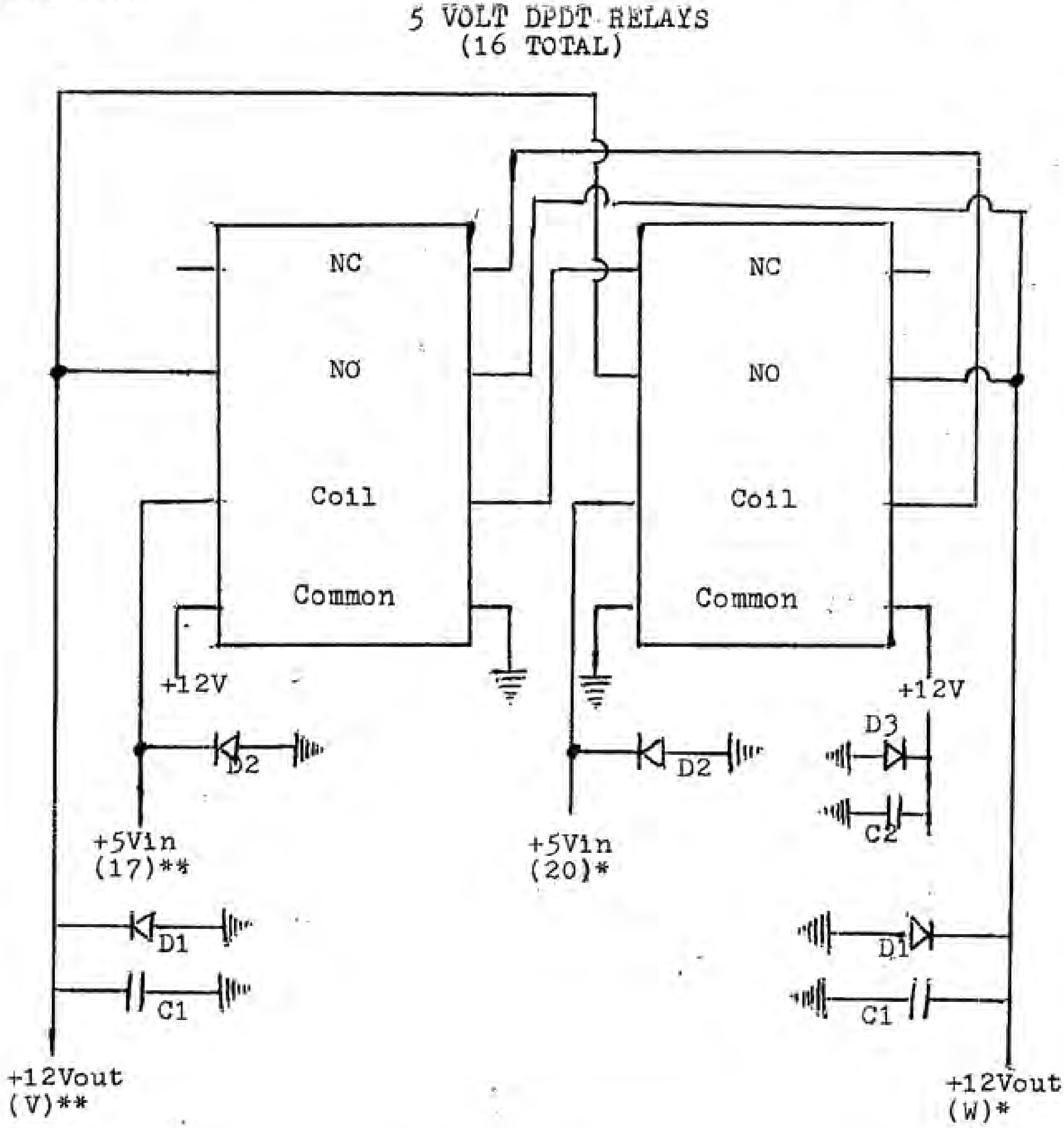
D2 1N4148

C1 .01 MFD, 10V

* For DIP #1 this would be wired to connector L

** For DIP #1 this would be wired to connector 10

Figure 5-14. Robot 5 volt coil DPDT motor control relay circuit schematic.

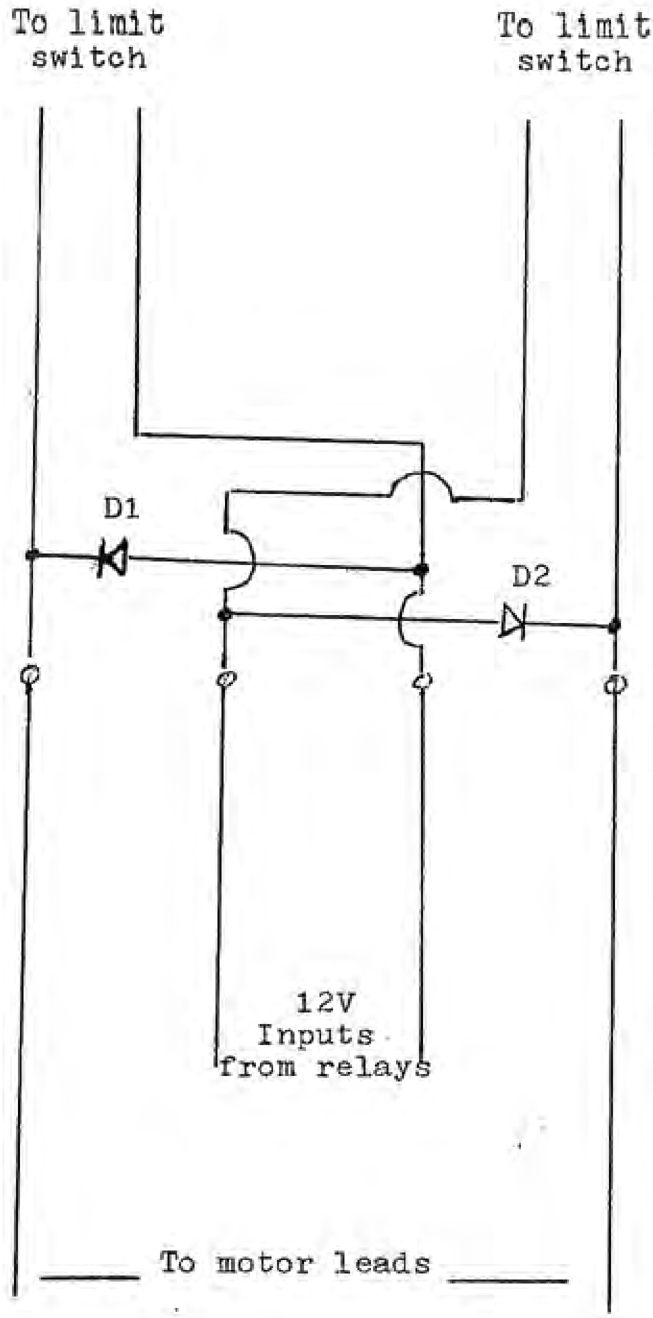


D1	1N4002
D2	1N4148
D3	1N5401
C1	.01 MFD 25V
C2	22 MFD 35V

NC = Normally closed contact

NO = Normally open contact

*Wired to this connector if relay #1
**Wired to this connector if relay #2



D1, D2 1N4002

Figure 5-15. Robot motor limit switch diode terminal schematic.

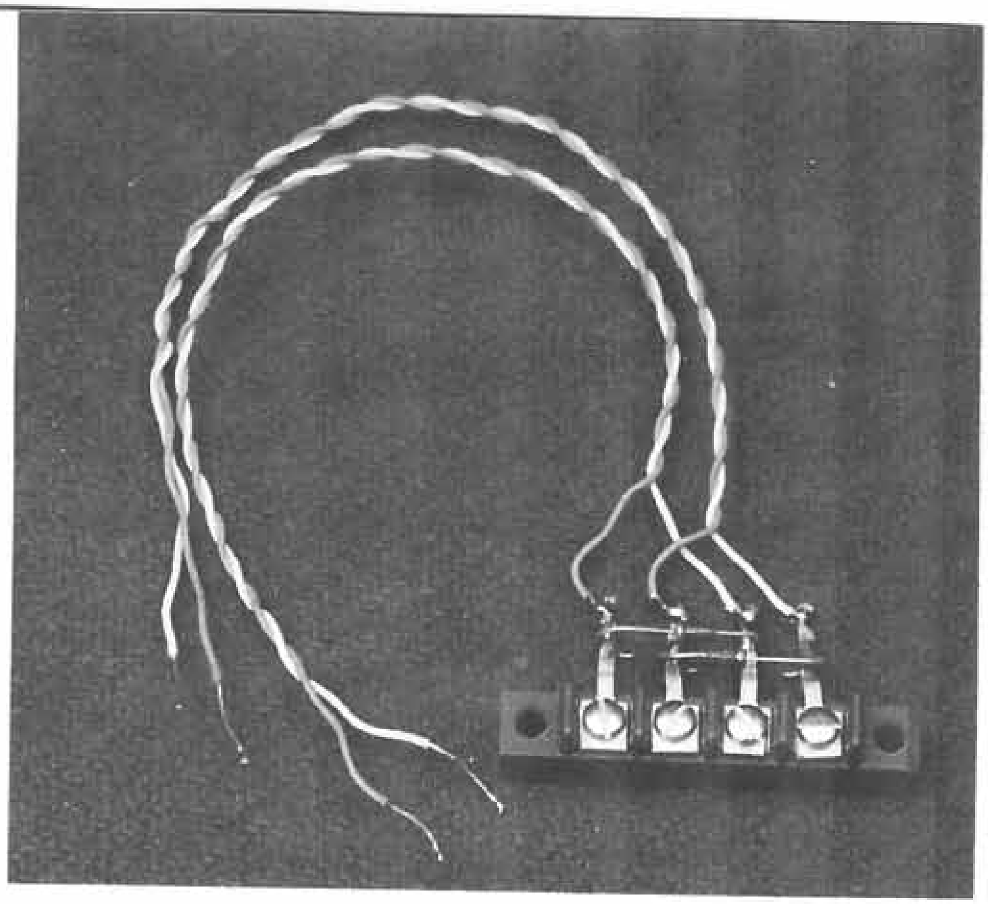


Figure 5-16. Robot motor limit switch terminal strip and diodes.

installed normally open contact microswitches to the exterior of the robot with spring wire extensions covered with foam rubber pads. The bumper switches include one as a front bumper, one as a right bumper and one as a left bumper. I actually installed two sets of two front bumper switches each, one set with a spring wire horizontally between them and another set with a spring wire vertically between them. This arrangement gives a larger striking area if the robot runs head on into something.

An alternate way to accept input commends to the computer is to wire the external microswitches as keys on the computer keyboard. This method works, but is not totally satisfactory. The computer can be set up to scan the keyboard for specific key inputs but a problem arises when two keys (or microswitches) are closed at the same time. This can easily happen if the robot works its way into a corner and hits the front and the side sensors together. The computer will not normally accept any keyboard input if this happens and it could even be seen as an illogical input and blow the program.

With sensors wired to the RX81 Input/output board a program can be written to accept any combination of simultaneous sensor inputs at the same time, and properly recognize them. Also, a machine code routine can be written to scan the inputs much faster than the keyboard method. More on that in the software explanations.

SOFTWARE TO GET STARTED

Now that your robot motors and bumper switches are wired up to the ZX81 computer as described, you are ready to start writing software to "control the world" or at least the world of your robot. All programs are for 8K ROM. The second listing will require 16K RAM.

A few programming instructions, in BASIC, are explained in the data sheet that comes with the RX81, to get you started, but my programs written for direct robot control will be explained in detail. First some simple program sequences will be stepped through as a learning experience; then a sample robot demonstration program; and lastly a complete robot control and demonstration program of about 13K RAM will be explained and listed. This last program is the one H.E.N.R.Y. used to win the Golden Droid award.

As explained in Chapter 3, all programs designed to drive the RX81 I/O circuit start with a REM statement containing the machine code. To enter the machine code, enter a "1 REM" statement containing at least 24 spaces. Then enter the listing from Table 3-2.

Enter "RUN" and as each address location is displayed, enter the appropriate code from the CODE table (Table 3-3) in Chapter 3. Then list the program and check the 1 REM statement by comparing it with the listing in Figure 3-3 (Chapter 3). If a symbol is incorrect, just correct it by POKEing the address directly. If it is ok you can delete lines 10 through 50 and procede with the sample BASIC software. This machine code now stored in the 1 REM statement can be SAVEd along with the BASIC listing.

To activate an output line you first POKE the binary number for the line, then POKE the number of the output board (if you have more than one board) and then activate the command with an OUT USR statement. For example, if you wanted to activate output line #3 on an output board wired as "OUT 7" you would need the sequence in lines 100 through 120 of Table 5-5. As listed in Table 5-4, this would make the right wheel move the robot forward. To make the right and left wheels go forward you would need lines 130 through 150 of Table 5-5. To turn off these motors you would POKE 0, as is listed in lines 160 through 180 of Table 5-5.

As you might have gathered from Table 5-4, there are some memory saving shortcuts to this programming. If you add the LET statements (lines 10 through 30) from Table 5-6 to the program in Table 5-5, you save memory and make the program statements easier to type into the computer. Lines 100 through 180 in Table 5-5 can now be shortened to those depicted in Table 5-6.

Other refinements are possible to further save program steps. For example, lines 140 and 170 can be omitted because location B (16536) was POKEd at the beginning of the sequence and the output remains latched throughout the sequence. If you change from the OUT 7 board to a second board wired as OUT 6 and back again, then the POKE 16536 step would have to be included each time, so that the proper board would be addressed.

An example of a test program is provided in Table 5-7. This program activates a single line and then stops. If you enter CONT it will then turn the output line off and stop again.

Table 5-4
ROBOT CONTROL SOFTWARE COMMANDS

Function	Relay	Output	POKE A,	POKE B,
Body left	1	1-5	16	7
Body right	2	1-6	32	7
L wh fwd	3	1-1	1	7
L wh rev	4	1-2	2	7
R wh fwd	5	1-3	4	7
R wh rev	6	1-4	8	7
L sh dn	9	2-1	1	6
L sh up	10	2-2	2	6
R sh dn	11	2-3	4	6
R sh up	12	2-4	8	6
L el dn	15	2-5	16	6
L el up	16	2-6	32	6
R el up	17	2-7	64	6
R el dn	18	2-8	128	6
R hand close	19	1-7	64	7
R hand open	20	1-8	128	7

Switch	Input	IF IN=	POKE B,
Front bumper	1-1	1	7
Left bumper	1-2	2	7
Right bumper	1-3	4	7
Right palm	1-8	128	7

(Note: A= 16534 and B= 16536)

Enter CONT and it will activate the next output line and so forth, repeating the process.

The input line activation is a little simpler. You first ready the input for activation and then use IF statements to look for the proper input to activate a response. For example: "500 LET IN = USR 16514", readies the input for activation. "510 IF IN = 1 THEN GOTO 100", jumps the program to line 100 if input line #1 is activated. If more than one input board is used, a "POKE 16436" line would have to precede tha USR line to identify the board in accordance with how it is wired. In my robot only one board is used for input as eight input lines are more than enough for the bumper switches. As I add more sensors such as optical encoders and ultrasonic rangefinder for example, the second input board will have to be used. Remember that the same memory saving techniques can be used (LET F = 16514). A sample

Table 5-5
DEMONSTRATION PROGRAM

```
  1 REM                 (with machine code characters)
100 POKE 16534,4       (addresses line #3)
110 POKE 16536,7       (addresses board wired as OUT 7)
120 LET OUT = USR 16533      (activates command)
130 POKE 16534,5       (addresses line #1 and #3 together)
                        by adding their binary numbers 1+4)
140 POKE 16536,7       (addresses board wired as OUT 7)
150 LET OUT = USR 16533      (activates command)
160 POKE 16534,0       (deactivates all output lines)
170 POKE 16536,7       (addresses board wired as OUT 7)
180 LET OUT = USR 16533      (activates command)
```

Table 5-6
DEMONSTRATION PROGRAM REFINEMENTS

```
 10 LET A = 16534
 20 LET B = 16536
 30 LET C = 16533
100 POKE A,4
110 POKE B,7
120 LET OUT = USR C
130 POKE A,5
140 POKE B,7
150 LET OUT = USR C
160 POKE A,0
170 POKE B,7
180 LET OUT = USR C
```

Table 5-7. Robot control program listing 1.

```
  1 REM <=" J?<=" J▒4 PLOT  =" J▒4
LIST ▘ TAN Y" PEEK ▘TAN
 20 LET A=16534
 30 LET B=16536
 40 LET C=16533
 50 POKE A,1
 60 POKE B,6
 70 LET OUT=USR C
 80 STOP
 90 GOSUB 530
100 STOP
110 POKE A,2
120 POKE B,6
130 LET OUT=USR C
140 STOP
150 GOSUB 530
160 STOP
170 POKE A,4
180 POKE B,6
190 LET OUT=USR C
200 STOP
210 GOSUB 530
220 STOP
230 POKE A,8
240 POKE B,6
250 LET OUT=USR C
260 STOP
270 GOSUB 530
280 STOP
290 POKE A,16
300 POKE B,7
310 LET OUT=USR C
320 STOP
330 GOSUB 570
340 STOP
350 POKE A,32
360 POKE B,7
370 LET OUT=USR C
380 STOP
390 GOSUB 570
400 STOP
410 POKE A,64
420 POKE B,7
430 LET OUT=USR C
440 STOP
450 GOSUB 570
460 STOP
470 POKE A,128
480 POKE B,7
490 LET OUT=USR C
500 STOP
510 GOSUB 570
520 STOP
530 POKE A,0
540 POKE B,6
550 LET OUT=USR C
560 RETURN
570 POKE A,0
580 POKE B,7
590 LET OUT=USR C
600 RETURN
```

routine to scan for inputs is depicted in Table 5-8.

Table 5-8
INPUT DEMONSTRATION

```
900 LET IN = USR F
910 IF IN = 1 THEN GOTO 100
920 IF IN = 2 THEN GOTO 200
930 IF IN = 4 THEN GOTO 300
940 GOTO 9000
```

Table 5-9 is an extract from the program that was in H.E.N.R.Y. when he won the Golden Droid award. It moves the robot forward (right + left wheel forward), while scanning for hits on the microswitch sensors in the form of inputs. If the right bumper input is activated the robot stops its forward motion, backs up, turns left about 30 degrees and than continues forward, again sensing for bumper inputs. The sequence is similar for a left bumper input except it turns to the right about 30 degrees before continuing. The sequence for a front bumper hit is a little different in that a random number generator is used so that 50% of the time the robot turns right 60 degrees and 50% of the time it turns left 60 degrees before continuing forward. A counting step is also included as part of the input scanning routine so that the robot moves forward for about nine seconds and then generates a random number between zero and one. One third of the time it will stop and go into a body rotating and arm demonstration subroutine.

Several timing techniques are used in the program. PAUSE is a good technique when interruption is not required, such as arm movements, if there is no danger of hitting or running into something. The FOR NEXT loop is a good technique where input scanning is required, when there is a possibility of the movement causing a collision with another object. A counting technique as a loop will also work well in this situation.

Expanded version software

Provided below is a complete listing and documentation explanation of the 13K program used by H.E.N.R.Y. when he won the Golden Droid award in Albuquerque. This software includes operating instructions for the voice recognition and voice output hardware described in chapters that follow. The complete program listing is contained in Table 5-10. Complete documentation is contained in Table 5-11. The documentation is broken down into routines and subroutines, identifying them with program line numbers. Variables are identified and special instructions are also provided in the table. Remember that if a peripherial (like the voice input device) is not installed, you must delete the corresponding software routines or else the program will get hung up. The documentation provided should simplify this task.

SOME FINAL ADVICE

At this point you should be off to a good start with your robot project. My whole idea in the beginning of the project to internally control a robot with a ZX81 computer was to demonstrate how much power the little computer really has. I have still not maxed out the 16K RAM, although I am ready with a 16K Byte-Back module to add to the Sinclair 16K RAM pack. The Computer Continuum expansion board was chosen because of its three ampere capacity for five volt supply as additional circuits are added to the robot.

Don't be scared off a project like this if you are not willing to go the full expansion route with the buffered bus expansion board. The RX81 I/O board can be plugged directly into the ZX81/TS1000 bus just like the printer and 16K RAM are plugged into the back of the computer. If you want to add more boards a simple "Y" or ribbon connector will do the trick but be careful that you don't overload the output capacity of the computer.

If you want to use a Computer Continuum expansion board you already purchased, here is some additional information. Computer Continuum (CC) made two versions of the board, neither of which can be used as I have described without some modification. The earlier version of the board will accept the RX81 I/O board plugged directly into an expansion edge connector (50 pin, .1 inch centers) soldered to the CC board. However, the logic of this earlier version will not work without an additional simple decoder circuit you will have to build. As I explained in Chapter 2, the Sinclair ZX and TS 2040 printers will not work without this decoder either. The newer version of the CC expansion board came with the decoder circuit built into the board, but the expansion pad pinout was reversed so that the RX81 board could not be plugged directly into the CC board. The RX81 board could be plugged onto the bus connection for the 16K RAM but then you would still have to work out another location for the RAM. A ribbon connector could be used, but the compact layout you see on H.E.N.R.Y.'s head could not be accomplished. If you already have a CC expansion board, the easiest way to tell the two versions apart is by the absence of presence of a 74LS27 IC chip located next to the optional LM323 voltage regulator. By the way, I highly recommend using the optional voltage regulator, as it lets you bypass the voltage regulator in the computer and you can kiss overheating problems good-by forever. Anyway, if your CC board has a 74LS27 chip it is the newer version of the board. If no 74LS27 is present, it is the older version.

If you have the older version you can plug the RX81 board directly onto an edge connector on the CC board, as I have done with H.E.N.R.Y. If you need to install the required decoder circuit, just refer to Figure 2-2, schematic and Figure 2-3, legend (Chapter 2) for the required information. Then the 74LS27 can be installed just as it is in the new version. It can be installed in the same spot as in the new version with an IC socket, a few jumper wires and cuts in the circuit board foil. If you do this you essentially have a new version.

If you have the newer version of the Computer Continuum board the decoder is already installed. In order to connect the RX81 board(s), if you don't want to solder them directly to the board, you will either have to buy a ribbon connector (CC sells them) or you will have to build an intermediate connector board.

Table 5-9. Robot control program listing 2.

```
  1 REM <=" J?<=" J▒4 PLOT   =" J▒4
LIST ." TAN Y" PEEK ▒TAN
 20 GOTO 2050
 30 POKE A,0
 40 POKE B,7
 50 LET OUT=USR C
 60 PAUSE 30
 70 POKE A,10
 80 GOSUB E
 90 PAUSE 70
100 GOSUB D
110 IF RND<.5 THEN GOTO 140
120 POKE A,6
130 GOTO 135
140 POKE A,9
150 GOSUB E
160 PAUSE 37
170 GOSUB D
180 GOTO 2210
190 POKE A,0
200 LET OUT=USR C
210 PAUSE 30
220 RETURN
230 LET OUT=USR C
240 RETURN
250 POKE A,0
260 POKE B,7
270 LET OUT=USR C
280 PAUSE 30
290 POKE A,10
300 GOSUB E
310 PAUSE 37
320 GOSUB D
330 POKE A,9
340 GOSUB E
350 PAUSE 5
360 GOSUB D
370 GOTO 2210
380 POKE A,0
390 POKE B,7
400 LET OUT=USR C
410 PAUSE 30
420 POKE A,10
430 GOSUB E
440 PAUSE 37
450 GOSUB D
460 POKE A,6
470 GOSUB E
480 PAUSE 5
490 GOSUB D
500 GOTO 2210
510 STOP
520 REM RANDOM SELECT
530 IF RND<.67 THEN GOTO 2250
540 POKE A,0
550 POKE B,7
560 GOSUB E
570 POKE A,10
580 GOSUB E
590 PAUSE 100
```

Table 5-9 (Continued)

```
 500 POKE A,0
 510 GOSUB E
 520 POKE A,9
 530 GOSUB E
 540 PAUSE 37
 550 POKE A,0
 560 GOSUB E
 570 POKE A,5
 580 GOSUB G
 590 FOR W=1 TO 12
 700 LET IN=USR F
 710 IF IN=4 THEN GOTO 760
 720 IF IN=1 THEN GOTO 760
 730 IF IN=5 THEN GOTO 760
 740 IF W=12 THEN GOTO 780
 750 NEXT W
 760 GOSUB D
 770 GOTO 380
 780 GOSUB D
 790 POKE A,10
 800 GOSUB G
 810 FOR W=1 TO 20
 820 IF W=20 THEN GOTO 840
 830 NEXT W
 840 GOSUB D
 850 POKE A,6
 860 GOSUB E
 870 PAUSE 90
 880 POKE A,0
 890 GOSUB E
 900 POKE A,5
 910 GOSUB G
 920 FOR W=1 TO 12
 930 LET IN=USR F
 940 IF IN=2 THEN GOTO 990
 950 IF IN=1 THEN GOTO 990
 960 IF IN=3 THEN GOTO 990
 970 IF W=12 THEN GOTO 1010
 980 NEXT W
 990 GOSUB D
1000 GOTO 250
1010 GOSUB D
1020 POKE A,10
1030 GOSUB G
1040 FOR W=1 TO 18
1050 IF W=18 THEN GOTO 1070
1060 NEXT W
1070 GOSUB D
1080 POKE A,9
1090 GOSUB E
1100 PAUSE 37
1110 POKE A,0
1120 POKE B,7
1130 GOSUB E
1140 POKE A,32
1150 POKE B,7
1160 GOSUB G
1170 PAUSE 80
1180 GOSUB D
```

Table 5-9 (Continued)

```
1190 POKE A,16
1200 GOSUB E
1210 PAUSE 130
1220 GOSUB D
1230 POKE A,32
1240 GOSUB E
1250 PAUSE 45
1260 GOSUB D
1270 POKE A,144
1280 GOSUB E
1290 POKE A,18
1300 GOSUB G
1310 PAUSE 30
1320 POKE A,154
1330 GOSUB E
1340 PAUSE 20
1350 GOSUB D
1360 POKE A,128
1370 POKE B,7
1380 GOSUB E
1390 PAUSE 70
1400 GOSUB D
1410 FOR W=1 TO 300
1420 LET IN=USR F
1430 IF IN=128 THEN GOTO 1460
1440 IF W=300 THEN GOTO 1460
1450 NEXT W
1460 POKE A,64
1470 POKE B,7
1480 GOSUB E
1490 POKE A,96
1500 POKE B,6
1510 GOSUB E
1520 PAUSE 50
1530 POKE A,33
1540 GOSUB G
1550 PAUSE 10
1560 POKE A,97
1570 GOSUB E
1580 PAUSE 5
1590 GOSUB D
1600 POKE A,70
1610 POKE B,7
1620 GOSUB G
1630 PAUSE 220
1640 POKE A,64
1650 GOSUB E
1660 POKE A,69
1670 GOSUB G
1680 PAUSE 280
1690 POKE A,64
1700 GOSUB E
1710 POKE A,144
1720 POKE B,6
1730 GOSUB E
1740 POKE A,18
1750 GOSUB G
1760 PAUSE 10
1770 POKE A,146
1780 GOSUB E
```

Table 5-9(Continued)

```
1790 PAUSE 20
1800 GOSUB D
1810 LET IN=USR F
1820 IF IN=0 THEN GOTO 1840
1830 PAUSE 400
1840 POKE A,128
1850 POKE B,7
1860 GOSUB E
1870 POKE A,96
1880 POKE B,6
1890 GOSUB E
1900 PAUSE 100
1910 POKE A,33
1920 GOSUB G
1930 PAUSE 10
1940 POKE A,101
1950 GOSUB E
1960 GOSUB D
1970 POKE A,64
1980 POKE B,7
1990 GOSUB E
2000 PAUSE 40
2010 POKE A,6
2020 GOSUB G
2030 PAUSE 230
2040 GOSUB D
2050 GOTO 2210
2060 LET A=16534
2070 LET B=16536
2080 LET C=16533
2090 LET D=190
2100 LET E=200
2110 LET F=16514
2120 LET G=230
2130 POKE A,0
2140 POKE B,7
2150 LET OUT=USR C
2160 POKE A,0
2170 POKE B,6
2180 LET OUT=USR C
2190 PAUSE 400
2200 LET RN=1
2210 POKE A,5
2220 POKE B,7
2230 LET OUT=USR C
2240 LET TN=0
2250 LET TN=TN+1
2260 LET IN=USR F
2270 IF IN=1 THEN GOTO 30
2280 IF IN=2 THEN GOTO 250
2290 IF IN=4 THEN GOTO 380
2300 IF IN=3 THEN GOTO 250
2310 IF IN=5 THEN GOTO 380
2320 IF IN=7 THEN GOTO 30
2330 IF TN=20 THEN GOTO 530
2340 GOTO 2250
```

Table 5-10. Complete 13K robot control listing used to win the Golden Droid Award.

```
   1 REM          ?ABS RND??INKEY$?
 LIST INKEY$ COS COPY COPY W S P
I FAST  RNDQ  74 IF   COPY LPRIN
T : RETURN GOSUB ? PAUSE  RNDG G
OSUB ? IF  RND? RETURN RNDABS  R
NDQ COS GOSUB ? IF  RNDI ) >= COP
Y .? COS GOSUB ? PAUSE LEN RND?
RNDEVRND FAST LN ??LN E Y E£RND
FOR E(RND  GOSUB ?) COPY   GOSUB
 ?S XM CLS INKEY$; VAL SPI?  Q S
74 IF LN  RNDSPI?  PI???  FAST A
 ?; FOR LPRINT ?F???  Y  K REM AT
 $C,VAL U CLS INKEY$ C""Y8 E£RND)
? ;?  2 LN P??  INKEY$U RND  RND 5
PIVAL  X4 UNPLOT ??5PI?????Y  AC
S ,ACS .X4 SAVE ?  SGN 1C STR$ /
 CONT LPRINT YRND COS LN F?TAN E
VRND FAST E RND GOSUB ? RND: RET
URN GOSUB ? IF  INKEY$,? 4 POKE
? ASN LLIST INKEY$F/ GOTO LN ??Y
 COPY M CLEAR INKEY$M UNPLOT INK
EY$LN  RND5INKEY$PI  COPY  VAL )
 PI:  RND, K  JW K  Y COPY ??< 4 L
OAD ?AT  K ?M CLEAR INKEY$?M UNP
LOT INKEY$ Y  4USR U CLEAR INKEY
$?U RND Y COPY S U UNPLOT INKEY$
W?   LPRINT YRND COS LN F?TAN
5INKEY$PI   Q 7""? COPY  ?TAN   <
?++->£""
.CA3.?
 ? H A / 7 6 B F N V U K ? ? ?
? ? ? ? ? I 6 A / ( ? = ( £

<=  J?<=  J 4 PLOT <=  J 4 LIST   T
AN Y  PEEK  TAN    :  <=  J NEW  ?
<=  J NEW   C SAVE <=  J NEW   C L
IST <=  J NEW   C CONT ?( FAST $4
 STEP TAN Y  PEEK  TAN
```

Table 5-10 (Continued)

```
  2 REM  SPEECH RECOGNITION
    COPYRIGHT 1983
    BY BRAD BENNETT

    ROBOT CONTROL
    COPYRIGHT 1984
    BRUCE C. TAYLOR

  4 DIM W$(8,10)
  5 GOTO 5000
 10 POKE 17861,79
 11 LET J=0
 12 POKE 32763,128
 20 GOTO 9000
 99 REM A1
100 SLOW
101 POKE A,0
102 POKE B,7
105 LET OUT=USR C
106 GOSUB 9900
114 POKE A,10
115 GOSUB E
120 PAUSE 70
125 GOSUB D
127 IF RND<.5 THEN GOTO 132
130 POKE A,6
131 GOTO 135
134 POKE A,9
135 GOSUB E
140 PAUSE 37
145 GOSUB D
150 GOTO 9150
179 REM C1
180 POKE A,0
185 LET OUT=USR C
186 PAUSE 30
190 RETURN
194 REM C2
195 LET OUT=USR C
197 RETURN
199 REM A2
200 SLOW
201 POKE A,0
202 POKE B,7
205 LET OUT=USR C
206 GOSUB 9900
207 IF TN<=15 THEN LET J=J+3
208 LET H=2
209 IF J=14 THEN GOTO 5200
210 POKE A,10
215 GOSUB E
220 PAUSE 37
225 GOSUB D
230 POKE A,9
235 GOSUB E
240 PAUSE 5
245 GOSUB D
250 GOTO 9150
```

Table 5-10 (Continued)

```
299 REM A3
300 SLOW
301 POKE A,0
302 POKE B,7
305 LET OUT=USR C
306 GOSUB 9900
307 IF TN<=15 THEN LET J=J+4
308 LET H=3
309 IF J=14 THEN GOTO 5200
310 POKE A,10
315 GOSUB E
320 PAUSE 37
325 GOSUB D
330 POKE A,5
335 GOSUB E
340 PAUSE 5
345 GOSUB D
350 GOTO 9150
399 REM V1
400 SLOW
401 GOSUB D
410 FOR K=0 TO 143
415 POKE M,0
420 POKE N,K
430 PAUSE 70
435 IF K=143 THEN GOTO 485
440 IF K>130 THEN NEXT K
450 POKE M,1
460 POKE N,K
470 PAUSE 70
480 NEXT K
485 POKE A,0
490 GOTO 9150
499 REM D1
500 SLOW
501 POKE A,37
510 LET OUT=USR C
515 PAUSE 60
525 POKE A,5
530 GOSUB E
532 POKE A,21
533 GOSUB E
535 PAUSE 90
545 POKE A,5
550 GOSUB E
552 POKE A,37
553 GOSUB E
555 PAUSE 31
560 GOTO 9150
599 REM D2
600 SLOW
601 POKE A,0
605 LET OUT=USR C
606 PAUSE 30
610 POKE A,18
612 POKE B,5
615 GOSUB E
620 PAUSE 60
625 GOSUB D
626 POKE A,64
```

Table 5-10 (Continued)

```
627 POKE B,7
628 GOSUB E
630 POKE A,32
632 POKE B,6
635 GOSUB E
640 PAUSE 10
645 GOSUB D
650 POKE A,16
655 GOSUB E
660 PAUSE 35
665 GOSUB D
670 POKE A,32
675 GOSUB E
677 PAUSE 40
680 POKE A,33
685 GOSUB E
690 PAUSE 25
695 GOSUB D
696 POKE A,0
697 POKE B,7
698 GOSUB E
699 GOTO 9150
700 REM V21
701 SLOW
702 POKE A,0
703 POKE B,7
704 LET OUT=USR C
705 FOR I=0 TO 1
710 POKE M,1
720 POKE N,4
730 PAUSE 50
740 POKE M,0
750 GOSUB 8400
760 NEXT I
770 POKE M,1
780 POKE N,63
790 PAUSE 40
800 POKE M,0
810 POKE N,96
820 PAUSE 20
830 POKE N,94
840 PAUSE 20
850 POKE M,1
860 POKE N,118
870 PAUSE 25
880 POKE N,100
890 PAUSE 70
900 POKE N,124
910 PAUSE 30
920 POKE M,0
930 POKE N,2
940 PAUSE 20
950 POKE M,1
960 POKE N,63
970 PAUSE 70
980 POKE N,3
990 PAUSE 35
1000 POKE M,0
1010 POKE N,96
1020 PAUSE 25
1030 POKE N,115
```

Table 5-10 (Continued)

```
1040 PAUSE [illegible]
1045 FOR I [illegible] TO 1
1050 POKE M,40
1060 PAUSE [illegible]
1070 POKE [illegible]0
1080 PAUSE [illegible]
1090 POKE M,1
1100 POKE N,95
1110 PAUSE 25
1120 POKE N,33
1130 PAUSE 20
1140 POKE N,35
1150 PAUSE 35
1160 POKE N,105
1170 PAUSE 25
1180 POKE N,85
1190 PAUSE 35
1200 POKE M,0
1210 POKE N,4
1220 PAUSE 20
1230 POKE M,1
1240 POKE N,7
1250 PAUSE 70
1260 POKE M,0
1270 POKE N,76
1280 PAUSE 35
1290 POKE N,2
1300 PAUSE 35
1310 POKE M,1
1320 POKE N,63
1330 PAUSE 50
1340 POKE M,0
1350 POKE N,52
1360 PAUSE 20
1370 POKE N,49
1380 PAUSE 20
1390 POKE M,1
1400 POKE N,80
1410 PAUSE 25
1420 POKE N,101
1430 PAUSE 60
1433 IF I=1 THEN POKE M,0
1435 IF I=1 THEN GOTO 9150
1440 POKE N,8
1450 PAUSE 35
1460 POKE N,8
1470 PAUSE 40
1480 POKE M,0
1490 NEXT I
2040 STOP
2100 REM D3
2101 SLOW
2105 POKE N,65
2110 IF RND<.67 THEN GOTO 9185
2115 POKE N,66
2120 POKE A,0
2122 POKE B,7
2124 GOSUB E
2129 REM V2
2130 POKE N,38
2132 PAUSE 18
```

Table 5-10 (Continued)

```
2134 POKE N,143
2136 PAUSE 4
2138 POKE N,96
2140 LET H=1
2145 PAUSE 60
2150 GOTO 5200
2160 POKE A,10
2170 GOSUB E
2180 PAUSE 100
2190 POKE A,0
2194 GOSUB E
2196 GOSUB 8000
2200 POKE A,9
2210 GOSUB E
2220 PAUSE 37
2230 POKE A,0
2234 GOSUB E
2240 POKE A,5
2250 GOSUB G
2260 FOR W=1 TO 12
2265 LET IN=USR F
2270 IF IN=4 THEN GOTO 2320
2280 IF IN=1 THEN GOTO 2320
2290 IF IN=5 THEN GOTO 2320
2300 IF W=12 THEN GOTO 2340
2310 NEXT W
2320 GOSUB D
2325 GOSUB 8300
2330 GOTO 310
2340 GOSUB D
2350 POKE A,10
2370 GOSUB G
2380 FOR W=1 TO 20
2390 IF W=20 THEN GOTO 2410
2400 NEXT W
2410 GOSUB D
2420 POKE A,6
2430 GOSUB E
2440 PAUSE 90
2450 POKE A,0
2470 GOSUB E
2480 POKE A,5
2500 GOSUB G
2510 FOR W=1 TO 12
2515 LET IN=USR F
2520 IF IN=2 THEN GOTO 2570
2530 IF IN=1 THEN GOTO 2570
2540 IF IN=3 THEN GOTO 2570
2550 IF W=12 THEN GOTO 2590
2560 NEXT W
2570 GOSUB D
2575 GOSUB 8300
2580 GOTO 210
2590 GOSUB D
2600 POKE A,10
2620 GOSUB G
2630 FOR W=1 TO 18
2640 IF W=18 THEN GOTO 2660
2650 NEXT W
2660 GOSUB D
```

Table 5-10 (Continued)

```
2670 POKE A,9
2680 GOSUB E
2690 PAUSE 37
2700 POKE A,0
2710 POKE B,7
2720 GOSUB E
2795 POKE A,16
2796 POKE B,6
2797 GOSUB E
2800 POKE A,18
2820 GOSUB E
2830 PAUSE 70
2840 GOSUB D
2845 POKE A,128
2847 GOSUB E
2850 POKE A,135
2870 GOSUB E
2880 PAUSE 50
2890 GOSUB D
2900 GOSUB 2920
2910 GOSUB 2920
2915 GOTO 3020
2920 POKE A,128
2930 POKE B,7
2940 GOSUB E
2950 PAUSE 75
2960 GOSUB D
2970 POKE A,64
2980 GOSUB E
2990 PAUSE 70
3000 GOSUB D
3010 RETURN
3020 POKE A,32
3030 POKE B,7
3040 GOSUB G
3045 POKE N,55
3050 PAUSE 80
3055 POKE N,66
3060 GOSUB D
3070 POKE A,16
3080 GOSUB E
3085 POKE N,66
3090 PAUSE 130
3095 POKE N,55
3100 GOSUB D
3110 POKE A,32
3120 GOSUB E
3125 POKE N,55
3130 PAUSE 45
3135 POKE N,66
3140 GOSUB D
3145 POKE A,32
3146 POKE B,6
3147 GOSUB E
3148 PAUSE 30
3150 POKE A,33
3170 GOSUB E
3180 PAUSE 10
3190 GOSUB D
3195 POKE A,64
```

Table 5-10 (Continued)

```
3197 GOSUB E
3198 PAUSE 30
3200 POKE A,65
3210 GOSUB E
3220 PAUSE 15
3230 GOSUB D
3235 POKE A,144
3237 GOSUB E
3240 POKE A,18
3250 GOSUB G
3260 PAUSE 30
3270 POKE A,154
3280 GOSUB E
3290 PAUSE 20
3300 GOSUB D
3310 POKE A,128
3320 POKE B,7
3330 GOSUB E
3340 PAUSE 70
3350 GOSUB D
3354 REM V3
3355 POKE N,120
3356 PAUSE 60
3360 FOR W=1 TO 300
3365 LET IN=USR F
3370 IF IN=128 THEN GOTO 3400
3380 IF W=300 THEN GOTO 3400
3390 NEXT W
3400 REM V16
3402 POKE M,1
3404 POKE N,116
3406 PAUSE 30
3408 POKE M,0
3410 POKE N,52
3412 PAUSE 60
3414 POKE M,0
3416 POKE A,64
3418 POKE B,7
3420 GOSUB E
3425 POKE A,96
3426 POKE B,6
3427 GOSUB E
3428 PAUSE 50
3430 POKE A,33
3450 GOSUB G
3460 PAUSE 10
3470 POKE A,97
3480 GOSUB E
3490 PAUSE 5
3500 GOSUB D
3510 POKE A,70
3520 POKE B,7
3530 GOSUB G
3540 PAUSE 220
3550 POKE A,64
3560 GOSUB E
3570 POKE A,69
3580 GOSUB G
3590 PAUSE 280
3600 POKE A,64
3610 GOSUB E
```

Table 5-10 (Continued)

```
3615 POKE A,144
3616 POKE B,6
3617 GOSUB E
3620 POKE A,18
3640 GOSUB G
3650 PAUSE 10
3660 POKE A,146
3670 GOSUB E
3680 PAUSE 20
3690 GOSUB D
3695 LET IN=USR F
3696 IF IN=0 THEN GOTO 3700
3697 PAUSE 400
3700 POKE N,65
3701 PAUSE 20
3702 POKE N,66
3703 PAUSE 20
3704 POKE N,65
3705 POKE A,128
3710 POKE B,7
3720 GOSUB E
3725 POKE A,98
3726 POKE B,6
3727 GOSUB E
3728 PAUSE 100
3730 POKE A,33
3750 GOSUB G
3760 PAUSE 10
3770 POKE A,101
3780 GOSUB E
3800 GOSUB D
3810 POKE A,64
3820 POKE B,7
3830 GOSUB E
3832 PAUSE 40
3840 POKE A,6
3850 GOSUB G
3860 PAUSE 230
3870 GOSUB D
3875 POKE A,2
3876 POKE B,6
3877 LET OUT=USR C
3878 PAUSE 3
3879 GOSUB D
3880 POKE A,8
3881 LET OUT=USR C
3882 PAUSE 2
3883 GOSUB D
3990 GOTO 9150
4999 STOP
5000 REM R1
5010 FAST
5020 RAND USR 16526
5030 FOR Y=1 TO 8
5040 PRINT AT 10,10;"FILE NUMBER
 ";Y
5050 PRINT AT 12,10;"ENTER WORD"
5060 INPUT W$(Y)
5070 POKE 16529,Y
5080 RAND USR 16520
```

Table 5-10 (Continued)

```
5090 NEXT Y
5100 CLS
5110 FOR Y=1 TO 8
5120 PRINT AT (5+Y),10;Y;".  ";W$
(Y)
5130 NEXT Y
5140 PRINT AT (7+Y),1;"""GOTO 10
"" TO RUN ROBOT PROGRAM"
5150 SLOW
5160 STOP
5200 REM R2
5210 REM V4
5220 POKE N,59
5230 PAUSE 15
5240 POKE N,67
5250 PAUSE 30
5260 POKE N,115
5270 PAUSE 30
5280 POKE N,120
5290 PAUSE 30
5300 FAST
5305 LET I=0
5310 LET X=USR 16523
5320 IF H=1 THEN GOTO 5400
5330 IF H=2 OR H=3 THEN GOTO 550
0
5340 IF H=4 THEN GOTO 5645
5350 SLOW
5360 RETURN
5400 REM R3
5410 IF X=1 THEN GOTO 5460
5420 IF X=2 THEN GOTO 5480
5425 IF I=1 AND X=0 THEN GOTO 91
50
5430 IF X=0 THEN GOSUB 6020
5440 IF X=0 OR X>2 THEN GOSUB 58
00
5450 GOTO 5310
5460 GOSUB 5880
5465 GOTO 5600
5470 GOTO 9150
5480 GOSUB 5960
5485 GOTO 5600
5490 GOTO 2160
5500 REM R4
5510 IF X=3 THEN GOTO 5564
5520 IF X=4 THEN GOTO 5570
5530 IF X=5 THEN GOTO 5576
5540 IF I=1 AND X=0 THEN GOTO 55
80
5550 IF X=0 THEN GOSUB 6020
5560 IF X=0 OR X<3 OR X>5 THEN G
OSUB 5800
5562 GOTO 5310
5564 SLOW
5566 GOSUB 6230
5567 GOTO 5600
5568 GOTO 210
5570 SLOW
5572 GOSUB 6270
5573 GOTO 5600
```

Table 5-10 (Continued)

```
5574 GOTO 310
5575 SLOW
5577 GOSUB 5310
5578 GOTO 5600
5579 GOTO 114
5580 SLOW
5582 IF H=2 THEN GOTO 210
5584 IF H=3 THEN GOTO 310
5600 REM R5
5610 LET V=X
5620 LET R=H
5630 LET H=4
5640 GOTO 5300
5645 IF X=6 THEN GOTO 5790
5650 IF X=7 THEN GOTO 5700
5660 IF X=8 THEN GOTO 5750
5665 IF X<7 OR X>8 THEN GOSUB 58
00
5670 IF I=1 AND X=0 THEN GOTO 55
80
5680 IF X=0 THEN GOSUB 6020
5690 IF X=0 THEN GOSUB 5800
5700 REM R6
5710 SLOW
5720 IF V=3 THEN GOTO 210
5730 IF V=4 THEN GOTO 310
5740 IF V=5 THEN GOTO 114
5742 IF V=1 THEN GOTO 9150
5744 IF V=2 THEN GOTO 2160
5750 REM R7
5760 LET H=R
5770 SLOW
5780 GOTO 5200
5790 GOTO 400
5800 REM V5
5810 SLOW
5820 POKE N,58
5830 PAUSE 30
5840 POKE N,120
5850 PAUSE 30
5860 FAST
5870 RETURN
5880 REM V6
5890 SLOW
5900 POKE N,46
5910 PAUSE 20
5920 POKE N,42
5930 PAUSE 20
5940 FAST
5950 RETURN
5960 REM V7
5970 SLOW
5980 POKE N,88
5990 PAUSE 30
6010 RETURN
6020 REM V8
6030 SLOW
6040 POKE N,31
6050 PAUSE 30
6060 POKE N,63
```

Table 5-10 (Continued)

```
6070 PAUSE 50
6080 POKE N,40
6090 PAUSE 30
6100 POKE N,140
6110 PAUSE 30
6120 POKE N,58
6130 PAUSE 70
6140 LET I=I+1
6150 FAST
6160 RETURN
6170 REM V9
6180 POKE N,40
6190 PAUSE 30
6200 POKE N,86
6210 PAUSE 30
6220 RETURN
6230 REM V10
6240 GOSUB 6170
6250 POKE N,128
6260 RETURN
6270 REM V11
6280 GOSUB 6170
6290 POKE N,99
6300 RETURN
6310 REM V12
6320 GOSUB 6170
6330 POKE N,109
6340 RETURN
7999 REM V13
8000 POKE N,40
8010 PAUSE 15
8020 POKE N,73
8030 PAUSE 25
8040 POKE N,2
8050 PAUSE 10
8060 POKE N,138
8070 PAUSE 20
8080 POKE N,128
8090 PAUSE 30
8100 POKE N,60
8110 PAUSE 30
8120 POKE N,2
8130 PAUSE 10
8140 POKE N,138
8150 PAUSE 20
8160 POKE N,99
8170 PAUSE 25
8180 POKE N,4
8190 PAUSE 20
8200 POKE N,132
8210 PAUSE 60
8220 RETURN
8299 REM V14
8300 POKE N,31
8310 PAUSE 25
8320 POKE N,132
8330 PAUSE 50
8340 POKE N,2
8350 PAUSE 25
8360 POKE N,111
8370 PAUSE 60
```

Table 5-10 (Continued)

```
8360 RETURN
8399 REM V15
8400 POKE N,66
8410 PAUSE 30
8420 POKE N,65
8430 PAUSE 30
8450 RETURN
9000 LET A=17878
9010 LET B=17880
9020 LET C=17877
9030 LET D=180
9040 LET E=185
9050 LET F=17858
9060 LET G=195
9065 LET N=32760
9067 LET M=32762
9069 REM C3
9070 POKE A,0
9080 POKE B,7
9090 LET OUT=USR C
9100 POKE A,0
9110 POKE B,6
9120 LET OUT=USR C
9121 PAUSE 200
9122 GOSUB 8400
9123 GOSUB 8400
9124 GOSUB 8400
9125 GOSUB 9700
9126 GOSUB 8400
9127 GOSUB 9500
9128 GOSUB 8400
9129 REM V17
9130 POKE N,40
9131 PAUSE 30
9132 POKE N,135
9133 PAUSE 60
9134 POKE N,94
9135 PAUSE 50
9136 POKE N,2
9137 PAUSE 60
9138 POKE N,130
9139 PAUSE 40
9140 POKE N,129
9141 PAUSE 40
9142 PAUSE 100
9143 LET RN=1
9149 REM C4
9150 POKE A,5
9160 POKE B,7
9170 LET OUT=USR C
9180 LET TN=0
9183 IF J=14 THEN LET J=0
9185 FAST
9190 LET TN=TN+1
9195 IF TN>15 THEN LET J=0
9200 LET IN=USR F
9210 IF IN=1 THEN GOTO 100
9220 IF IN=2 THEN GOTO 200
9230 IF IN=4 THEN GOTO 300
9234 REM "O"
9235 IF TN=10 THEN POKE N,46
```

Table 5-10 (Continued)

```
9240 IF IN=8 THEN GOTO 400
9250 IF IN=16 THEN GOTO 500
9260 IF IN=32 THEN GOTO 600
9265 IF IN=64 THEN GOTO 700
9270 IF IN=3 THEN GOTO 200
9280 IF IN=5 THEN GOTO 300
9290 IF IN=7 THEN GOTO 100
9294 REM "K"
9295 IF TN=15 THEN POKE N,42
9300 IF TN=100 THEN GOTO 2100
9310 GOTO 9185
9499 REM V18
9500 POKE N,32
9510 PAUSE 50
9520 POKE N,57
9530 PAUSE 24
9540 POKE N,55
9550 PAUSE 40
9560 POKE N,26
9570 PAUSE 24
9580 POKE N,1
9590 PAUSE 50
9600 POKE N,75
9610 PAUSE 30
9620 POKE N,129
9630 PAUSE 20
9640 POKE N,106
9650 PAUSE 12
9660 POKE N,71
9670 PAUSE 60
9680 RETURN
9699 REM V19
9700 POKE M,1
9702 POKE N,57
9704 PAUSE 40
9706 POKE M,0
9716 POKE N,40
9718 PAUSE 30
9720 POKE N,59
9730 PAUSE 10
9740 POKE N,71
9750 PAUSE 30
9760 POKE N,39
9770 PAUSE 30
9780 POKE N,36
9790 PAUSE 30
9800 POKE N,45
9810 PAUSE 30
9820 POKE N,49
9830 PAUSE 30
9840 POKE N,56
9850 PAUSE 60
9860 RETURN
9899 REM "OH, OH"
9900 POKE N,46
9901 PAUSE 40
9902 POKE N,46
9903 PAUSE 40
9904 RETURN
```

THIS PAGE INTENTIONALLY BLANK

Table 5-11
ROBOT SOFTWARE DOCUMENTATION
(Complete 13K demonstration routine)

	Avoidance routines	(line #)
A1	Forward switch avoidance sequence	(100)
A2	Left switch avoidance or right turn sequence	(200)
A3	Right switch avoidance or left turn sequence	(300)
	Control subroutines	
C1	Clear or stop sequence with pause (GOSUB D = 180) OUT activate sequences with pause (GOSUB E = 185)	
C2	OUT activate, no pause (GOSUB G = 195)	
C3	Clears outputs on both output boards	9070-9120
C4	Activates forward motion and scans bumper switches for inputs	9150-9310
	Voice responds "OK" if scanning is taking place	9235-9295
	After 100 scans of all switch combinations (10 seconds elapsed time) goes to random number subroutine for possible stop to demonstrate arms	9300
	Demonstration routines	
D1	Body rotate while traveling forward sequence	(500)
D2	Left arm sequence with wheel motors stopped	(600)
D3	Random select to either continue forward or begin arm demonstration sequence	
	Continues forward 2/3 of the time	2110
	Stops, reverses, then stops	2120-2194
	Turns 45 degrees to right	2200-2234
	Checks for right/front or combination of bumper switches while traveling forward	2240-2320
	Resumes forward if switch is activated	2330
	Backs up to original location	2340-2400
	Turns 90 degrees to left	2410-2470

Table 5-11 (Continued)

Checks for left/front or combination of bumper switches while traveling forward	2480-2570
Resumes forward if switch is activated	2580
Backs up to original location	2590-2650
Turns 45 degrees to right (where originally stopped)	2660-2720
Left elbow down 50% Left shoulder up 80 degrees	2795-2840
Right elbow down 95% Right shoulder up 90 degrees	2845-2890
Right hand opens and closes two times	2900-3010
Upper body rotates 45 degrees to right (CW)	3020-3060
Upper body rotates 90 degrees to left (CCW)	3070-3100
Upper body rotates 45 degrees to right (CW) to original position	3110-3140
Left elbow up 25% Left shoulder down 80 degrees	3145-3190
Right elbow up 99% Right shoulder down 85 degrees	3195-3230
Left elbow down 50% Left shoulder up 50 degrees (to 75% position) Right elbow down 80% Right shoulder up 70 degrees	3235-3300
Right hand opens	3310-3350
Right hand closes Left elbow up/left shoulder down (to rest) Right elbow up (to limit)	3360-3500
Turn left 180 degrees/forward 2 feet	3510-3610
Left elbow down 50%/left shoulder up 45 degrees Right elbow down 80%	3615-3690
Right hand opens Left elbow up to maximum/left shoulder down 45 degrees Right elbow up to maximum/right shoulder down 45 degrees	3695-3800

Table 5-11 (Continued)

	Right hand closes Turn left 180 degrees (back to location/position where demonstration began)	3810-3870
	Left shoulder up to rest position Right shoulder up to rest position	3875-3883
	Voice Recognition Routines	
R1	Create voice recognition file	(5000)
R2	Voice recognition routine	(5200)
R3	"continue" forward or "arms" demonstration recognition subroutine	(5400)
R4	Move "right, left, back" recognition subroutine	(5500)
R5	Confirm/deny voice command with "stop, yes, no" recognition subroutine	(5600)
R6	Voice command verified with "yes"	(5700)
R7	Not verified with "yes", return and try again for proper command recognition	(5750)
	Voice subroutines	
V1	Pause and recite entire vocabulary	(400)
V2	"gee whiz"	(2130)
V3	"please"	(3355)
V4	"amp on please"	(5211)
V5	"again please"	(5800)
V6	"OK" response to continue forward command	(5880)
V7	"great" response to arms demonstration command	(5960)
V8	"zero case, I try again" response to no voice input	(6020)
V9	"I go"	(6170)
V10	"right"	(6230)
V11	"left"	(6270)
V12	"minus"	(6310)

Table 5-11 (Continued)

V13	"I check to the right and to the left for space"	(8000)
V14	"zero space, too near"	(8300)
V15	"bop, beep"	(8400)
V16	"thank you"	(3400)
V17	"I start in two seconds"	(9130)
V18	"A ZX81 controls me"	(9500)
V19	"I am H.E.N.R.Y."	(9700)
V20	"oh, oh"	(9900)
V21	"Alert, alert. Intruder is in this room. Warning to intruder. Alarm is on. I have reached emergency operator for assistance. Danger to intruder, you are not safe. I have reached emergency operator for assistance. Danger to intruder, you are not safe."	(700)

Variables

A = 17878 (output selected - POKE location)

B = 17880 (select I/O board - POKE location)

C = 17877 (OUT USR - POKE location)

D = 180 (clear or stop sequence - with pause)

E = 185 (OUT activate sequence - with pause)

F = 17858 (IN USR - POKE location)

G = 195 (OUT activate - no pause)

M = 32762 (voice ROM select - POKE location)

N = 32760 (voice word select - POKE location)

W FOR TO

OUT

IN

TN counter

Y FOR TO

Table 5-11 (Continued)

W$	word
H	voice recognition subroutine
I	counter for recognition abort
J	counter for corner trap
K	FOR TO
V	verify value
R	return value

SPECIAL INSTRUCTIONS

"RUN" to reload voice recognition files

"GOTO 10" to run with voice recognition files already loaded

One solution is shown in Figure 5-17. The pinouts of the .1 inch center, edge connectors are jumped to the correct fingers of the .156 inch center, 4 x 5 inch PC board.

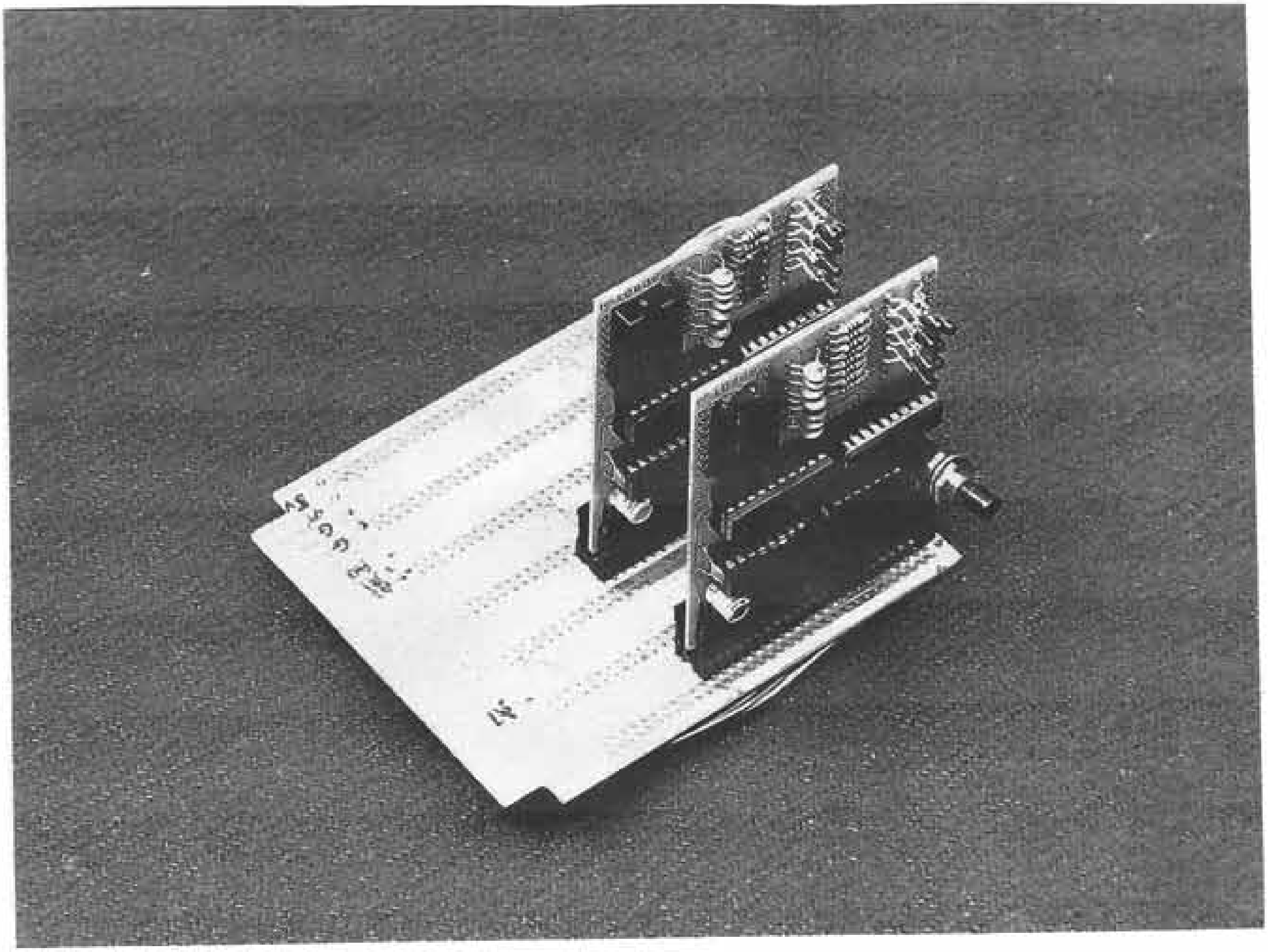
The software listed and documented in Table 5-10 and Table 5-11 includes the instructions required to operate the National Semiconductor digital voice board project described in Chapter 9. The software also includes the voice recognition routines by Brad Bennett, available from G. Russell Electronics, referred to in Chapter 10.

Future expansion plans for H.E.N.R.Y. include measurement of robot movements and feedback of this information into the computer memory so that the robot can leard as it moves about and functions. Details on this circuitry and software are found in the Optical Encoder chapter, Chapter 8.

Another expansion is the addition of the Polaorid ultrasonic rangefinder OEM kit described in Chapter 11.

And finally, even if you go with a full blown robot project, remember you do not have to sacrifice your ZX81 or TS1000 to dedicated robot control. With easy keyboard access, as in H.E.N.R.Y., or an external keyboard, you still have an operating computer to run any kind of program. I have also added a video plug so that the much clearer video display can be used instead of a TV set. This modification is detailed in Chapter 11.

Figure 5-17. RX81 to Expansion Board connector required for the last version of the Expansion Board sold by Computer Continuum.



Chapter 6
HOME CONTROL ON A BUDGET

By adapting the same circuits described in the previous chapter for robot control and combining them with an inexpensive wireless home controller, you can operate virtually any electrical device in your home with output from your ZX81/TS1000 or TS2068. The control system adapted is the BSR X-10 available through several sources including Radio Shack, Heath and Sears. The control unit is available from the Heathkit catalog (it is not a kit) for about $35.00. The control modules, which are connected to lamps, appliances or wall switches, cost from $17.00 to $18.00 and operate by recieving a signal transmitted over the house power wiring, requiring no direct hookup between the computer and the appliance. A complete parts list is provided in Table 6-1. I will first describe the modification required for the BSR X-10 controller.

HARDWARE ASSEMBLY

Remove the screw from the deep hole in the center of the bottom of the X-10 case. The two halves of the case will then pull apart. Now remove the circuit from the top half, the part with the 28 pin IC on it, by first removing the five screws. Be careful to keep the keyboard upside down (IC upright) because when you remove the circuit board there will be nothing to keep the key pad buttons from falling out. After the screws are removed, carefully pry the circuit board out and turn it over,

---

Table 6-1
HOME CONTROL PARTS LIST

(1) BSR X-10 Control System (Heathkit # GDP-1510 or DAK # 9775)
(1) BSR Lamp Module (Heathkit # GDP-1512 or DAK # 9779)
(1) BSR Appliance Module (Heathkit # GDP-1514 or DAK # 9781)
(1) ULN2003 Transistor Darlington array (MOUSER # 511-L203B)
(7) Magnacraft 5 volt coil DIP relays (KNAPCO # 171DIP262)
(1) 16 pin single ended DIP jumper (DIGI-KEY # R112-6)
(1) 16 pin double ended DIP jumper (DIGI-KEY # R116-18)
(1) PC board, Hobby Board (DIGI-KEY # K160-ND)
(1) Buffered Buss Expansion Board, bare board with parts list (Budget Robotics & Computing)
(1) RX81 I/O board, bare board with parts list (Budget Robotics & Computing)
Hookup wire

exposing the foil circuit side of the board (Figure 6-1). The total IC pinout to the keypad switches is listed in Table 6-2. Prepare eleven hookup wires each about 6 inches long. Solder one end of each to an IC pin circuit foil as listed in Figure 6-2 and pictured in Figure 6-1. After the wires are soldered, push them flat against the board and bend them over the edge of the board as shown in Figure 6-1. Then push the board back into the top half of the case. There will just be enough room for the wires between the case side and the board edge (Figure 6-3) if all wires are lined up and none overlap each other. Be careful that you don't dump out the key pad switch buttons. Keep the top half of the case upside down until the board is all the way back into the case top. Now that you see that it fits (you should have formed the wires against the edge of the board) remove the circuit board again. Next, cut a small rectangular hole in the side of the case, or drill some holes so that a 16 pin DIP socket can be attached later to the outside of the case with the hookup wires from the inside soldered to it. The jumper pinout is listed in Table 6-3. Now pull the wires through the hole in the case as you replace the circuit board into the top half of the case, for the final time (Figure 6-3).

Next, cut the hookup wires about an inch long outside the hole in the case and solder them to the DIP socket (Table 6-3 and Figure 6-4). Then glue or otherwise fasten the DIP socket to the outside of the case. You have now completed the modification to the BSR X-10. After the rest of the project is finished you will be able to plug a DIP jumper cable into the socket on the X-10 (Figure 6-5).

Now build or use the same RX81 circuits/boards as were described in the previous chapter for robot control. The RX81 outputs (D0-6) will be connected to transistor switch and relay circuits, similar to but a little simpler than the robot circuits. Basically this control circuit omits the large amperage 5 volt coil relays required in the robot control project. This is because the home control circuits are required to only switch very low power circuits (the BSR keypad) and not the high current robot motors. Also keep in mind that other computer output control circuits can be easily adapted to this design. One such example is BYTE-BACK's BB-1 control module with built in relays.

The circuit I will describe uses the RX81 board to drive a Darlington transistor array, activating 5 volt 14 pin DIP relays which in turn switch the BSR X-10. The transistor array and DIP relay pinouts are pictured in Figure 6-6.

As with the robot circuit I plugged the RX81 boards into the buffered bus Expansion Board, and will not repeat the description of that hookup again. Remember also that this home control circuit can be built without the Expansion Board, by plugging the RX81 I/O board directly into the computer.

The only additional PC board for this project is pictured in Figures 6-7 and 6-8. Again I used the OK Hobby Board (the Radio-Shack equivalent will work equally well) and plugged it into the expansion board with electrical interconnection required only for +5 volts and ground. Figure 6-7 shows a wire-up with the transistor array at the center right (socket below it is for a second transistor array); five adjacent DIP

Table 6-2
BSR X-10, Control Switch IC Pinout

SWITCH	CONTROL IC PINS	
1	1	17
2	17	28
3	1	20
4	20	28
5	1	18
6	18	28
7	1	19
8	19	28
9	1	16
10	16	28
11	1	27
12	27	28
13	1	21
14	21	28
15	1	26
16	26	28
ON	20	25
OFF	25	27
BRIGHT	19	25
DIM	25	26
ALL LIGHTS ON	18	25
ALL OFF	21	25

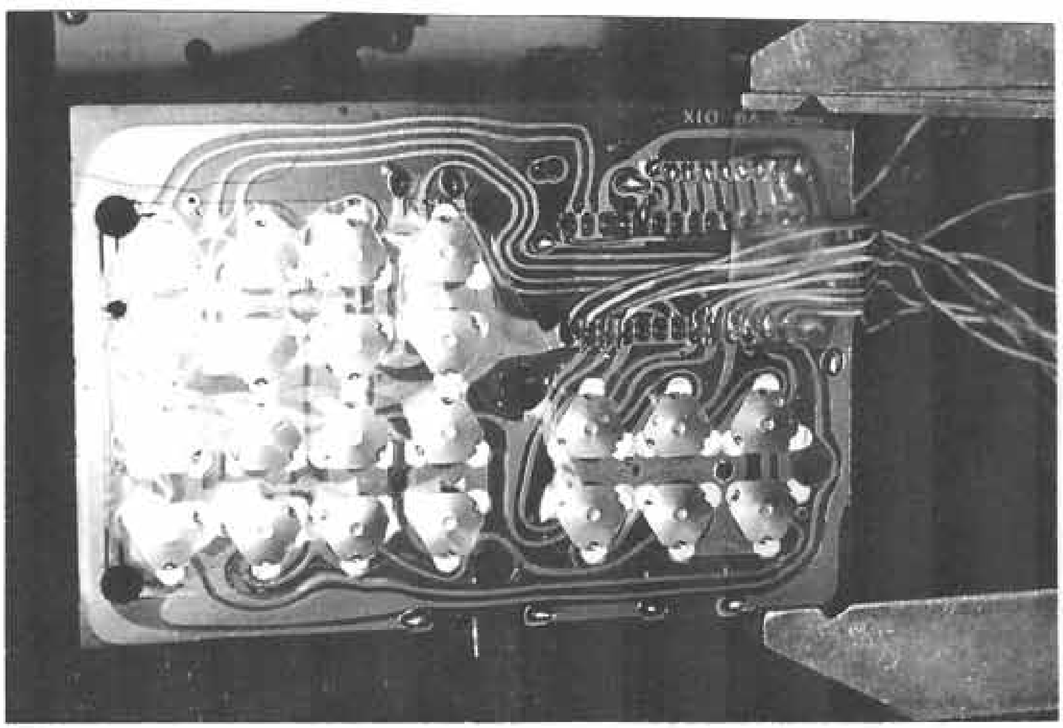
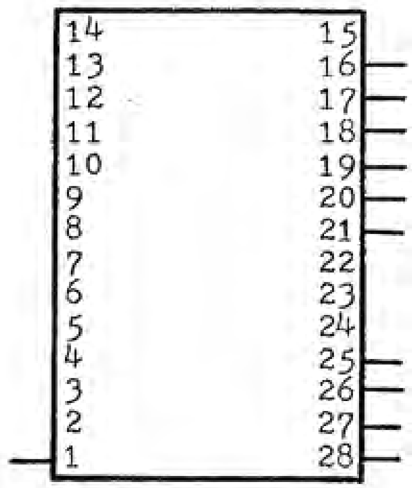


Figure 6-1. BSR controller circuit board.

Figure 6-2. BSR integrated circuit pinout.

BSR X-10
Switch Control IC



(foil side)

IC CONNECTION —— to ——	16 PIN DIP JUMPER SOCKET
1	1
16	2
17	3
18	4
19	5
20	6
21	7
25	8
26	9
27	10
28	11

Table 6-3
SUMMARY OF SWITCH CONNECTIONS TO 16 PIN JUMPER

SWITCH	JUMPER PINOUT		RELAY #
1	1	3	1
2	3	11	8
3	1	6	9
4	6	11	10
5	1	4	11
6	4	11	12
7	1	5	13
8	5	11	14
ON	6	8	2
OFF	8	10	3
BRIGHT	5	8	4
DIM	8	9	5
ALL LIGHTS ON	4	8	6
ALL OFF	7	8	7

Pinout for remaining switches not used in this project.

9	1	2
10	2	11
11	1	10
12	10	11
13	1	7
14	7	11
15	1	9
16	9	11

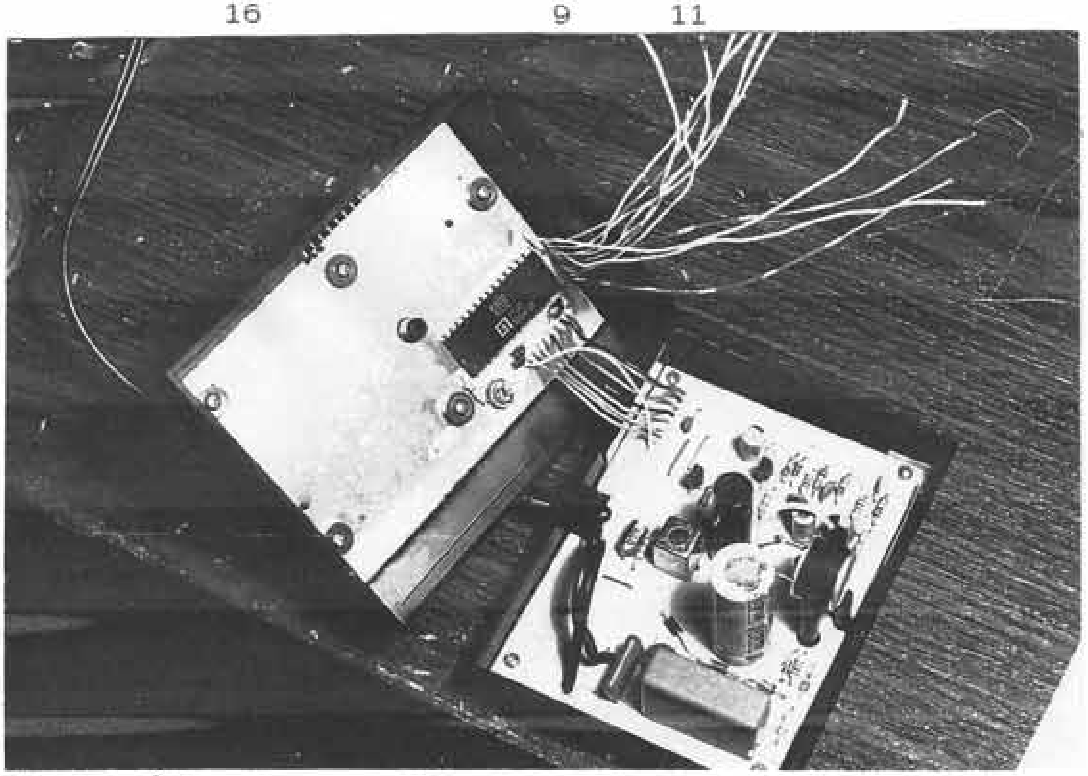


Figure 6-3. Wires soldered to BSR controller extending off circuit board.

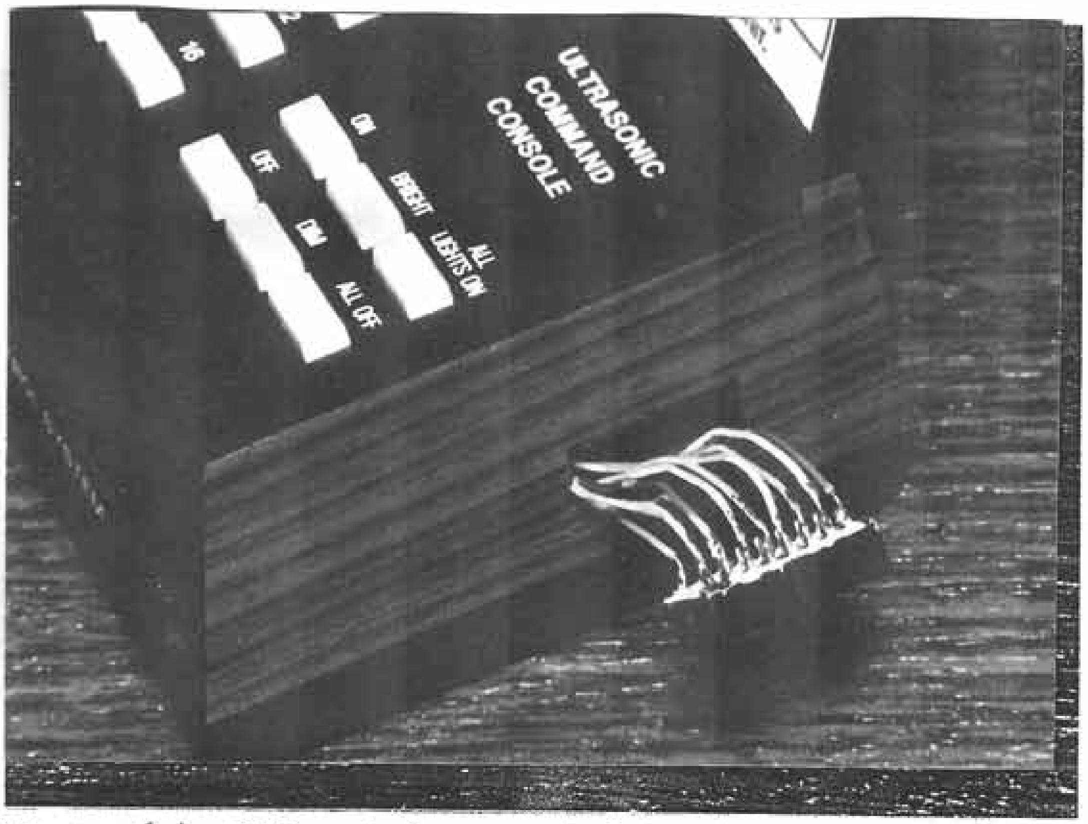


Figure 6-4. DIP socket connected to wires from BSR circuit board.

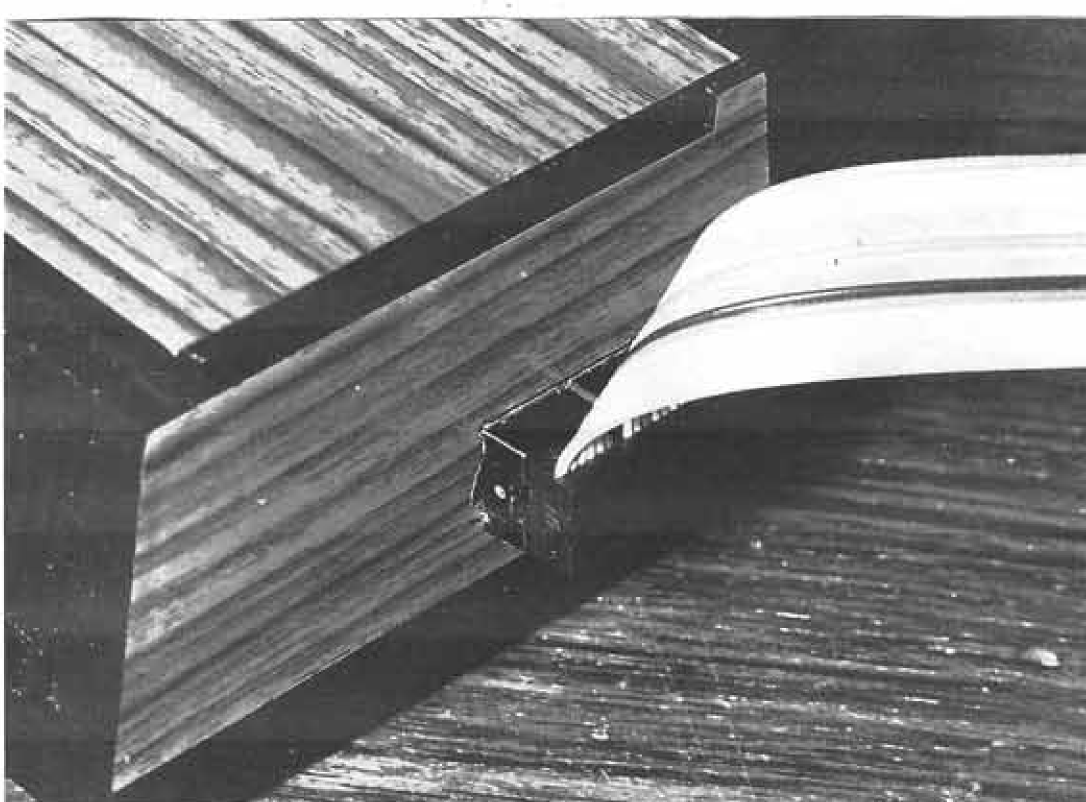
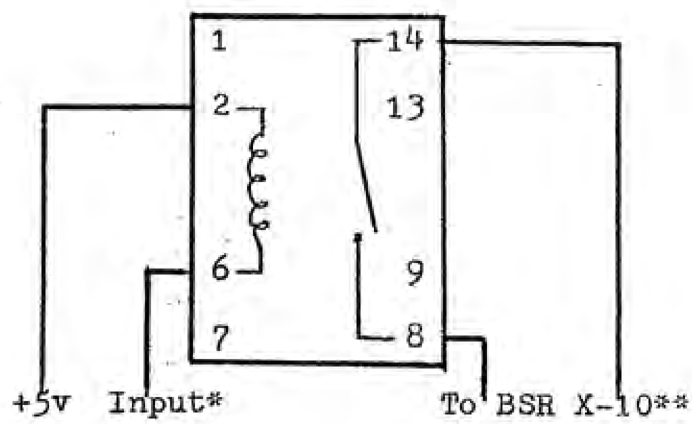


Figure 6-5. DIP socket fastened to BSR case with jumper cable connected.

Figure 6-6. BSR interface DIP relay/transistor array circuit.

DIP RELAY

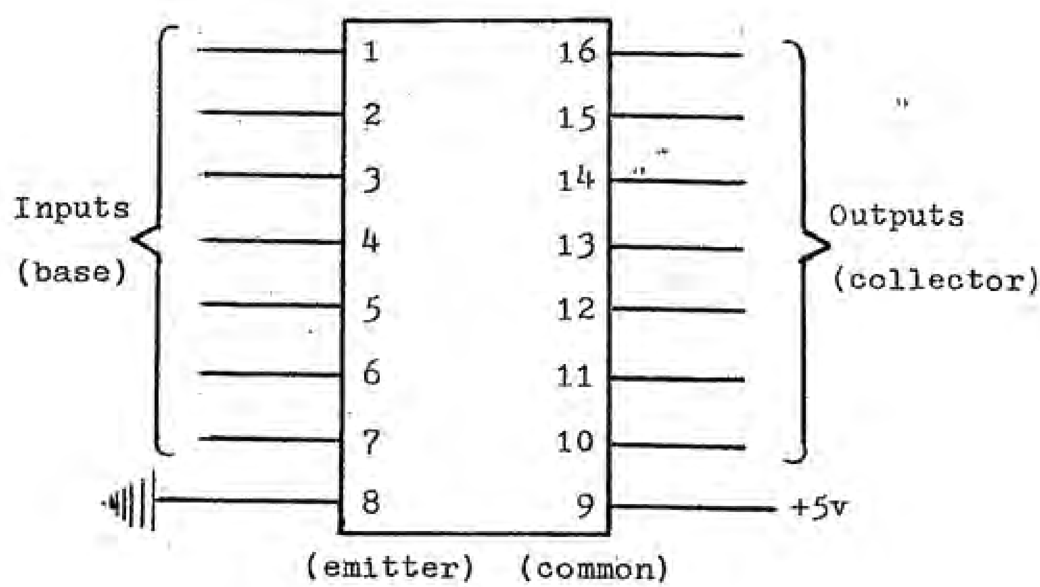


* For relay #1 this would be wired to pin 16 of trans array #1.

** For relay #1 these would be wired to pins 1 and 3 of jumper socket.

DARLINGTON TRANSISTOR

ARRAY



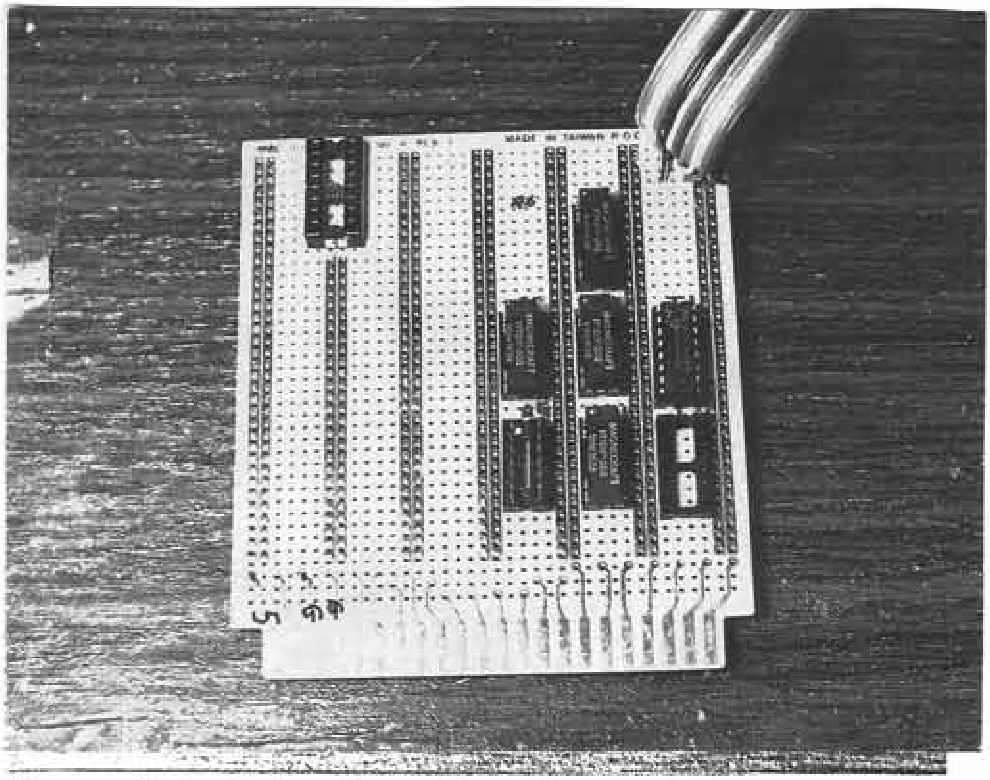


Figure 6-7. Interface circuit assembled on a Hobby Board circuit card, component side view.

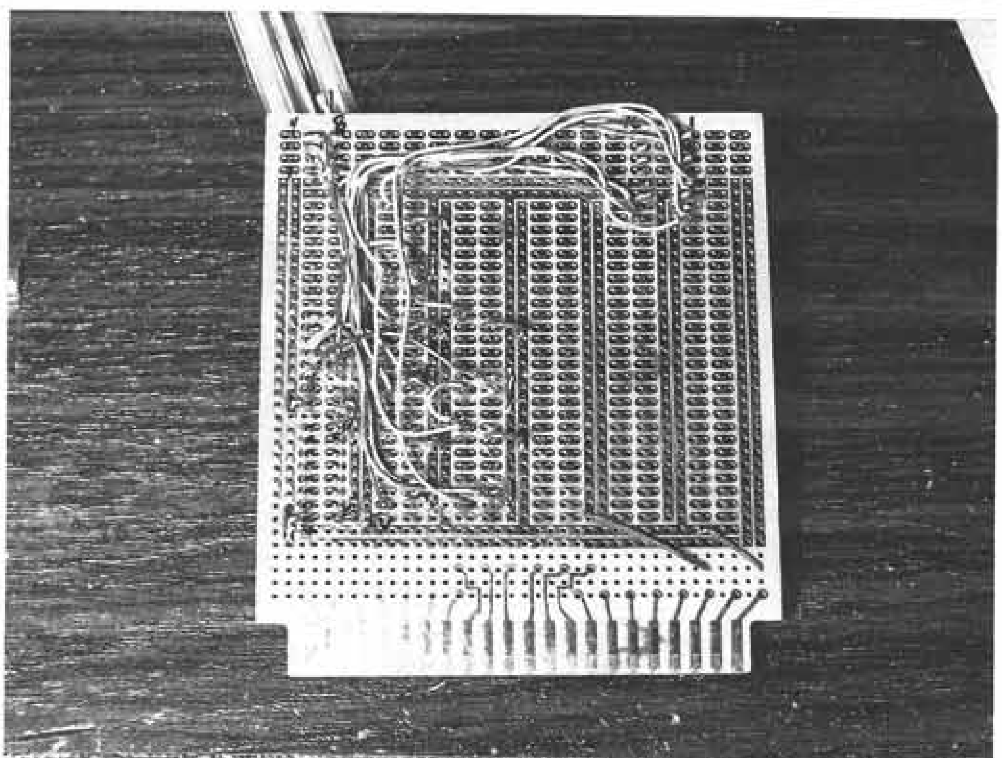


Figure 6-8. Interface circuit on Hobby Board, solder side.

relay sockets (relays only installed in four sockets); and, the 16 pin DIP socket at the upper left is the output to the X-10. The flat cable in the upper right corner is a jumper from the RX81 output. A complete pinout for the project is listed in Table 6-4.

For an example of the end-to-end wiring of the project, I will walk you through one control line. RX81 output D1 (referred to as 1-2) is connected to pin #2 of transistor array #1. The corresponding output of the array is pin #15. This is in turn connected to pin #6 of relay #2. Pin #8 of relay #2 is connected to pin #6 of the DIP jumper socket. In order to complete the switch action of the relay contacts, pin #14 of the same relay is connected to pin #8 of the DIP jumper socket. This line controls the "ON" switch of the X-10.

Table 6-4
CONTROLLER INTERFACE TRANSISTOR ARRAY/RELAY PINOUT

	Darlington Transistor Arrays *			DIP Relays			
RX-81 Output D0-6	Trans Array #	Input Pin #	Output Pin #	Input to Pin 6 of Relay #	Pin 8 to DIP Jumper Socket Pin #	Pin 14 to DIP Jumper Socket Pin #	Control Switch
1-1	1	1	16	1	1	3	"1"
1-2	1	2	15	2	6	8	"ON"
1-3	1	3	14	3	8	10	"OFF"
1-4	1	4	13	4	5	8	"BRIGHT"
1-5	1	5	12	5	8	9	"DIM"
1-6	1	6	11	6	4	8	"ALL ON"
1-7	1	7	10	7	7	8	"ALL OFF"
2-1	2	1	16	8	3	11	"2"
2-2	2	2	15	9	1	6	"3"
2-3	2	3	14	10	6	11	"4"
2-4	2	4	13	11	1	4	"5"
2-5	2	5	12	12	4	11	"6"
2-6	2	6	11	13	1	5	"7"
2-7	2	7	10	14	5	11	"8"

RX-81 output board #1 wired as "out 7"
RX-81 output board #2 wired as "out 6"

Example: Output D0 from board wired as "out 7" is listed above as 1-1.

* ULN2003A

Remember that ground and +5 volt connections also have to be made to each transistor array and relay (Figure 6-6). The +5 volts is connected to pin 2 of each relay and pin 9 of each transistor array. Also, a ground connection is made to pin 8 of the transistor array. Note that the ground connection of the coil of each relay is made by the output of the transistor array.

After all wiring is complete, the relay board is connected to the BSR X-10 with a 16 pin double male ended DIP jumper cable as pictured in Figure 6-9. The RX81 boards and expansion bus are also visable in the photo. My entire computer setup with the BSR X-10 sitting on top of the case housing the expansion board is pictured in Figure 6-10.

CONTROL SOFTWARE

Now that the system is wired up, it's time to punch up some programs and start controlling. First, enter the machine code into the "REM 1" statement for the ZX81/TS1000/TS1500 as described in Chapter 3 and as you did for the robot control software (Chapter 5). If you are plugging an RX81 board directly into the TS2068, then start the TS2068 software with statements 1 through 3 as described in Chapter 3. Table 6-6 summarizes the software commands, which are very similar to the robot control commands. In this application you POKE a command which latches a switch "ON", PAUSE a short time to activate the BSR X-10 circuit and move on to the next command. The sample program in Table 6-5 assigns each BSR X-10 function/switch to a key on the computer as listed in Table 6-6. To turn on appliance or lamp module coded as "1" you first press the computer key "1" and then the key "O". To turn all lamp modules "ON" you press computer key "A". Table 6-7 contains a short demonstration program that does not use keyboard input. Run this program and various "on" and "off" functions will occur automatically. After each "on" or "off" action the program will STOP. To continue to the next function, enter CONTinue and this control program will step through, performing one function at a time from each program segment.

Of course you can work out other control programs, limited only by your imagination. Happy controlling!

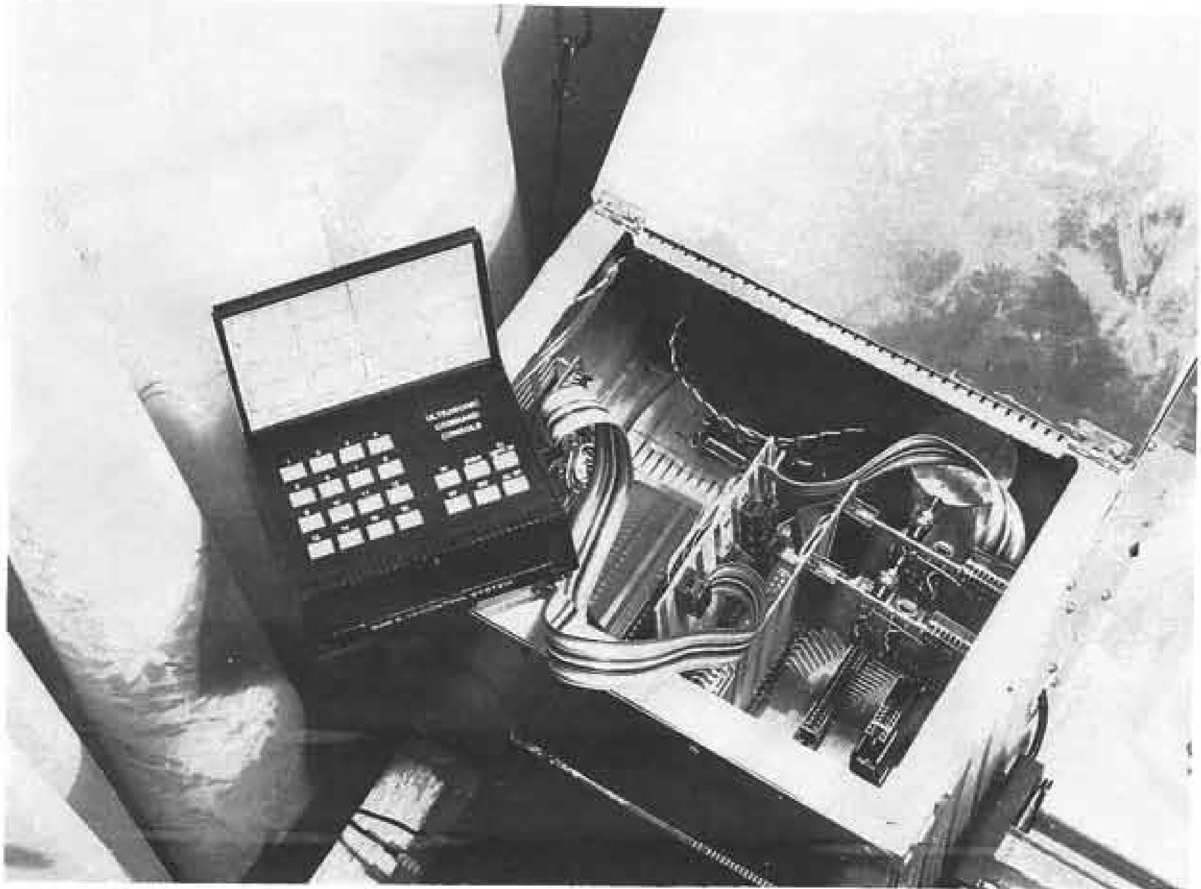


Figure 6-9. Jumper cable shown connecting BSR controller with RX81 board plugged into Expansion Board.

Figure 6-10. Home control center showing computer, external keyboard, expansion and interface boards in enclosure and television monitor.

Table 6-5. Home control software listing.

```
   1 REM <=* J?<=* ... PLOT  <=* ...
LIST * TAN Y* PEEK *TAN
  30 LET A=16534
  40 LET B=16536
  50 LET C=16533
  60 POKE A,0
  70 POKE B,7
  80 LET OUT=USR C
  90 POKE A,0
 100 POKE B,6
 110 LET OUT=USR C
 120 LET S$=INKEY$
 130 IF S$="1" THEN GOTO 290
 140 IF S$="O" THEN GOTO 350
 150 IF S$="F" THEN GOTO 410
 160 IF S$="B" THEN GOTO 470
 170 IF S$="D" THEN GOTO 530
 180 IF S$="L" THEN GOTO 590
 190 IF S$="A" THEN GOTO 650
 200 IF S$="2" THEN GOTO 710
 210 IF S$="3" THEN GOTO 780
 220 IF S$="4" THEN GOTO 840
 230 IF S$="5" THEN GOTO 900
 240 IF S$="6" THEN GOTO 960
 250 IF S$="7" THEN GOTO 1020
 260 IF S$="8" THEN GOTO 1080
 270 GOTO 120
 280 REM SW 1
 290 POKE A,1
 300 POKE B,7
 310 LET OUT=USR C
 320 PAUSE 30
 330 GOTO 60
 340 REM  ON
 350 POKE A,2
 360 POKE B,7
 370 LET OUT=USR C
 380 PAUSE 30
 390 GOTO 60
 400 REM  OFF
 410 POKE A,4
 420 POKE B,7
 430 LET OUT=USR C
 440 PAUSE 30
 450 GOTO 60
 460 REM  BRIGHT
 470 POKE A,8
 480 POKE B,7
 490 LET OUT=USR C
 500 PAUSE 30
 510 GOTO 60
 520 REM  DIM
 530 POKE A,16
 540 POKE B,7
 550 LET OUT=USR C
 560 PAUSE 180
 570 GOTO 60
 580 REM ALL LIGHTS ON
```

Table 6-5 (Continued)

```
 580 REM ALL LIGHTS ON
 590 POKE A,32
 600 POKE B,7
 610 LET OUT=USR C
 620 PAUSE 30
 630 GOTO 60
 640 REM ALL OFF
 650 POKE A,64
 660 POKE B,7
 670 LET OUT=USR C
 680 PAUSE 30
 690 GOTO 60
 700 REM SW 2
 710 POKE A,1
 720 POKE B,6
 730 LET OUT=USR C
 740 PAUSE 30
 750 GOTO 60
 760 REM SW 3
 780 POKE A,2
 790 POKE B,6
 800 LET OUT=USR C
 810 PAUSE 30
 820 GOTO 60
 830 REM SW 4
 840 POKE A,4
 850 POKE B,6
 860 LET OUT=USR C
 870 PAUSE 30
 880 GOTO 60
 890 REM SW 5
 900 POKE A,8
 910 POKE B,6
 920 LET OUT=USR C
 930 PAUSE 30
 940 GOTO 60
 950 REM SW 6
 960 POKE A,16
 970 POKE B,6
 980 LET OUT=USR C
 990 PAUSE 30
1000 GOTO 60
1010 REM SW 7
1020 POKE A,32
1030 POKE B,6
1040 LET OUT=USR C
1050 PAUSE 30
1060 GOTO 60
1070 REM SW 8
1080 POKE A,64
1090 POKE B,6
1100 LET OUT=USR C
1110 PAUSE 30
1120 GOTO 60
```

Table 6-6
HOME CONTROL SOFTWARE COMMANDS

Function/Switch	Relay	Output	POKE A,	POKE B,	Keyboard Entry
"1"	1	1-1	1	7	1
"ON"	2	1-2	2	7	O
"OFF"	3	1-3	4	7	F
"BRIGHT"	4	1-4	8	7	B
"DIM"	5	1-5	16	7	D
"ALL LIGHTS ON"	6	1-6	32	7	L
"ALL OFF"	7	1-7	64	7	A
"2"	8	2-1	1	6	2
"3"	9	2-2	2	6	3
"4"	10	2-3	4	6	4
"5"	11	2-4	8	6	5
"6"	12	2-5	16	6	6
"7"	13	2-6	32	6	7
"8"	14	2-7	64	6	8

(Note: A = 16534 and B = 16536)

Table 6-7. Home control demonstration without keyboard input.

```
  1 REM <=" J?<=" J▦4 PLOT <=" J▦4
LIST " TAN Y" PEEK ▦TAN
 20 LET A=16534
 30 LET B=16535
 40 LET C=16533
 50 POKE A,1
 60 POKE B,6
 70 LET OUT=USR C
 80 STOP
 90 GOSUB 530
100 STOP
110 POKE A,2
120 POKE B,6
130 LET OUT=USR C
140 STOP
150 GOSUB 530
160 STOP
170 POKE A,4
180 POKE B,6
190 LET OUT=USR C
200 STOP
210 GOSUB 530
220 STOP
230 POKE A,8
240 POKE B,6
250 LET OUT=USR C
260 STOP
270 GOSUB 530
280 STOP
290 POKE A,16
300 POKE B,7
310 LET OUT=USR C
320 STOP
330 GOSUB 570
340 STOP
350 POKE A,32
360 POKE B,7
370 LET OUT=USR C
380 STOP
390 GOSUB 570
400 STOP
410 POKE A,64
420 POKE B,7
430 LET OUT=USR C
440 STOP
450 GOSUB 570
460 STOP
470 POKE A,128
480 POKE B,7
490 LET OUT=USR C
500 STOP
510 GOSUB 570
520 STOP
530 POKE A,0
540 POKE B,6
550 LET OUT=USR C
560 RETURN
570 POKE A,0
580 POKE B,7
590 LET OUT=USR C
600 RETURN
```

THIS PAGE INTENTIONALLY BLANK

Chapter 7
STEPPER MOTOR CONTROL

This project requires that you first build the 8255 PPI I/O circuit described in the second part of Chapter 3. With the addition of the simple circuit described below, you can drive a moderately priced 12 volt stepper motor which enables precise and predictable positioning for the enhancement of projects already described or for the creation of new project ideas. For example, at the end of this chapter will be an illustration of how the stepper motor can be added to the robot arm to create a rotating wrist. Or, for a new project the stepper motor could be applied to building an X-Y plotter or a multiple axis precise positioning table or arm.

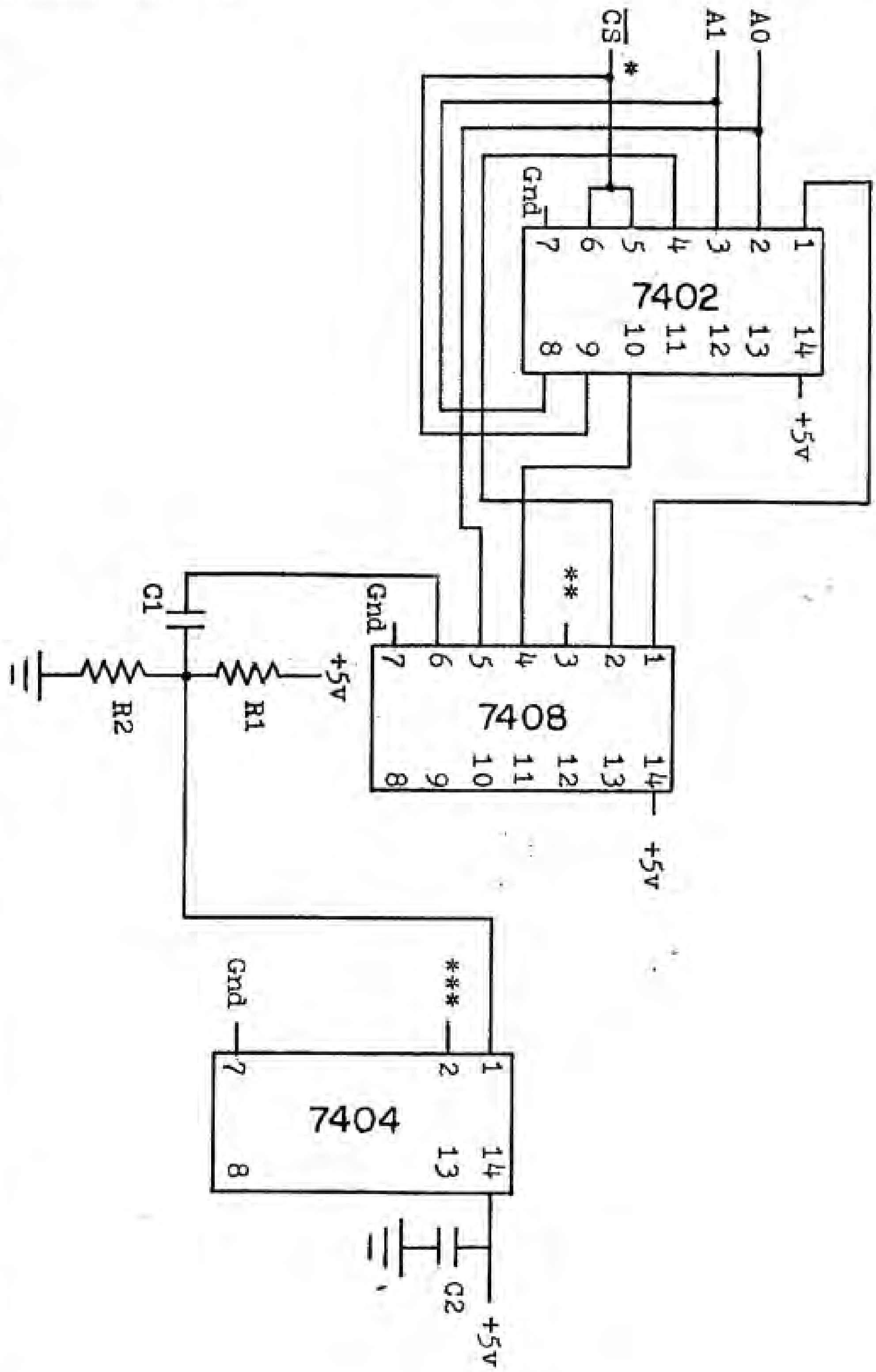
CIRCUIT ASSEMBLY

A complete parts list is provided in Table 7-1. Before proceding with the assembly of the above components, refer back to Figure 3-5 and wire up the 7402 and 7408 ICs (located above the 8255) in accordance with Figure 7-1. Also at this point wire up the 14 and 16 pin jumper sockets (to the left of the 8255 chip in Figure 3-5) if you have not yet done so. The 14 pin

Table 7-1
STEPPER MOTOR PARTS LIST

(1) PC board (Radio Shack # 276-168)
(1) 16 pin DIP socket
(2) 14 pin DIP sockets
(1) 3.5mm audio jack
(1) 5 pin auto speaker connector (Radio Shack # 274-1210 and # 274-1215)
(1) Single end 14 pin DIP jumper (DIGI-KEY # R102-12-ND)
(1) 7404 IC
(1) 7416 IC
(1) SAA 1027 motor driver IC (Gledhill Electronics)
(1) 100 ohm 1/4 watt resistor
(1) 1K ohm 1/4 watt resistor
(1) 2K ohm 1/4 watt resistor
(2) 5.1K ohm 1/4 watt resistor
(1) 150 ohm 1 watt resistor
(2) 0.01 uF capacitor
(1) .1uF capacitor
(1) Unipolar 12 volt stepper motor, AIRPAX, North American Philips, part number K82701-P2 (Gledhill Electronics)

Figure 7-1. Schematic for pulse coding stepper motor and Digitalker interfaces.



* from pin 8 of IC2 (7430) of 8255 I/O Port Address Decoder
** to pin 4 (WR) of Digitalker SPC
*** to pin 3 of 7416 in motor drive circuit

C1 = .01uF
C2 = .05uF
R1 = 2K ohm ¼ watt
R2 = 1K ohm ¼ watt

socket is required for this project and the 16 pin socket is required for the Digitalker project. Table 3-14 summarizes the pinouts for these sockets.

Now you can wire up the stepper motor circuit in accordance with Figures 7-2 and 7-3. A summary of the IC logic is also depicted in Figure 7-4. The wiring is fairly straight forward, but I will run through the steps that might not be obvious. First, you might be wondering about the audio jack. That will be your 12 volt input point. And where do you get the 12 volt supply? It is supplied from the old power supply that you used to power your computer before you added your Expansion Board. This power supply will power the circuit and stepper motor. The other component you might wonder about is the Radio Shack five pin auto speaker connector. As can be seen in Figure 7-2 this is used to make the connection to the stepper motor. Although there are eight leads coming from the motor, four of them are ground and can be connected together into one common pin on the connector plug.

Although detailed technical data is supplied with the motor and SAA 1027 when ordered from Gledhill, the following is a brief rundown. The two will operate within a voltage range of 9.5 to 18 VDC. The maximum current draw is 800 mA. The motor steps 7.5 degrees per pulse which equates to 48 steps per revolution. Pin 2 on the SAA 1027 is the set input (S); pin 15 is the trigger input (T) causing the IC to step the motor on the positive edge of a high low high pulse; and, pin 3 is the

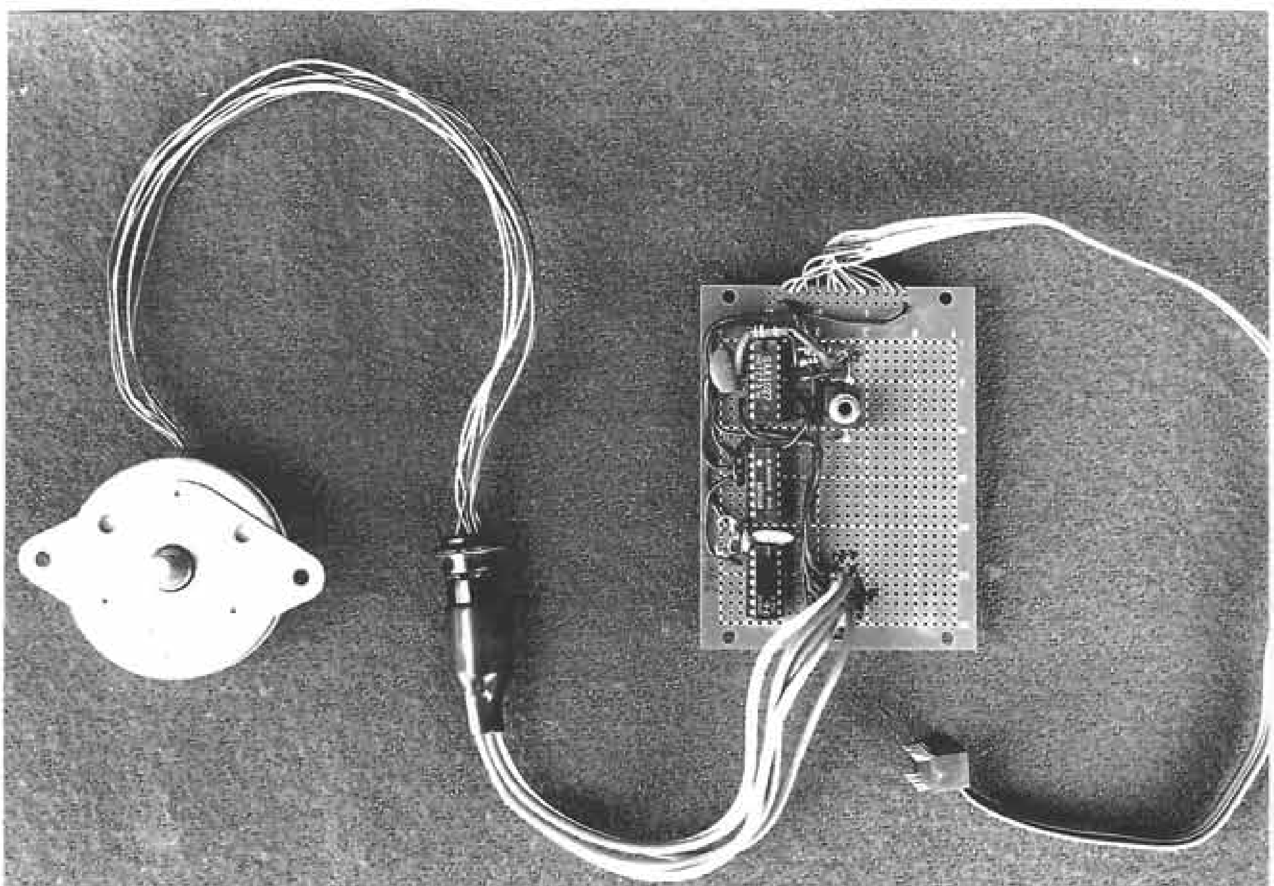
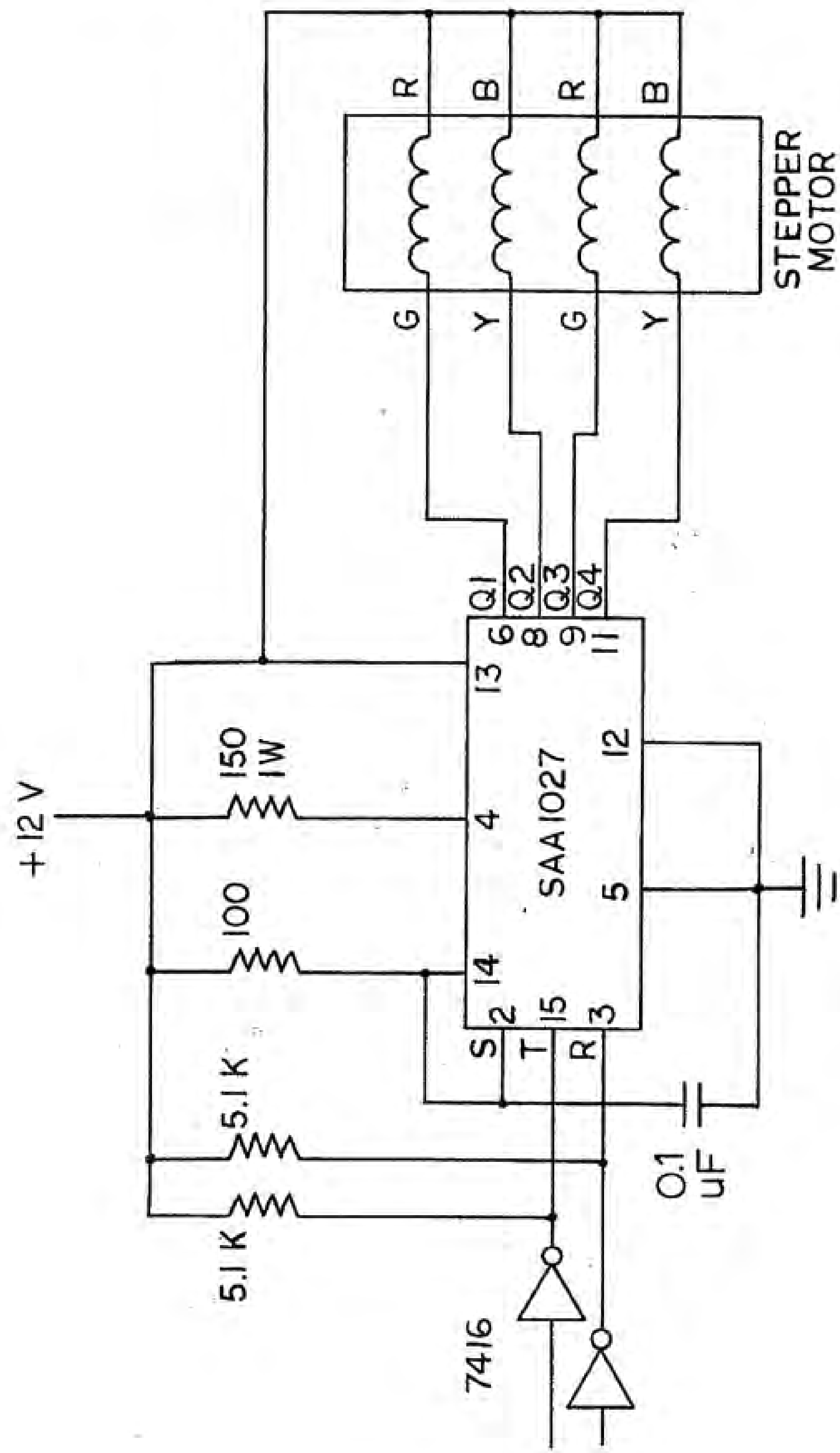


Figure 7-2. Stepper motor driver board and motor.

Figure 7-3. Stepper motor driver circuit schematic.



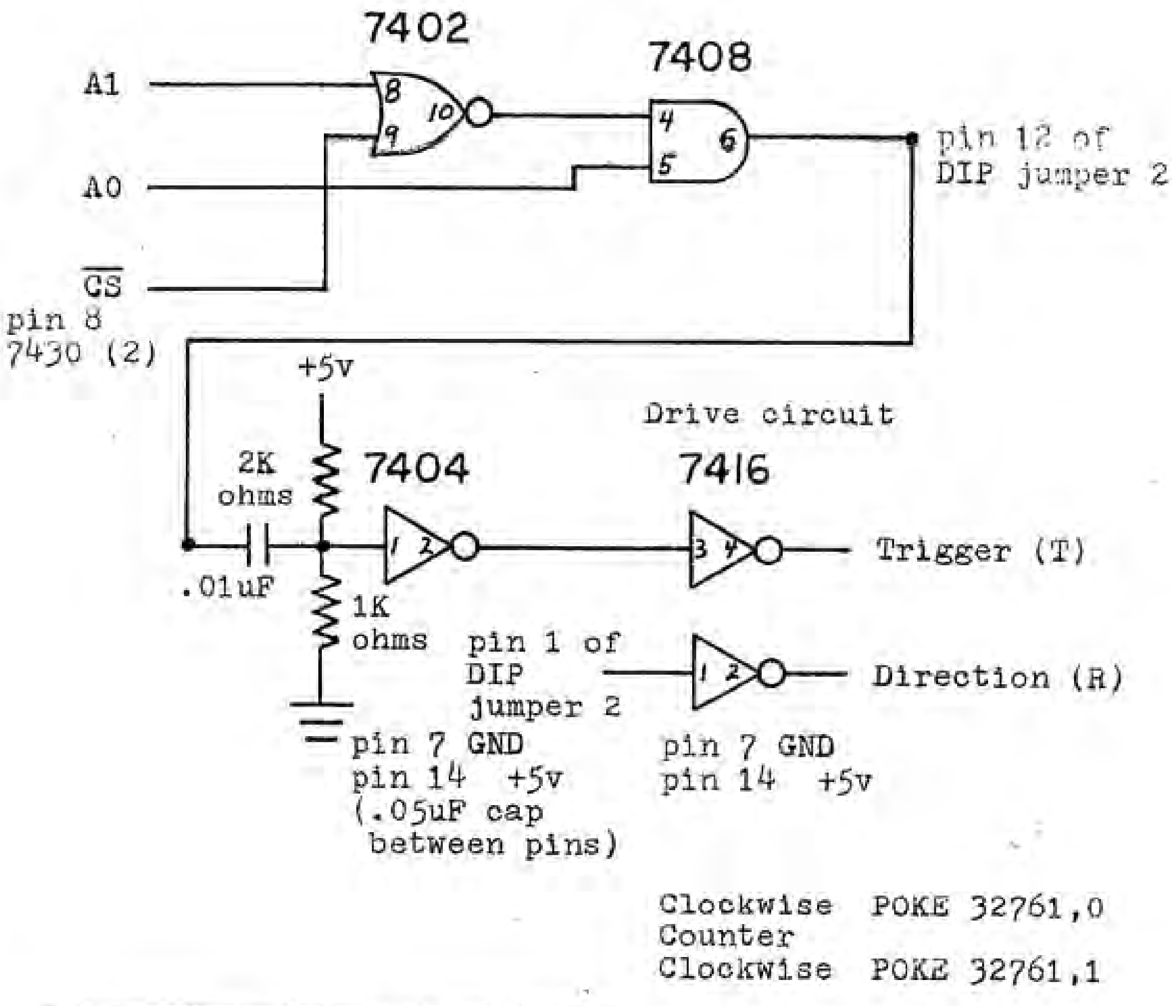


Figure 7-4. Output pulse coding from 8255, port B for input to stepper motor drive circuit.

direction input with a high causing counterclockwise (CCW) stepping and a low causing clockwise (CW) stepping. The motor steps with approximately seven ounce inches of torque at one or less steps per second ranging down to almost zero torque at 200 steps per second. The motor will feel quite hot to touch when it is on, and holding, but do not worry as the maximum operating temperature is 100 degrees centigrade (212 degrees fahrenheit).

MOTOR CONTROL

As you saw from the wiring outlined in Table 3-14, the 14 pin DIP jumper supplies the stepper motor circuit with the required signals and +5 volt power for the two logic ICs, and the control comes from port B of the 8255. The power for the motor and SAA 1027 comes from the 12 volt supply plugged directly into the stepper driver circuit board.

The software is quite simple. First reset RAMTOP if necessary and then POKE the command into the 8255 to turn on port B for output. The circuit uses only data line B0 from port B. All you do to pulse the motor one step CW is "POKE 32761,0". For CCW motion you pulse the motor one step with a "POKE 32761,1". The routine provided in Table 7-2 would first rotate the motor one revolution CW and then return the shaft to the

Table 7-2
STEPPER MOTOR DEMONSTRATION SOFTWARE

```
10 FOR N = 1 TO 48
20 POKE 32761,0
30 NEXT N
40 FOR N = 1 TO 48
50 POKE 32761,1
60 NEXT N
```

original position by rotating it one revolution CCW.

This routine will drive the motor at about 27 pulses per second which equates to one complete revolution in 1.8 seconds. If you want the motor to run faster, insert the following statements in the above program: "5 FAST" and "35 SLOW".

This will drive the motor at about 137 pulses per second in the clockwise direction, equating to one complete revolution in one third of a second. On the other hand, the revolution speed, which is governed by the pulse rate, can be slowed by introducing a pause statement between the POKE and NEXT statements, or otherwise slowing the program.

APPLICATION

Figure 7-5 should fuel your imagination even if you haven't built a robot. The driver board (Figure 7-2) has enough room for at least three more motor drive outputs if one is not sufficient. All you have to add in the way of hardware is an SAA 1027 for each motor. The two logic chips have five more gates available and only one is required for each additional circuit. Just remember to beef up your 12 volt supply to accomodate additional motors.

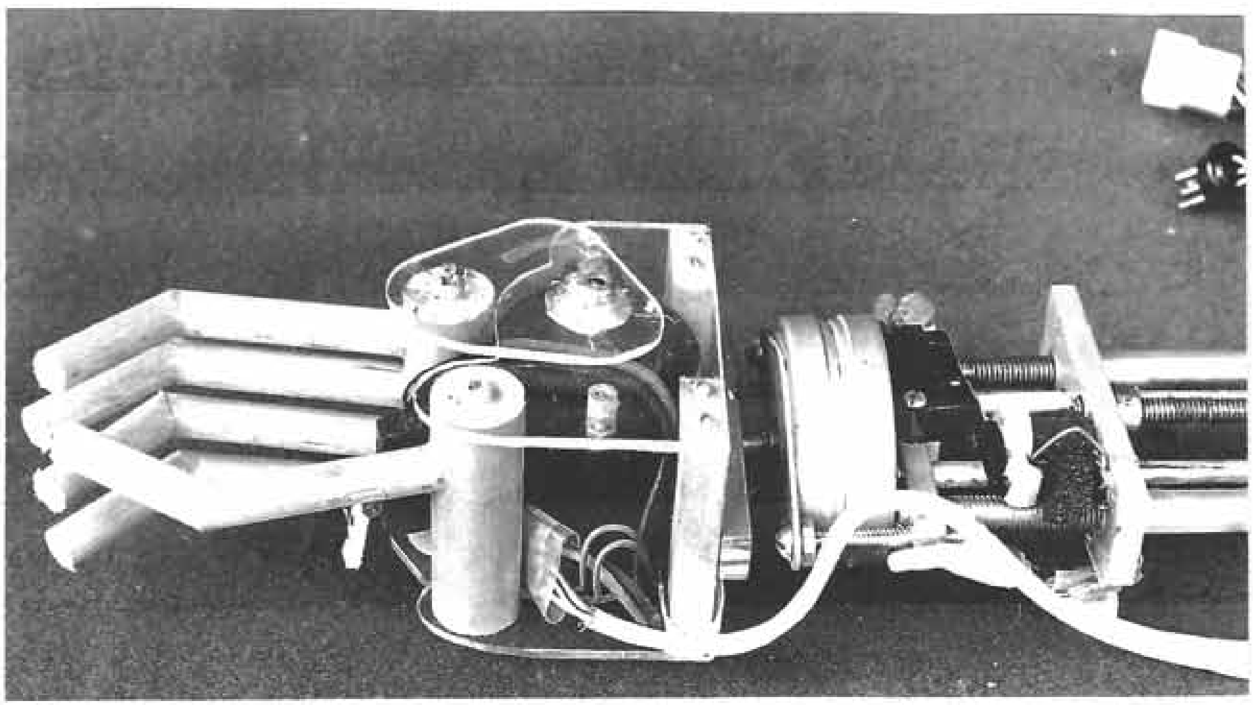


Figure 7-5. Robot lower right arm with stepper motor added for wrist movement.

Chapter 8
LOW COST OPTICAL ENCODING

Optical encoding is a method of inputting data into your computer which is cheap and fairly easy to accomplish. Optical encoding can be used to count, measure or position by translating the physical motion of an object or device into pulses which a computer can count. You have probably seen optical encoders but have just not recognized them. The supermarket checkout device at the cash register that reads the bar codes printed on the product package is a type of optical encoder. The small hand held gun like device connected to the cash register that the sales clerk uses to scan and read the numbers printed on the back of your credit card is an optical encoder.

The "front end" of the optical encoder consists of a light source and a phototransistor. These two components are aligned so that they either point toward each other, or aligned so that the light source reflects off the area to be read, back into the phototransistor. If they point toward each other, a mechanical device such as a disc or tube with slots or holes is placed between the light source and the phototransistor so that when the device is rotated or moved, the transmitted light is interrupted momentarily. These alternations in light and dark (or no light) cause the phototransistor to turn on and off in the same way that a transistor switch turns on and off when power is applied to or withdrawn from the base of a transistor. The light activates the base of the phototransistor.

CIRCUIT EXPLANATION

In the circuit described here, I use an infrared (IR) transmitting light emitting diode (LED) and an IR sensitive phototransistor pointing toward each other. Using IR devices minimizes the interference caused by ambient light.

Also, to ensure that the device produces a "clean" pulse that can be easily read as computer input, a simple integrated circuit is added to the encoder circuitry. This device is called a "Hex Schmitt Trigger (inverting)" and comes in a 14 pin DIP package known as a 74LS14. The package is actually six separate Schmitt triggers each capable of operating an independant encoder circuit.

There are numerous applications of this circuit, limited only by your imagination. The application I will describe is for the robot project (Chapters 4 & 5) where I count motor shaft revolutions and enable precise positioning of devices driven by the motors. These devices include the robot drive wheels, arms, etc. The method for controlling the length of time the motor is turned on in the robot control chapter was a simple sequence using the PAUSE or FOR-NEXT routine. This method is not precise. Although the computer will consistently time these routines, the distance the motors travels (revolutions per second) will vary, depending on the motor voltage, which is directly effected by the battery condition (charge) at the time. However, by counting the motor revolutions with an optical encoder, it doesn't make any difference how fast or slow the motor runs. A precise revolution count will yield a precise and consistent distance of travel.

For example, by counting the revolutions of the drive wheels, the robot can accurately measure the distances it travels, mapping out its location and "learning" from its travels around a room. Another example would be to count the number of revolutions which the robot elbow motor turns in order to precisely control the distance that the arm would extend, in order to perform a function like picking up an object.

One way to accomplish this measurement with an encoder is to place a hollow tube over the motor shaft with a hole drilled through the tube on a perpendicular to the shaft. The optical encoder is placed over this shaft end tube in such a way so that, as the shaft turns, the hole in the tube will allow the light from the LED to pass through the hole and strike the photo transistor twice during each shaft revolution. Of course it alternately (between each hole alignment) blocks the light twice during each revolution. This will produce four pulses to the computer for each revolution of the motor shaft. For the purposes of my robot application, these quarter revolution measurements are accurate enough. If greater positioning accuracy is desired, a small wheel with many holes can be placed on the shaft. The optical encoder would then be placed so that it lines up with the holes on this encoder wheel and even more positions per revolution can be measured. For example, an encoder wheel with eight holes will measure sixteen positions of the shaft during each revolution. To keep this explanation as simple as possible, I will describe a system with one hole perpendicular to the shaft, counting four pulses or positions per shaft revolution (Figure 8-1).

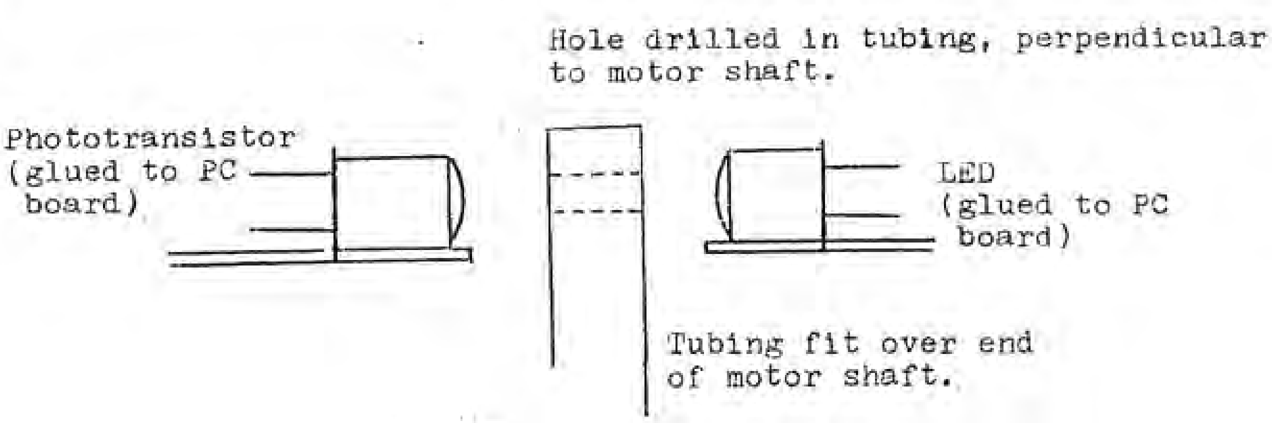


Figure 8-1. Optical encoder alignment with motor shaft.

In addition to the optical encoder circuit, an input/output (I/O) circuit is required between the encoder and the computer. I chose the RX81 I/O board, the same one I used in the robot circuits and the Home Control chapter (Chapter 6). Although other I/O boards could also be adapted, the RX81 is cheap, simple and very suited for the job. It will handle up to eight encoders, one encoder on each data line. I will also present several different software variations for this I/O board and the encoder.

The software used in this circuit is an important consideration and must be matched to the application. A "slower" performing software routine must be used for applications measuring lower pulse per second rates in order to prevent false pulses. I will explain the false pulse problem in a moment. On the other hand, "faster" performing software (all machine code) is required to enable accurate measurement of high pulse per second encoding. I will also describe medium speed software using a minimum basic language routine. To illustrate with an extreme example of software mismatch, consider an application where a shaft is turning one revolution per second. At four pulses per revolution, the encoder would be measuring four pulses per second. The all machine code software routine is capable of measuring at a rate in excess of hundreds of pulses per second. This means that the machine code version would be "looking" for a new pulse, hundreds of times per second. No motor runs smoothly without oscillating slightly during its rotation, even though this oscillation is not perceptable to the human eye. False pulse counts might be produced. Here is why. As the edge of the hole passes through the light path from the LED, the slight oscillation of the motor could be read as several on-off cycles on the encoder. Thus the software must be matched with the application, based on the pulse rate expected to be produced.

CIRCUIT ASSEMBLY

The assembly of the encoder circuit is fairly simple and can be adapted to fit the physical layout of the LED and phototransistor that suits your application. A parts list is provided in Table 8-1. Figure 8-2 shows a layout using a Radio Shack Experimentor Socket. You will probably want to construct a small PC board, based on your application like the layout of the

Table 8-1
OPTICAL ENCODER PARTS LIST

(1) 33K ohm, 1/4 watt resistor (R1)
(1) 330 ohm, 1/4 watt resistor (R2)
(1) Light emitting (IR) diode (MLED 930) or (TIL 31) or (TCG 3028) or (Radio Shack 276-143)
(1) Phototransistor (IR) (MRD 370) or (TIL 99) or (TCG 3036) or (Radio Shack 276-142)
(1) 14 pin DIP, Hex Schmitt Triggers (inverting) 74LS14
(1) Archer Experimentor Socket (Radio Shack 176-175)
(1) RX81 I/O board, bare board with parts list (Budget Robotics & Computing)

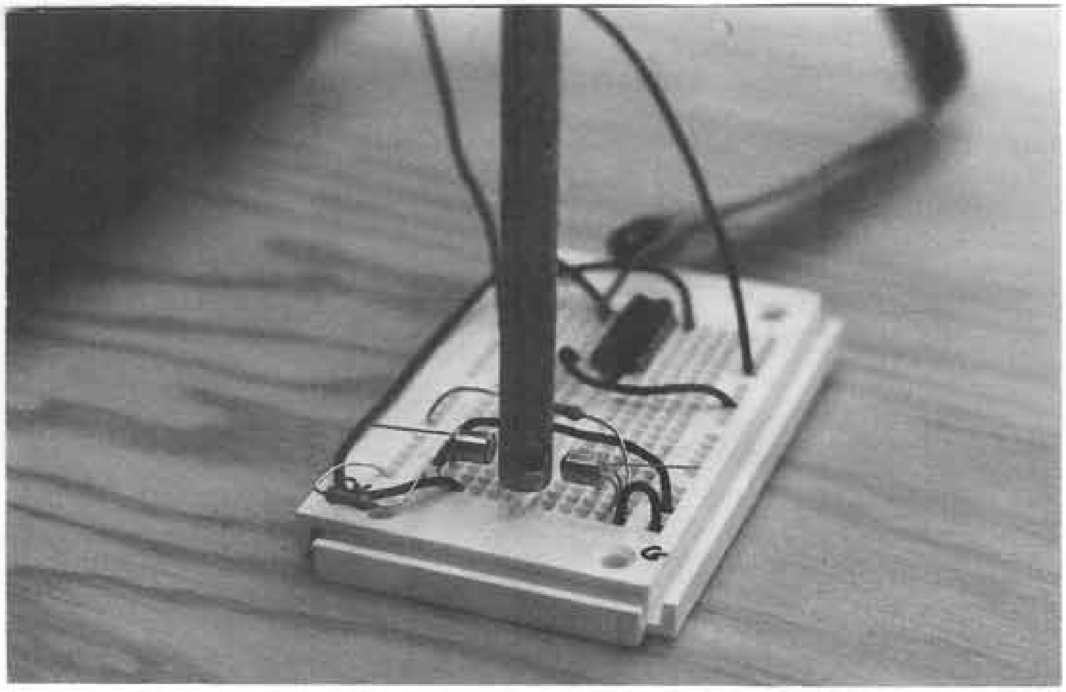
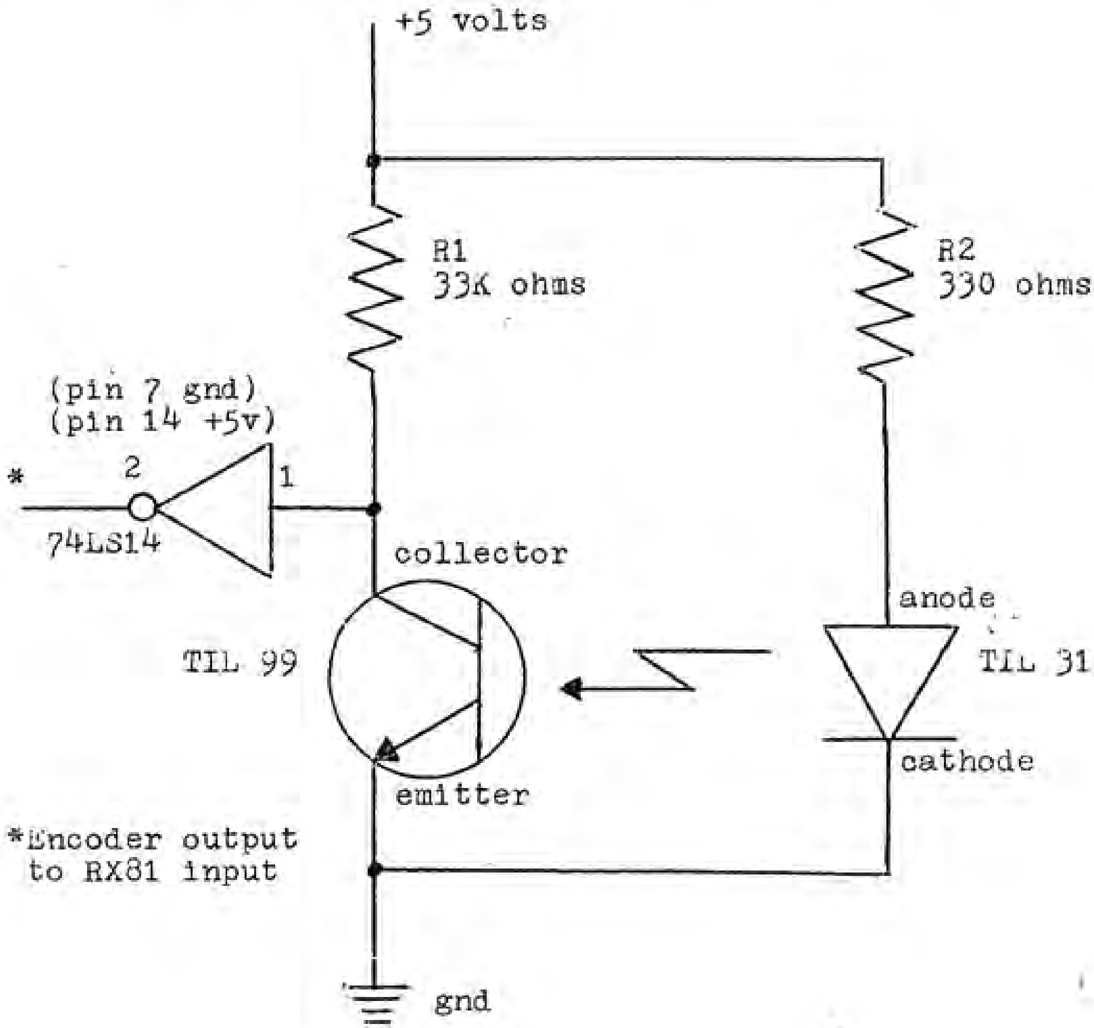
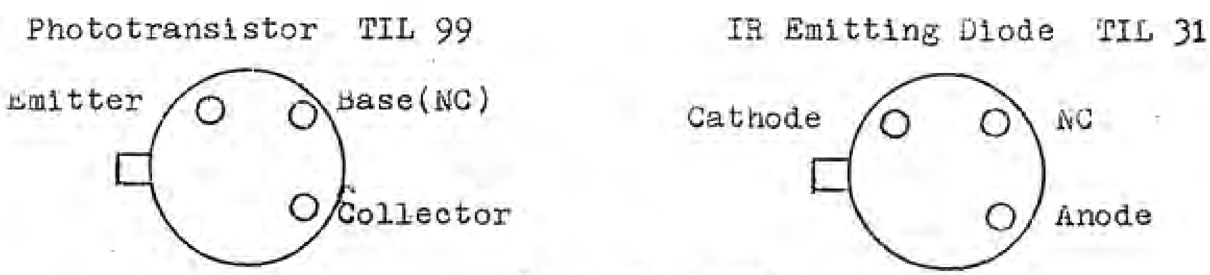


Figure 8-2. Optical encoder circuit board with shaft extention.

LED and phototransistor indicated in Figure 8-1. The prime components, the LED and phototransistor, will probably have to be purchased locally as I have not found them available from a mail order source. I found a Radio Shack equivalent after locating three other supplier and part numbers for each. You should have no problem finding one of the part numbers at an electronics parts supplier near you. If all else fails, I have also provided the address of a Tucson store who will respond to your mail request and supply the TIL part number in a Dowpak electronics parts package. Although all the part numbers I have listed are interchangable, you may have to vary the resistance of R1 to get the phototransistor to trigger properly. For example, if you use a TIL 99 (Figure 8-3) the value of R1 should be 33K ohms. If you use an MRD 370, the value of R1 should be 220K ohms. Although you can decrease the value of R2 (to 220 ohms for example) to increase the IR output of the LED (making it brighter) you will probably decrease its operating lifetime. Also keep in mind when you operate the circuit, that the LED is producing IR light which will not be visible to your eye. The LED won't appear to light up. I recommend you buy a cheap logic probe if you are going to be building circuits like this, as it makes the checkout for proper operation much easier. The only external connections from the encoder circuit are the output, +5 volts and ground. As shown in Figure 8-4, I used a DIP jumper to connect the output of the 74LS14 (pin 2) to the RX81 I/O board data line 1 (D0). The +5 volt and ground connection can come either directly from the computer bus connector or from the RX81 board.

Figure 8-3. Optical encoder circuit & TIL pinouts.





(bottom views)

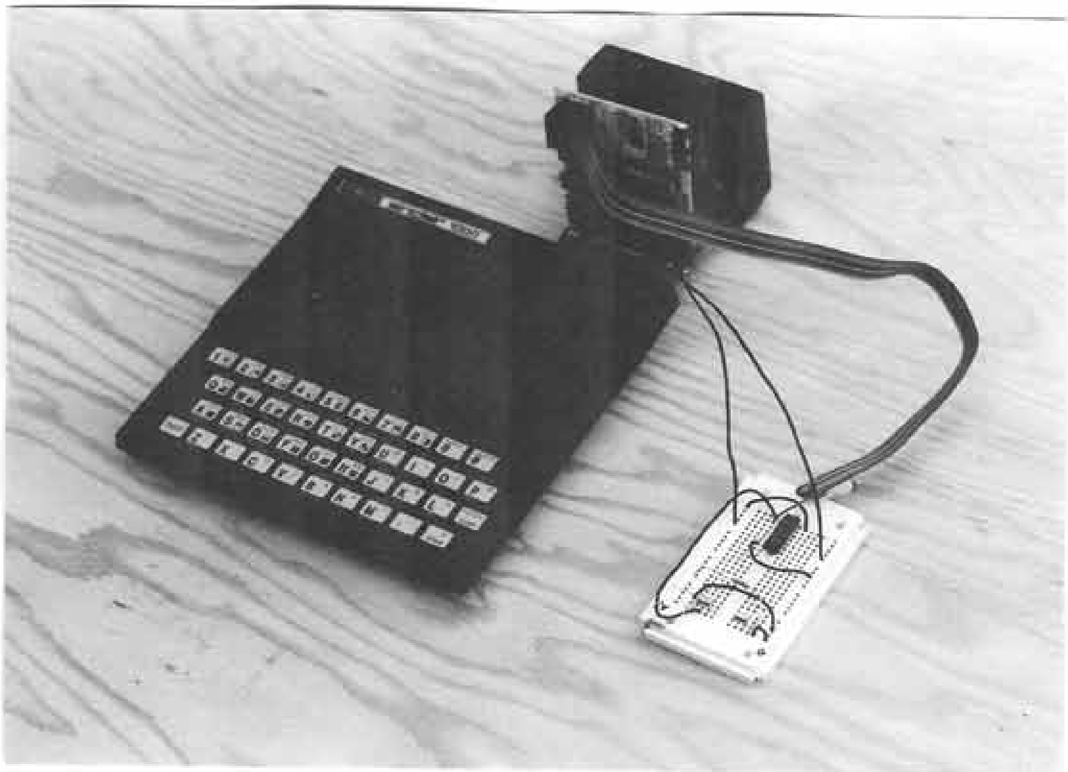


Figure 8-4. Overall view of computer, RX81 I/O board, 16K RAM and optical encoder

SOFTWARE DETAILS

Now for the software. Version 1 (Figure 8-5) will handle input up to the rate of about 45 pulses per second. This equates to a shaft speed of 675 revolutions per minute (rpm) using a single hole perpendicular to the shaft (4 pulses per revolution). Version 2 (Figure 8-6) will handle input up to the rate of about 73 pulses per second. This equates to a shaft speed of 1100 rpm at four pulses per revolution. For very high speed measurement, use version 3 (Figure 8-7) is all machine code. This version will handle input of hundreds of pulses per second, which equates to several thousand rpm at four pulses per second. I actually don't know how high an input rate this version will handle because I haven't found an application that exceeds its capability. If you have used the RX81 boards previously, you will notice that the machine code versions 1 and 2 differ from that supplied with these boards prior to 1984. The change in the machine code (documented in Chapter 3) allows the input to be read three times before the input is accepted by the computer as being valid. Table 8-2 contains an explanation of the version 3, all machine code instructions. I found that the technique of multiple inputs for confirmation was necessary even if the inputs were mechanical switches as described in Chapter 5. The rapid inputs of the encoder make this technique even more critical. If you have built the input/output circuit described in the robot article in SYNC magazine (Jul-Aug 83), you may want to modify the machine code statement in REM 2 to match that in versions 1 and 2 here. Remember that the input and output

Figure 8-5. Optical encoder Version 1 software.

```
  1 REM <=² J?<=² J▓4 PLOT <=² J▓4
 LIST ▗² TAN Y² PEEK ▓TAN
  10 PRINT "ENTER NUMBER OF REVO
LUTIONS TO  BE COUNTED"
  20 INPUT RV
  30 FAST
  40 LET PS=0
  50 LET IN=USR 16514
  60 IF IN=0 THEN GOTO 100
  70 IF IN=1 THEN GOTO 200
  80 IF IN>1 THEN GOTO 50
 100 LET IN=USR 16514
 110 IF IN=0 THEN GOTO 100
 120 LET PS=PS+.25
 130 IF PS=RV THEN GOTO 300
 140 IF IN=1 THEN GOTO 200
 200 LET IN=USR 16514
 210 IF IN=1 THEN GOTO 200
 220 LET PS=PS+.25
 230 IF PS=RV THEN GOTO 300
 240 IF IN=0 THEN GOTO 100
 300 PRINT PS;" REVOLUTIONS COMP
LETE"
 310 SLOW
```

Figure 8-6. Optical encoder Version 2 software.

```
  1 REM <=² J?<=² J▓4 PLOT <=² J▓4
 LIST ▗² TAN Y² PEEK ▓TAN
  10 PRINT "ENTER NUMBER OF REVO
LUTIONS TO  BE COUNTED"
  20 INPUT R
  30 FAST
  40 LET P=0
  50 LET I=USR 16514
  60 IF I=0 THEN GOTO 100
  70 IF I=1 THEN GOTO 200
  80 IF I>1 THEN GOTO 60
 100 LET I=USR 16514
 110 IF I=0 THEN GOTO 100
 120 LET P=P+.25
 130 IF P=R THEN GOTO 300
 200 LET I=USR 16514
 210 IF I=1 THEN GOTO 200
 220 LET P=P+.25
 230 IF P=R THEN GOTO 300
 240 GOTO 100
 300 PRINT P;" REVOLUTIONS COMPL
ETE"
 310 SLOW
```

Figure 8-7. Optical encoder Version 3 software.

```
  1 REM : ²<=² J NEW ²?²² <=² J NEW
 ²▓C SAVE <=² J NEW ²▓C LIST <=² J
 NEW ²▓C CONT T( FAST 64 STEP TA
N Y² PEEK ▓TAN
  10 LET IN=USR 16514
  20 PRINT "PULSE COUNT COMPLETE
"
```

Table 8-2
RX81 MACHINE CODE DOCUMENTATION
VERSION 3 OPTICAL ENCODING

ZX81/T-S2068 address	Code	Hex	Z80 Assembler	
(encoder input)				
16514/65268	14	0E	ld c,N	
16515/65269	2	02		(number of shaft revolutions)
16516/65270	219	DB	in a,N	
16517/65271	1	01		(select I/O board) (i.e. IN 1)
16518/65272	47	2F	cpl	
16519/65273	230	E6	AND N	
16520/65274	1	01		(mask for D0)
16521/65275	103	67	LD h,a	
16522/65276	6	06	LD b,N	
16523/65277	4	04		(pulses per rev)
16524/65278	219	DB	in a,N	
16525/65279	1	01		(select I/O board)
16526/65280	47	2F	cpl	
16527/65281	230	E6	AND N	
16528/65282	1	01		(mask for D0)
16529/65283	188	BC	CP h	
16530/65284	40	28	JR Z,N	(same)
16531/65285	248	F8		-8
16532/65286	219	DB	in a,N	
16533/65287	1	01		(select I/O board)
16534/65288	47	2F	cpl	
16535/65289	230	E6	AND N	
16536/65290	1	01		(mask for D0)
16537/65291	188	BC	CP h	
16538/65292	40	28	ZR Z,N	(same)
16539/65293	240	F0		-16
16540/65294	219	DB	in a,N	
16541/65295	1	01		(select I/O board)
16542/65296	47	2F	cpl	
16543/65297	230	E6	AND N	
16544/65298	1	01		(mask for D0)
16545/65299	188	BC	CP h	
16546/65300	40	28	JR Z,N	(same)
16547/65301	232	E8		-24
16548/65302	103	67	LD h,a	
16549/65303	16	10	DJ NZ,N	(same)
16550/65304	229	E5		-27
16551/65305	13	0D	DEC C	
16552/65306	32	20	JR NZ,N	(rev)
16553/65307	224	E0		-32
16554/65308	201	C9	ret	
(output control)				
16555/65309	62	3E	ld a,N	
16556/65310	1	01		(output D0 select) (i.e. D0=1, D1=2)

Table 8-2 (Continued)

16557/65311	211	D3	out N,a	
16558/65312	7	07		(select I/O board wired as 7)
16559/65313	201	C9	ret	

To activate these input or output routines the following BASIC commands are given:

To read encoder, "LET USR = 16514" or "RAND USR 16514" Subsitute the address "65268" in a TS2068 program.

To change number of shaft revolutions read, "POKE 16515,x", where x = the number of shaft revolutions.

To change number of pulses per revolution, "POKE 16523,x", where x = the number of pulses per revolution.

If encoder is wired to an input line other than D0, remember to change the mask by POKEing 16520, 16528, 16536 and 16544 with the appropriate binary number (i.e. D0=1, D1=2, D2=4, etc.).

To activate an output, "LET OUT = USR 16555" or "RAND USR 16555" Again, substitute the address "65309" in a TS2068 program.

After executing the selected machine code routine, the computer will return to the next statement following the BASIC command that activated the machine code routine. But remember, once you enter the encoder machine code routine the specified number of pulses must be read before the computer will exit the machine code routine. "STOP" or "BREAK" from the keyboard will not halt the computer in this situation.

---

statements have changed memory location (i.e., LET OUT = USR 16533).

Load the machine code for versions 1 and 2 as follows. This is the same procedure used in Chapter 3. First, enter "1 REM" statement containing at least 24 spaces. Then enter the program as listed in Table 3-2.

Enter RUN and as each address location is displayed, enter the appropriate code from Table 3-3. Now list the program and check the 1 REM statement by comparing it with Figures 8-5 or 8-6. If a symbol is incorrect, just correct it by POKEing the address directly. If there are several errors, repeat the entire code input process. After you are sure the REM statement is correct, delete lines 10 through 50 and enter the rest of the listing for version 1 or 2 (Figures 8-5 and 8-6 respectively).

Addresses 16515, 16519 and 16525 contain code 1 (hex 01) to select the proper input address (RX81 I/O boards are wired for input 1). If you add a second I/O board or otherwise change the board wiring, these addresses must be changed to match the proper address. All versions also contain the machine code for output. Address 16534 for versions 1 and 2 contains the code for output D0 (output line 1). Remember that this code is a binary number (i.e., 1 = line 1; 2 = line 2; 4 = line 3; 8 = line 4; 16 = line 5; etc.). And , address 16536 for versions 1 and 2 contains the code for selection of the output board (RX81 boards come wired as output 7).

To load the machine code version 3, first enter "1 REM" statement containing at least 46 spaces. Then enter the program as listed in Table 8-3. Remember the REM statement with the machine code must always be the first statement of the program of the program, no exceptions. (See Chapter 12 for a complete explanation)

Enter RUN and as each address location is displayed, enter the appropriate code from Table 8-4. List the program and check the contents of the REM statement against Figure 8-7. Make any corrections required. Then delete lines 10 through 50 and enter the remainder of Figure 8-7.

Similar to versions 1 and 2, addresses 16517, 16525, 16533 and 16541 in version 3 contains code 1 to select the proper input board. This version also contains the output routine with the output line address 16556 and the output board select at address location 16558. But, the pulses per revolution and shaft revolutions are contained within the machine code in version 3 as opposed to having them in the BASIC instructions as in versions 1 and 2. The number of shaft revolutions is at address location 16515 in version 3. Two (2) shaft revolutions are entered in this listing. This number can be changed at any time during the RUNning of this version by simply POKEing a new number. For example, if you wanted to count 100 revolutions you would ass the statement POKE 16515,100 at the appropriate location in your BASIC routine. Also, the number of pulses per revolution is contained within the machine code at address location 16523. So, as with the revolutions, is you had a different number of pulses per revolution you would have to POKE this address (16523) location with the appropriate number.

There is one more aspect of this machine code input routine that you must be familiar with for proper use. It has a feature which simplifies the arithmetic of adding up of counting successive pulses. The original RX81 machine code, explained first in Chapter 3, read each input pulse as the binary number identifying each data line (i.e., D7 = binary 128 or D6 = binary 64, etc.). However, to enable a rapid counting of input pulses, this machine code routine uses a "masking" technique. This masking technique converts each "high" or "on" pulse to a zero. Thus it rapidly counts each pulse, high or low, as a zero. The routine in Figure 8-7 (documented in Table 8-2) is written to accept input only from data line D0 (remember the binary number read from D0 is the number 1). Thus the routine reads each on or high pulse as a 1, immediately coverts it to a zero and counts all input as zeros. This masking takes place at address locations 16520, 16528, 16536 and 16544. So, if the routine was to read input from an encoder on data line D1 (coded as a binary 2) these four mask locations would each have a code 2 POKEd into them.

Now, with your I/O circuit hooked up to your computer and your software loaded, before you RUN the program, one caution. Remember that Version 3 cannot be STOPped or interrupted with a

Table 8-3
MACHINE CODE ENTRY PROGRAM

```
 1 REM    (with at least 46 spaces)
10 FOR N = 16514 TO 16559
20 PRINT AT 10,10; N
30 INPUT I
40 POKE N,I
50 NEXT N
```

Table 8-4
VERSION 3 CODE TABLE

ZX81/TS2068 Address	Code	Address	Code
16514/65268	14	16537/65291	188
16515/65269	2	16538/65292	40
16516/65270	219	16539/65293	240
16517/65271	1	16540/65294	219
16518/65272	47	16541/65295	1
16519/65273	230	16542/65296	47
16520/65274	1	16543/65297	230
16521/65275	103	16544/65298	1
16522/65276	6	16545/65299	188
16523/65277	4	16546/65300	40
16524/65278	219	16547/65301	232
16525/65279	1	16548/65302	103
16526/65280	47	16549/65303	16
16527/65281	230	16550/65304	229
16528/65282	1	16551/65305	13
16529/65283	188	16552/65306	32
16530/65284	40	16553/65307	224
16531/65285	248	16554/65308	201
16532/65286	219	16555/65309	62
16533/65287	1	16556/65310	1
16534/65288	47	16557/65311	211
16535/65289	230	16558/65312	7
16536/65290	1	16559/65313	201

Table 8-5
ROBOT ARM DEMONSTRATION ROUTINE, USING OPTICAL ENCODER

```
10 POKE 16558,6
20 POKE 16556,128
30 LET OUT = USR 16555
40 LET IN = USR 16514
50 POKE 16556,0
60 LET OUT = USR 16555
```

BREAK or any other keyboard entry because it is in a machine code only loop.

Finally, here is a practical application for the encoder project. If you have built the circuits described in the robot electronics chapter (Chapter 5), the following will operate the Right Elbow, down function (Relay 18). Starting with the REM statement in Figure 8-7, enter new BASIC statements as listed in Table 8-5. The first three lines (10 through 30) will turn the elbow motor on.

Line 40 will activate the pulse counting routine. Then, with the proper number of pulses counted (your choice), the last two statements (50 and 60) will turn the motor off.

TS2068 APPLICATION

In addition to this optical encoder circuit and software being compatible with the ZX81, TS1000 and TS1500, it can be made to work with the TS2068 by making some minor changes to the software. The RX81 I/O board is fully compatible with the TS2068 and will plug directly into the rear peripherial connector on the TS2068. The reason it works is because the ZX81/TS1000 and 1500 is a subset of the TS2068 peripherial connector pins. When aligned with the slot, the pins of the TS2068 correspond to the ZX/TS pinout with three exceptions. These three pins (RAM CS, ROM CS and 9 volts dc) are not used by the RX81 I/O board. So, welcome to the world of control with your TS2068.

The only change to the above software, required for the TS2068 involves the machine code addresses and the techniques you use to compile machine code and SAVE and LOAD the software.

Also, as described in Chapter 3, one basic difference with the TS2068 is that machine code cannot be stored in a REM statement like it can in the ZX81, TS1000 and TS1500. Also, a new command of "CLEAR" must be used to reserve space for the machine code. I will describe two different methods for LOADing and SAVEing the above optical encoder software on the TS2068. But first, I will briefly review the CLEAR command which is common to both methods. As explained on pages 268 and 269 of the TS2068 User Manual, entering the command "CLEAR 65267" reserves space for 100 bytes of machine code between addresses 65268 and 65368, inclusive. The command can either be entered in the immediate mode directly from the keyboard or included in your program.

The first method to compile the machine code is as follows. For versions 1 and 2 (Figures 8-5 and 8-6) modify the original machine code loading routine (Table 3-2) as follows. Begin the routine with the following instructions: "1 REM "rx81"" and "5 CLEAR 65267".

Then change line 10 to read: "10 FOR n = 65268 TO 65291". Now enter the rest of the routine, remembering to enter variables as lower case letters for the TS2068 (i.e. "i" and "n"). This requirement is for the easier comprehension of the human (you) as the computer will recognize a letter as the same variable whether it is a captial letter or a lower case letter.

Now run the routine and enter the TS2068 code from Table 3-3 starting with address 65268 and ending with address 65291. To check and see if it has been entered correctly, enter the routine in Table 8-6 and RUN it with a "GOTO 100" before

Table 8-6
TS2068 CODE ENTRY CHECK ROUTINE

```
100 FOR n = 65268 TO 65291
110 PRINT n; " "; PEEK n
120 NEXT n
```

entering the remainder of Figures 8-5 or 8-6.

If the code is correct, enter the routine in Figures 8-5 or 8-6 omitting lines 1, 30 and 310 and changing the USR address (line 50) to USR 65268. You already have a REM statement containing the program title and the FAST and SLOW commands are not used in the TS2068.

To enter version 3 using this same technique, repeat the above, entering lines by first entering lines 1, 5 and 10, while only changing the address locations. In other words, change line 10 to read "FOR n = 65268 TO 65313". RUN the routine entering the code in Table 8-4, but you will start with address 65268 and end with 65313. Check the code by entering the routine from Table 8-6 (remember line 100 is now "FOR n = 65268 TO 65313"). If code has been entered correctly, enter the rest of Figure 8-7, changing only the USR instruction to USR 65268 vice USR 16514. Delete all statements beyond statement 20.

The only drawback to loading the machine code using this method is that the machine code and the BASIC program must each be SAVEd and LOADed separately. To SAVE machine code for Versions 1 or 2 enter the following: "SAVE "rx81" CODE 65268,24". To SAVE the machine code for Version 3, enter the following: "SAVE "rx81" CODE 65268,46". Then, to SAVE the BASIC program for each, enter: "SAVE "rx81"". To LOAD the machine code from the tape enter the following: "LOAD "rx81" CODE 65268". To LOAD the BASIC program enter the following: "LOAD "rx81"".

The second method for adapting these programs to the TS2068 eliminates the requirement to SAVE and LOAD the machine code separate from the BASIC program. This is done by putting the machine code into a DATA statement and then READing it into the proper address locations when the program is RUN.

For versions 1 and 2 begin the BASIC program (Figures 8-5 and 8-6) with the program lines in Table 8-7.

Table 8-7
FIRST FOUR PROGRAM STATEMENTS, SECOND METHOD
USING THE TS2068

```
    1 REM "rx81"
    2 CLEAR 65267: FOR n = 65268 TO 65291
    3 READ d: POKE n,d: NEXT n
    4 DATA 219,1,47,79,219,1,47,185,32,246,219,1,47,185,32,240,
6,0,201,62,1,211,7,201
```

Then enter the rest of Figures 8-5 or 8-6, starting with statement 20 and omitting the FAST and SLOW statements. Remember to change the USR address (line 50) to USR 65268. Note that the numbers in the DATA statements are the code from Table 3-3 (Chapter 3). This program can now be SAVEd and LOADed in a single step using the commands SAVE "rx81" and LOAD "rx81". When the program is RUN the machine code is automatically loaded into the proper address locations.

Version 3 (Figure 8-7) is created for the TS2068 in the same manner. The only difference involves the addresses (line 2) and the DATA line (line 4). The addresses in line 2 would be "65268 TO 65313". The DATA in line 4 would be that contained in Table 8-4. Remember that the USR address is 65268 for this TS2068 version.

May the encoder be with you. Happy controlling!

Chapter 9
DIGITAL VOICE OUTPUT

This project is my most favorite because it is fun. Yes the robot project is fun, but a robot talking is far out! This voice output circuit opens up a whole new world of interaction between human and machine. In keeping with the theme of the book, this application uses low cost hardware which is simple to adapt. It uses the National Semiconductor "Digitalker" chips which are among the clearest and most understandable voice synthesizers produced. It stores human voice in digital form, on read only memory (ROM) chips, and reproduces words on demand from the computer. One big advantage of the system is that the computer only controls the system, with no computer memory wasted by storing the actual voice data. This is not a direct text-to-talk conversion, a system which costs hundreds of dollars. The basic chip set consists of three integrated circuit chips costing about $30 (Figure 9-1). Two of the chips are ROMs containing the digitized voice. The third chip, a Speech Processor Chip (SPC), acts as a ROM controller and converts the stored digitized voice to recognizable speech. This set produces 144 words (several "words" are word endings and pauses). It is expandable by 131 additional words with the addition of two more ROM chips available for about $25 (Figure 9-2). The manufacturer states that more ROM chips will become available in the future.

The system I will describe in this chapter uses all four ROM chips and because of computer control using the software I will describe, the 275 words stored in ROM are expanded to provide an even larger vocabulary. This versatility made possible by the computer control can best be illustrated with the following example. The expression "gee-whiz" can be produced as follows. First, sound for the letter "G" is produced. This is followed without pause by the word "weight". However, the entire word is not allowed to be completely processed and is interrupted during processing after the "wei" portion is produced. This is followed, again without pause, by the word "is". The result produces the sounds for "G" "wei" "is" which sounds to the listener exactly like "Gee-whiz". More than 100 examples of the expanded vocabulary are included in the software section.

CIRCUIT DESCRIPTION

This project uses the 8255 PPI input/output circuit described in Chapter 3 for interface control with your computer. Figure 9-3 depicts the logic used to provide the signal which activates the Digitalker speech processor chip (SPC). This

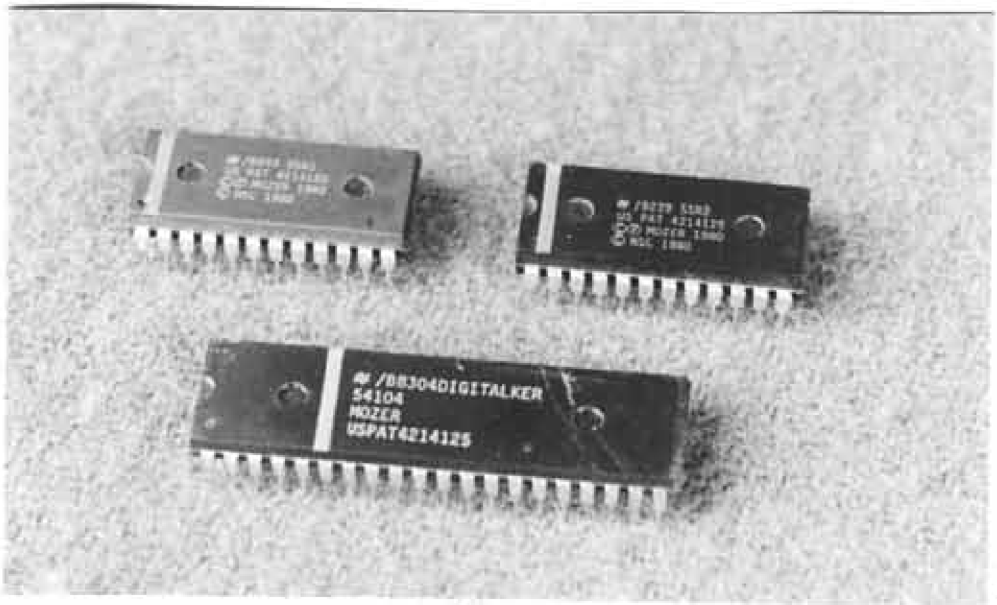


Figure 9-1. Three chip Digitalker Read Only Memory (ROM) set.

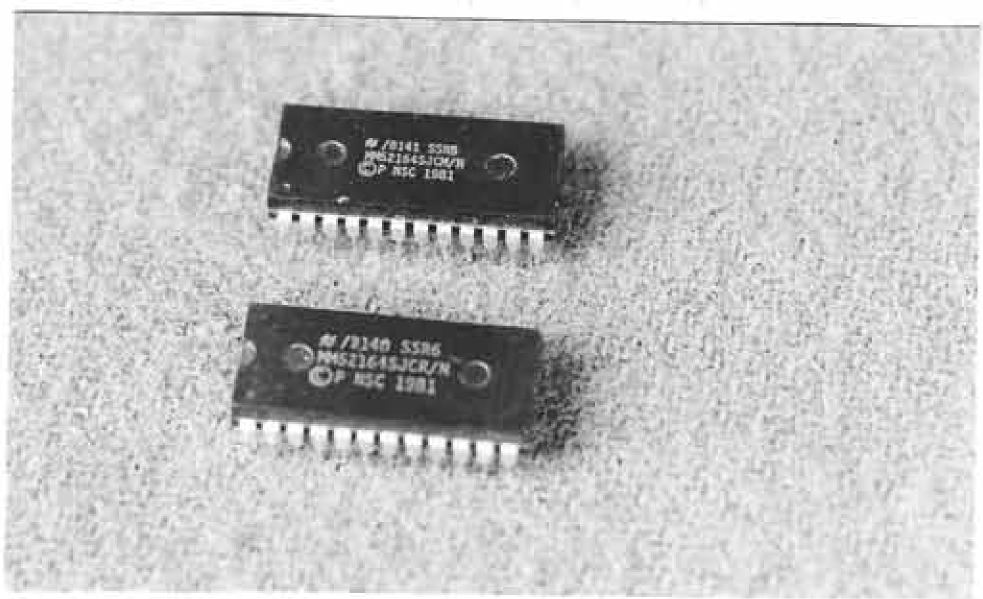
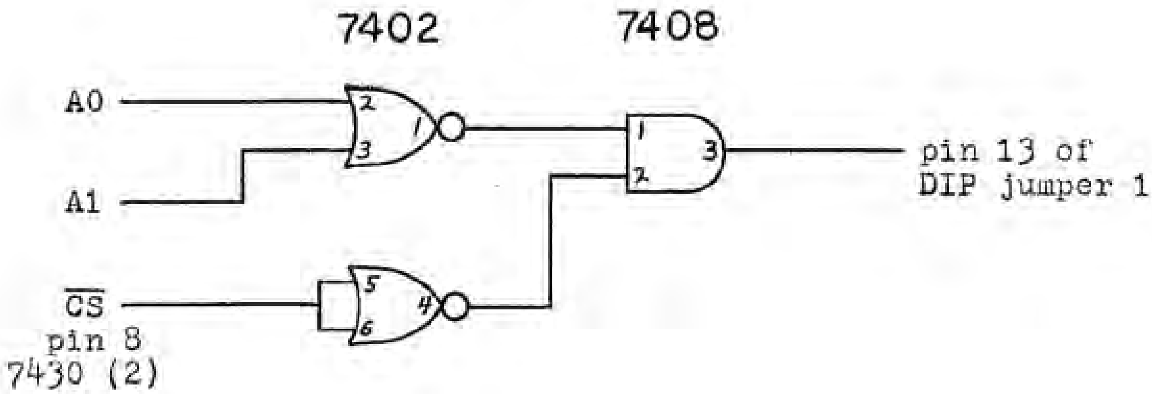


Figure 9-2. Two chip Digitalker expansion Read Only Memory (ROM) set.



(pin 7 GND & pin 14 +5v for 7402 & 7408)

Figure 9-3. Output pulse coding from 8255, Port A for input to Digitalker circuit.

signal is activated by the same signal that activates the 8255 chip. Port A of the 8255 provides the signal for the selection of the proper word from the ROM chips. Port B (using only data line B0) provides the signal for selection of the proper ROM chip set (remember there are two ROM chip sets in this circuit). More on this process in the software section.

If you have prepared the 8255 I/O circuit as described in Chapter 3, you are ready to proceed with the construction of the Digitalker circuit. If you have not, go back to Table 3-14 (Chapter 3) and wire up the proper connections to the 16 pin DIP jumper socket (jumper 1) on the 8255 I/O board. This DIP jumper ribbon cable is the only connection required to the Digitalker circuit board. In other words, the Digitalker board does not need to be plugged into the Expansion Board, but only needs to be close enough to the 8255 board (which is plugged into the Expansion Board) for the jumper cable to reach. As can be seen in Figure 5-5 (Chapter 5) I have plugged the robot's Digitalker board into the Expansion Board, next to the 8255 I/O board, but there are no electrical connections. A piece of circuit board with no metal surfacing is just plugged into an expansion connector to hold the board securely.

Assembly

A complete parts list is presented in Table 9-1. Figure 9-4, Digitalker schematic, and Figure 9-5, a photo of the circuit board layout, should provide enough information to complete assembly. First lay out the DIP sockets on the board like they are in Figure 9-5, then wire up the components as illustrated in Figure 9-4. The DIP jumper should be wired in accordance with the jumper pin out (Table 3-14) and the connections on the left side of the schematic (Figure 9-4). The speaker can either be wired directly to the speaker output line from LM386 N-1 (don't forget to wire the other side of the speaker coil to ground from the board) or using the RCA type phono connectors. Some builders have experienced problems with

Table 9-1
DIGITALKER CIRCUIT PARTS LIST

(1) DT1050 Digitalker chip set (JAMECO # DT1050)
(1) DT1057 expansion chip set (JAMECO # DT1057)
(1) LM386 N-1 amp IC
(1) LM324 op amp IC
(1) EAC D1C05 DIP reed relay 5 volt coil (Digi-Key # Z619-ND)
(1) 4 MHz crystal (JAMECO # CY3A)
(4) 28 pin DIP sockets
(1) 40 pin DIP socket
(2) 14 pin DIP sockets
(1) 8 pin DIP socket
(1) 8 ohm speaker
(1) 50K ohm Linear Taper potentiometer (JAMECO # 43P-50K)
(1) 10 ohm 1/4 watt resistor
(1) 1.5K ohm 1/4 watt resistor
(1) 9.1K ohm 1/4 watt resistor
(1) 10K ohm 1/4 watt resistor
(1) 1M ohm 1/4 watt resistor
(1) 20uF capacitor
(1) 10uF capacitor
(9) 0.1uF capacitor
(1) .05uF capacitor
(1) 50pF capacitor
(1) 20pF capacitor
(1) 2N2222 transistor
(1) 16 pin DIP jumper, single ended (Digi-Key # R112-12-ND)
(3) 1N914 switching diodes
(1) 4.5" x 6.5" circuit board (JAMECO and VECTOR # 8001)
(Budget Robotics & Computing may begin distributing a bare version of this board)
(1) RCA type phono jack
(1) RCA type phono plug

---

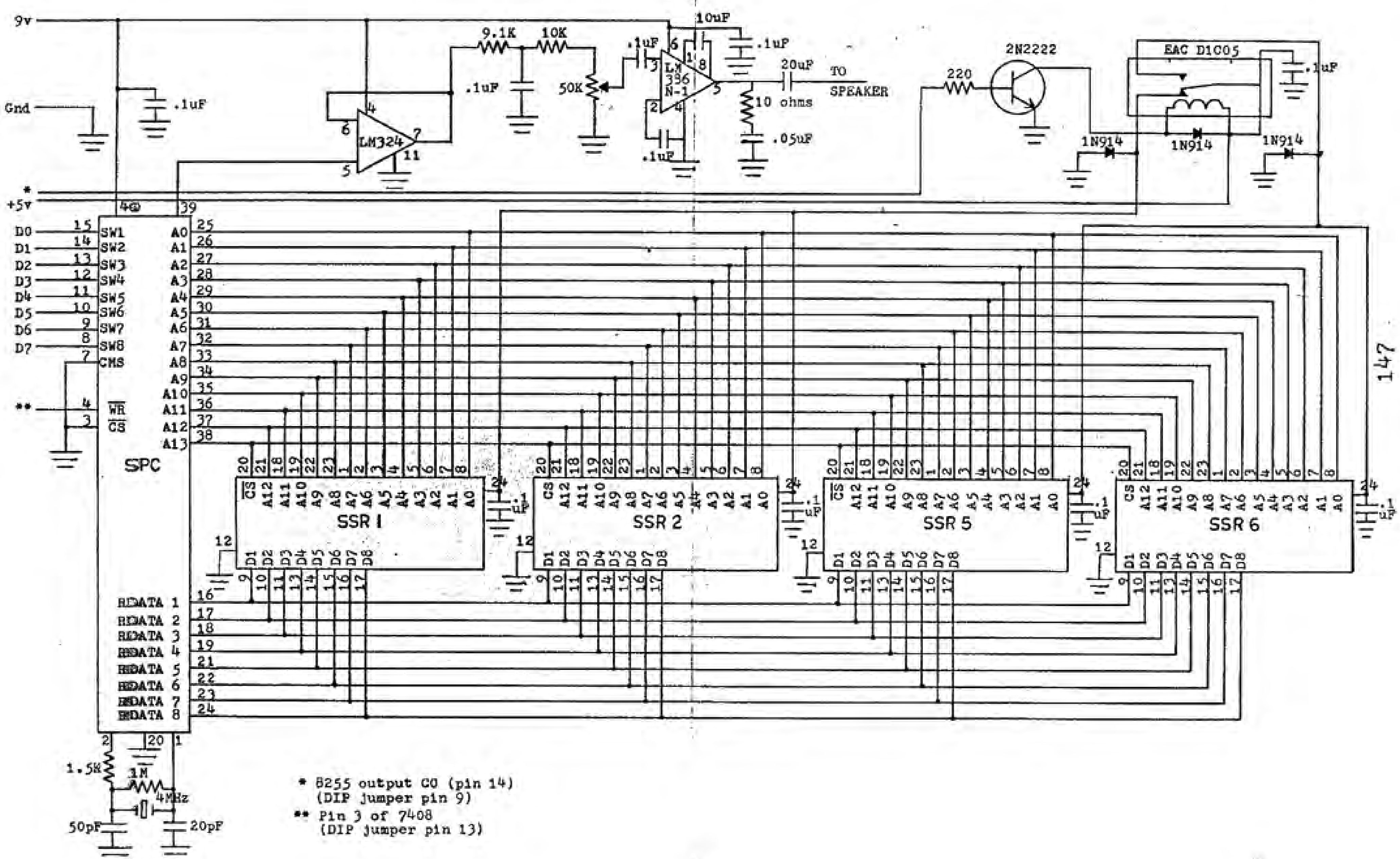
laying out the ROM chips as depicted in Figure 9-5. The alternate method is to mount the ROMs all in a row, extending out from the SPC, and parallel to the SPC. Be sure to mount the 50K potentiometer so that the screw head slot is accessible because this is the volume control for the sound output of the amplifier. If you do not think you can handle the assembly of this circuit on your own, it may be possible to buy the voice synthesizer accessory from JAMECO (Part # JE520CM) and modify it to this circuit. Also, Budget Robotics & Computing may manufacture a bare board version.

A word of caution on handling the Digitalker IC chips. Of course, you should take normal precautions against damage from static electricity. However, the SPC chip is especially susceptible to damage. If you do damage the SPC, a new chip is available from JAMECO,as the MM54104, for about $12.

SOFTWARE

Now the fun begins. Remember to lower RAMTOP if necessary (see Chapter 3, 8255 PPI section) and set up the 8255 I/O device for output (POKE 32763,128) before proceeding with writing

Figure 9-4. Digitalker circuit schematic.



* 8255 output C0 (pin 14)
(DIP jumper pin 9)

** Pin 3 of 7408
(DIP jumper pin 13)

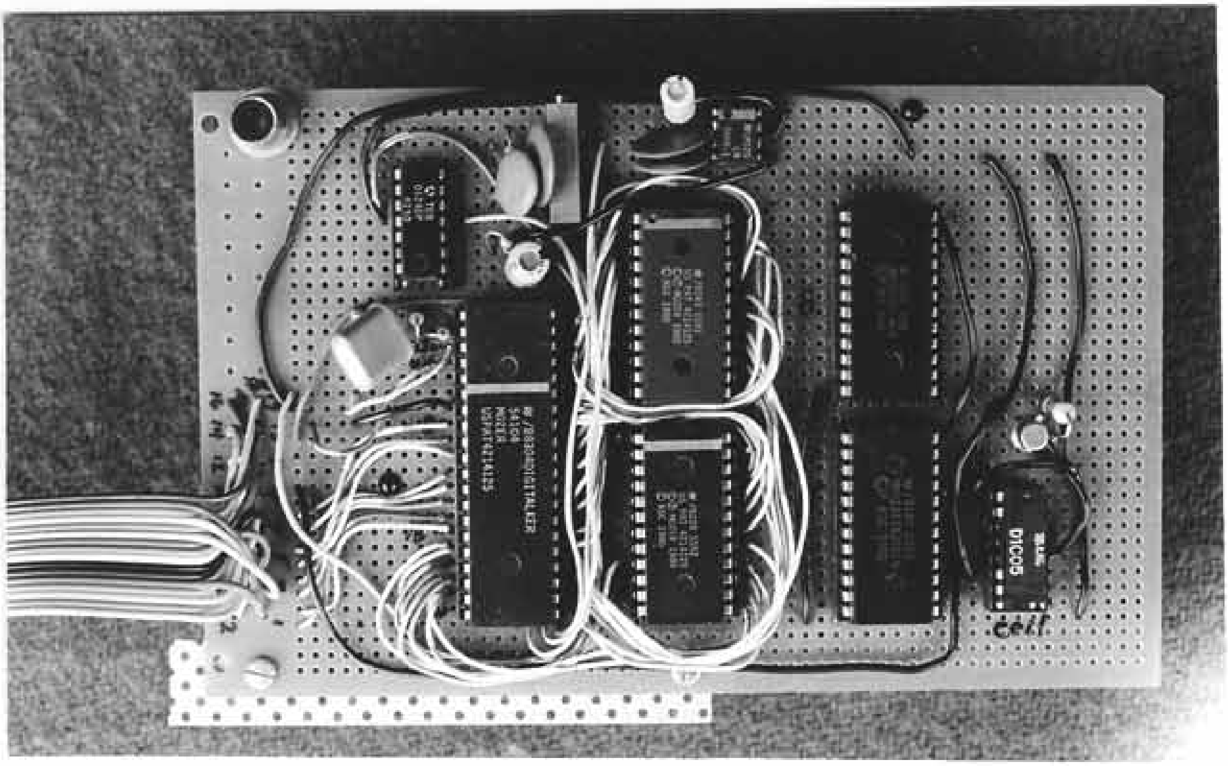


Figure 9-5. Component side of Digitalker circuit board.

control software. In order to produce a word two commands or POKEs are necessary.

First you must POKE the command to select one of the two ROM chip sets. You POKE 32762 with a "0" (POKE 32762,0) to select ROMs 1 & 2. You POKE 32762 with a "1" (POKE 32762,1) to select ROMs 5 & 6. However, the circuit automatically sets to select ROMs 1 & 2 when you power up. This is because the relay activates the first ROM set until you POKE the circuit with a "1". You then have to POKE a "0" to turn the relay coil off.

The second required command both selects the proper word from ROM and activates the SPC to produce a word. These words are selected by assigned number through 8255 port A. In other words to select the second word in the first ROM chip set you "POKE 32760,1". This word is the word "one". The first word in this ROM set is actually not a word but a phrase in a female voice. The the phrase "This is Digitalker". All other words are in a male voice.

The best illustration of the capability of this system is accomplished by the routine listed in Table 9-2. This program will cycle through the entire vocabulary alternating from one ROM set to the other and back, until all 275 words are produced. It will then automatically start over from the beginning. One complete cycle will take about five and one-half minutes. Once started, you can only stop the routine with a "BREAK" from the computer keyboard.

There is one aspect to this control method which I have not yet explained. You may have noticed the PAUSE statements in the above program. These are required to allow the word to be processed and spoken before signalling the SPC for the next word. If you don't pause, the words will run together and the end of one word will be "stepped on" by the beginning of the

Table 9-2
VOICE DEMONSTRATION SOFTWARE ROUTINE

```
100 FOR K = 0 TO 143
110 POKE 32762,0
120 PAUSE 70
130 POKE 32760,K
140 PAUSE 70
150 IF K = 143 THEN GOTO 220
160 IF K > 130 THEN NEXT K
170 POKE 32762,1
180 PAUSE 70
190 POKE 32760,K
200 PAUSE 70
210 NEXT K
220 GOTO 100
```

next word. This aspect of the system can also be used to advantage in producing additional vocabulary. Remember the "Gee-whiz" example I cited in the beginning of this chapter? A working depiction of how it works is listed in Table 9-3.

A complete listing of the vocabulary of both ROM chip sets plus additional vocabulary made up of parts of several words is found in Table 9-4. The entire vocabulary has been alphabetized, but you can pick out the words in each ROM set as follows. If a word entry ends in the third column (with the "Word POKE" column) look to the second column (ROM POKE) and you will see either a "0" or a "1". If it is a "0" then the word is contained in the first ROM chip set (ROMs SSR 1 & SSR 2). If it is a "1" the word is from the second ROM set (ROMs SSR 5 & SSR 6).

Before we move on to the voice recognition chapter, a few words on the robot voice application. You may have wondered why I refer to the robot as "H.E.N.R.Y." instead of "Henry". It is because the word "Henry" is not in the vocabulary. Therefore, when he recites his name he says, "I am H. E. N. R. Y.", by spelling his name. And you thought it was some clever acronym! There are many words and phrases incorporated into the robot software (Table 5-10) including the above example routines. May the digitized voice be with you.

Table 9-3
ANALYSIS OF "GEE-WHIZ" SPEECH DEMO ROUTINE

```
10 POKE 32762,0          (selects ROMs 1 & 2)
20 POKE 32760,38         (activates word "G")
30 PAUSE 18              (waits for word to be spoken)
40 POKE 32760,143        (activates word "weight")
50 PAUSE 4               (allows only "wei" to be spoken)
60 POKE 32760,96         (activates word "is")
                         (The result is the word "Gee-whiz".)
```

Table 9-4
DIGITALKER VOCABULARY SOFTWARE COMMANDS

Word	ROM POKE 32762,	Word POKE 32760,	PAUSE	Word POKE 32760,	PAUSE	Word POKE 32760,	PAUSE
abort	1	0					
add	1	1					
adjust	1	2					
again	0	58					
alarm	1	3					
alert	1	4					
all	1	5					
am	0	59	10	70			
ampere	0	59					
and	0	60					
are	0	49					
ask	1	6					
assistance	1	7					
at	0	61					
attention	1	8					
bee	0	33					
before	0	33	8	4			
below	0	33	9	103			
berate	0	33	9	25			
brake	1	9					
button	1	10					
by	1	11					
buy	1	11					
call	1	12					
cancel	0	62					
case	0	63					
caution	1	13					
cent	0	64					
centi	0	72					
charge	1	14					
check	0	73					
circuit	1	15					
clear	1	16					
close	1	17					
coffee	0	74	3	114	21	36	
comma	0	74					
complete	1	18					
con	0	74	5	115			
connect	1	19					
continue	1	20					
control	0	75					
copy	1	21					
correct	1	22					
court	1	93	13	34			
danger	0	76					
date	1	23					
day	1	24					
decrease	1	25					

Table 9-4 (Continued)

defeat	0	35	6	82		
defeat	0	77	6	82		
degree	0	77				
deposit	1	26				
dial	1	27				
dig	0	77	9	70		
divide	1	28				
doll	0	78	12	70		
dollar	0	78				
door	1	29				
down	0	79				
dull	0	78	12	70		
east	1	30				
ed	1	31-34				
emergency	1	35				
end	1	36				
enter	1	37				
entry	1	38				
equal	0	80				
er	1	39				
error	0	81				
evacuate	1	40				
exit	1	41				
fail	1	42				
failure	1	43				
farad	1	44				
fast	1	45				
faster	1	46				
fee	0	82	15	70		
feet	0	82				
fifth	1	47				
five	1	48				
first	1	49				
floor	1	50				
flow	0	83				
forward	1	51				
friend	1	49	14	36		
from	1	52				
fu-ee	0	84	14	36		
fuel	0	84				
gal	0	85	11	70		
gallon	0	85				
gas	1	53				
gee-whiz	0	38	18	143	4	96
get	1	54				
go	0	86				
going	1	55				
gram	0	87				
grape	0	88	14	118	--	70
great	0	88				
greater	0	89				
gray	0	88	14	70		
grey	0	88	14	70		

Table 9-4 (Continued)

half	1	56					
have	0	90					
hell	1	57	9	34			
hello	1	57					
help	1	58					
helper	1	58	20	39			
hertz	1	59					
high	0	91					
higher	0	92					
ho-ho-ho	1	60	10	34	8	60	10
(cont)		34	8	60	10	34	8
hold	1	60					
holder	1	60	23	39			
hour	0	93					
hurt	1	59	16	34			
in	0	94					
inch	0	95	12	70			
inches	0	95					
incorrect	1	61					
increase	1	62					
intruder	1	63					
is	0	96					
it	0	97					
just	1	64					
key	1	65					
kill	0	98	9	70			
kilo	0	98					
knee	1	75	14	34			
left	0	99					
less	0	100					
lesser	0	101					
level	1	66					
leveler	1	66	22	39			
limit	0	102					
load	1	67					
lock	1	68					
low	0	103					
lower	0	104					
mark	0	105					
me	0	106	9	70			
meat	0	106	12	70			
meet	0	106	12	70			
meg	1	69					
mega	1	70					
meter	0	106					
mic	1	71	14	34			
micro	1	71					
mile	0	107					

Table 9-4 (Continued)
DIGITALKER VOCABULARY SOFTWARE COMMANDS

Word	ROM POKE 32762,	Word POKE 32760,	PAUSE	Word POKE 32760,	PAUSE	Word POKE 32760,	PAUSE
milli	0	108					
mine	0	109	14	70			
mini	0	110	9	32			
minus	0	109					
minute	0	110					
mi-ti	0	107	16	70	6	139	18
(cont)		70					
more	1	72					
move	1	73					
mover	1	73	29	39			
my	0	107	16	70			
nano	1	74					
near	0	111					
neck	1	76	16	34			
need	1	75					
next	1	76					
no	1	77					
normal	1	78					
north	1	79					
not	1	80					
notice	1	81					
numb	0	112	16	70			
number	0	112					
of	0	113					
off	0	114					
oh	1	82	12	34			
oh-oh	0	113	10	67	15	46	
ohms	1	82					
on	0	115					
onward	1	83					
open	1	84					
opener	1	84	28	39			
operator	1	85					
or	1	86					
ouch	0	93	8	73	5	70	
out	0	116					
over	0	117					
parenthesis	0	118					
pass	1	87					
passer	1	87	23	39			
peanut	0	47	8	112	8	20	3
(cont)		70					
pico	1	89					
place	1	90					
please	0	120					
plus	0	121					
point	0	122					
pound	0	123					

Table 9-4 (Continued)

press	1	91			
pressure	1	92			
pulses	0	124			
quart	1	93	13	34	
quarter	1	93			
range	1	94			
rate	0	125			
re	0	126			
reach	0	126	12	73	4	70
reach	1	95				
reacher	1	95	22	39		
read	0	127	10	70		
red	0	127	10	70		
ready	0	127				
receive	1	96				
receiver	1	96	29	39		
record	1	97				
recorder	1	97	30	30		
remark	0	126	7	105		
replace	1	98				
reverse	1	99				
right	0	128				
room	1	100				
safe	1	101				
sea	0	34				
second	0	130				
secure	1	102				
see	0	34				
select	1	103				
selector	1	103	30	39		
send	1	104				
sender	1	104	23	39		
service	1	105				
set	0	131				
sex	0	130	14	129		
sexy	0	130	14	129	8	36
side	1	106				
sis	1	113	10	34		
slow	1	107				
slower	1	108				
smoke	1	109				
smoker	1	109	24	39		
sorry	0	131	7	116	6	126
south	1	110				
space	0	132				
speed	0	133				
ss	0	129				
stand	0	136	7	60		
star	0	134				
start	0	135				
starter	0	135	23	1	39	
station	1	111				
stop	0	136				

Table 9-4 (Continued)

switch	1	112				
switcher	1	112	25	39		
system	1	113				
test	1	114				
tester	1	114	22	39		
th	1	115				
than	0	137				
thank	1	116				
the	0	138				
third	1	117				
thirsty	0	13	10	129	10	51
this	1	118				
tie	0	139	18	70		
time	0	139				
to	0	2				
too	0	2				
total	1	119				
try	0	140				
turn	1	120				
up	0	141				
use	1	121				
user	1	121	22	39		
uth	1	122				
volt	0	142				
wait	0	143				
waiting	1	123				
warn	1	124	11	34		
warning	1	125				
watt	1	125	11	34		
weight	0	143				
west	1	126				
what	1	125	11	34		
which	1	127				
why	0	56				
wind	1	128	14	34		
witch	1	127				
window	1	128				
yes	1	129				
you	0	52				
zero	0	31				
zone	1	130				
zero	0	31				
one	0	1				
two	0	2				
three	0	3				
four	0	4				
five	0	5				
six	0	6				
seven	0	7				
eight	0	8				
nine	0	9				
ten	0	10				

Table 9-4 (Continued)

eleven	0	11
twelve	0	12
thirteen	0	13
fourteen	0	14
fifteen	0	15
sixteen	0	16
seventeen	0	17
eighteen	0	18
nineteen	0	19
twenty	0	20
thirty	0	21
forty	0	22
fifty	0	23
sixty	0	24
seventy	0	25
eighty	0	26
ninety	0	27
hundred	0	28
thousand	0	29
million	0	30
a	0	32
b	0	33
c	0	34
d	0	35
e	0	36
f	0	37
g	0	38
h	0	39
i	0	40
j	0	41
k	0	42
l	0	43
m	0	44
n	0	45
o	0	46
p	0	47
q	0	48
r	0	49
s	0	50
t	0	51
u	0	52
v	0	53
w	0	54
x	0	55
y	0	56
z	0	57
this is Digitalker	0	0

Chapter 10
VOICE CONTROL INPUT

Voice input or voice control opens yet another door on the man/computer interface possibilities. This project uses a minimum of hardware, requiring only a simple battery powered amplifier and microphone put together from a few dollars worth of Radio Shack parts. The heart of this project is a machine code routine developed by Mr. Brad Bennett (Advanced Interface Designs, P.O. Box 1350, State College, PA 16801). The routine I will describe here is the basis for, but not the same routine included in, the 13K robot control software already presented in Chapter 5 (Tables 5-10 and 5-11). The primary reason that I will not present the version used in the 13K robot software is because the routine is too lengthy to be inserted into the computer via the keyboard. To attempt manual insertion of that machine code routine would be a very frustrating task involving input of decimal codes for over one thousand addresses. Any error, if not detected before the program was run would destroy the entire program. The routine of Mr. Bennett's that I will describe, requires the entry of 251 such codes, which is no small task in itself. The larger applications oriented routine that is used in the 13K robot program is available commercially on cassette including complete documentation. More information on that program will be provided at the end of this chapter.

Before proceding, let me caution you that this is not a speech recognition project with a high rate of accuracy. You can buy those systems for big bucks. This project is intended to educate and demonstrate the kinds of things that can be done with speech recognition. The application in the robot project accomplishes that purpose.

The speech recognition program has three basic parts. One is a speech input routine where a "voice print" in the form of a histogram is produced for each word spoken. It is a simple time versus amplitude depiction. The second part of the program is a file system that lets you create and store up to ten separate words. The third part of the program is the speech recognition routine. It compares the newly spoken word with those on file and displays the best match.

HARDWARE

The only hardware required is a small single circuit board with microphone and amplifier which is connected to the "EAR" input on the computer. The parts listed in Table 10-1 can be obtained from Radio Shack or parts or an assembled board can be obtained from G. Russell Electronics.

Table 10-1
AMPLIFIER CIRCUIT PARTS LIST

(1) Experimenter grid board (Radio Shack 276-158)
or (1) bare silk screened circuit board (G. Russell Electronics)
(1) Crystal microphone element (Mouser 25LM025)
(1) 8 pin DIP socket (Radio Shack 276-1995)
(1) 3 conductor 3.5mm audio jack (Radio Shack 274-249)
(1) TL081 or TL091 op amp (Radio Shack 276-1716 or 276-1745)
(1) DPDT micro miniature toggle switch (Radio Shack 275-626)
(2) 9 volt battery clips (Radio Shack 270-325)
(2) 9 volt batteries (Radio Shack 23-464)
(1) 10M ohm 1/4 watt resistor
(1) 470K ohm 1/4 watt resistor
(3) 10K ohm 1/4 watt resistors
(1) 1K ohm 1/4 watt resistors
Three feet 22 guage (or smaller) hookup wire

Assemble the board in accordance with Figure 10-1 (Component layout) and Figure 10-2 (Schematic). To use the board, plug your cassette recorder cable between the jack on the board and the ear connector on your computer and turn the amplifier on by turning on the board's toggle switch. Figure 10-3 shows the amplifier "ear" hanging on H.E.N.R.Y.

SOFTWARE

Table 10-2 is a listing of the complete speech recognition program. Before entering this listing into your computer, you must first compile the machine code contained in the "1 REM" statement. First, type into the computer a 1 REM statement containing at least 270 spaces. Then enter the listing from Table 10-3. RUN this program and INPUT each decimal code entry from Table 10-4 as the appropriate address is displayed on the screen. After you enter the last code (for address 16770) exit this program by inputting any non-numeric code (such as an A). Then check your work by entering "PRINT USR 16758". The result displayed should be "64".

If you got the correct answer, then enter the rest of the program from Table 10-2. If you did not get the correct answer, you must find the error in the machine code. Do so by entering the routine in Table 10-5. RUN this BASIC routine and it will dump forty bytes of memory at a time. Compare the decimal code displayed with that in Table 10-4. The next succesive forty bytes can be displayed by inputting any number and pressing ENTER. When an error is found, note the location and when all errors are found, enter the correct value(s) by POKEing them directly into each address. You can abort the routine at any time by entering a non-numeric character.

Before RUNning the program, SAVE at least one copy on tape. Then you can connect the amplifier, turn it on and RUN the program. The menu should appear and you can make your choice. Menu selection 1. provides a histogram of each word immediately after you say it. 2. asks you to type in a word (up to ten letters long) and then waits for you to say it, after you press ENTER. You then pronounce the word, and the routine repeats until you have pronounced it eight times. A "Y" repeats the

Figure 10-1. Voice input amplifier component layout.

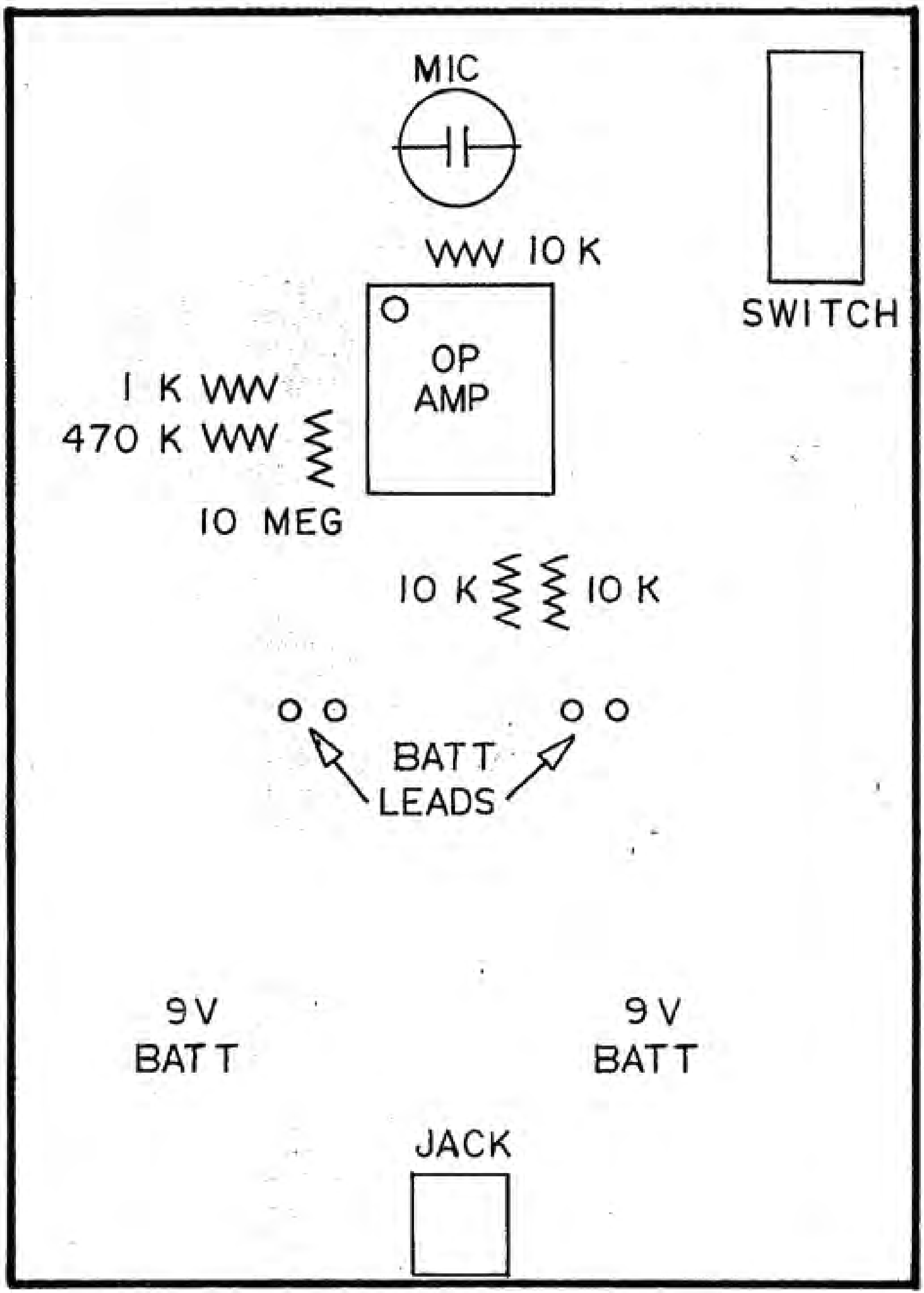
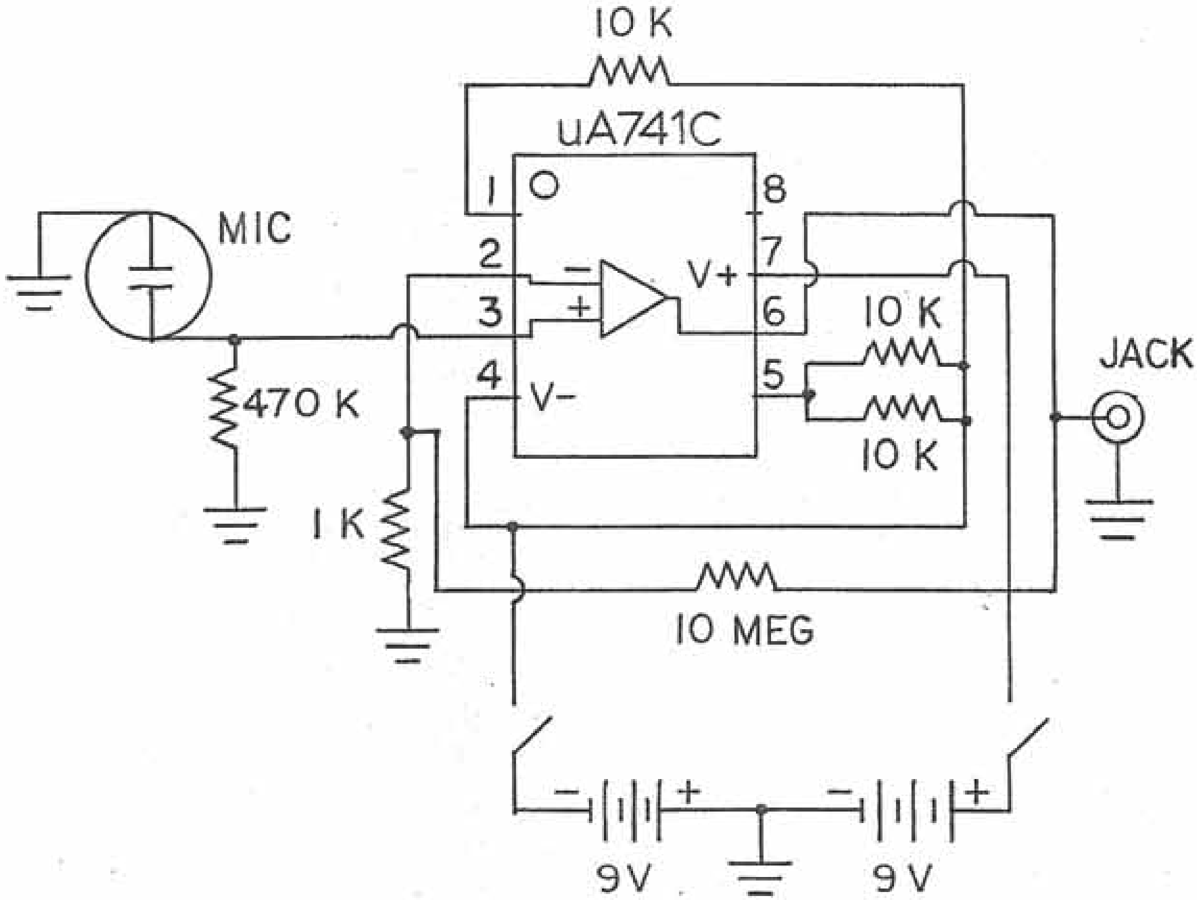


Figure 10-2. Voice input amplifier circuit schematic.



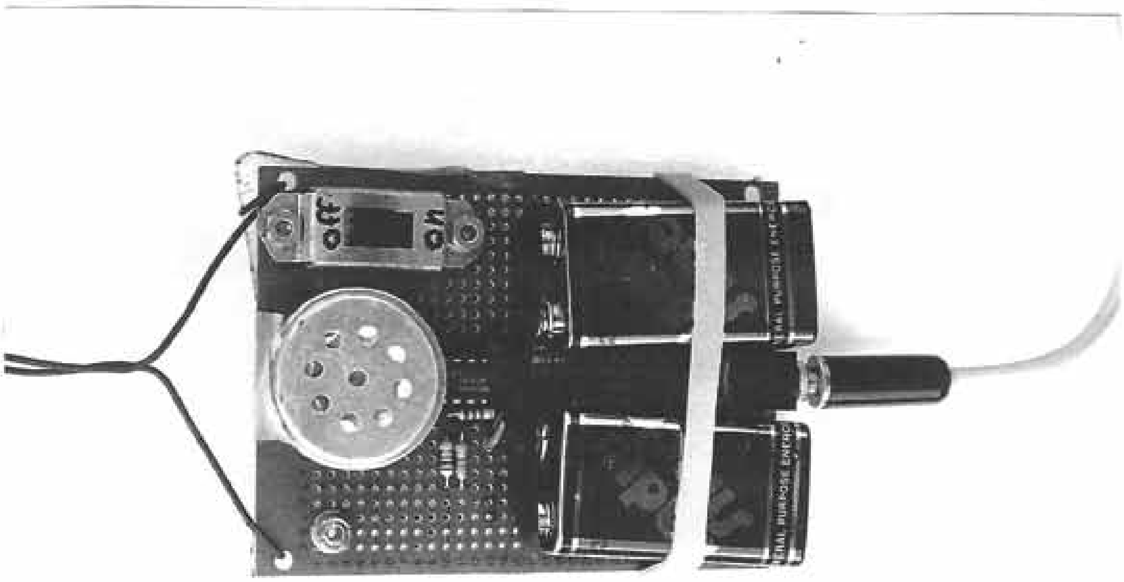


Figure 10-3. Amplifier "ear" which hangs on H.E.N.R.Y.

Table 10-2. Speech recognition program.

```
   1 REM        5 ?■ COPY Q ■7TAB
 ■RND■ COPY 5 ?: RETURN GOSUB ?
PAUSE ■RNDG GOSUB ? IF ■RNDO■COS
 GOSUB ? IF ■RNDI )>= COPY .?■CO
5 GOSUB ? PAUSE ■RND?■RND5 ?: 2R
NDLN INT RND7£14 SAVE TAN ■YF?VA
L STR$ FAST LN ■" LPRINT SGN AT
■TAB OR RNDTAN ■RND5■?U■? ■?) ?5T
R$ )■ ;SGN ,?<■4 POKE TAN ■C?5■?
LN AINKEY$? ■Y?■4 PRINT U■?■RND
5??■X4 UNPLOT )C? FOR GOSUB ■TAN
 VAL Y■ ) ? FOR ■ FOR X74 SAV
E ■■■ACS ,ACS .■4 SAVE AT TAN 5■
?■ COPY VAL ) ?: ■RND,■K ■JW■K ■Y
 COPY ??<■4 LOAD ?AT ■M**?K■ ??M■
?■Y■■4USR TAN 5■RND■■ INPUT ■7■4
 CLS ?TAN

   2 REM  SPEECH RECOGNITION
          COPYRIGHT 1983
          BRAD BENNETT
   5 IF USR 16758<>64 THEN PRINT
 AT 10,10;"BAD LOAD"
  10 DIM C(1)
  15 DIM T$(10,10)
  20 CLS
  25 PRINT AT 7,1
  30 PRINT TAB 12;"MENU"
  35 PRINT TAB 6;"1. VOICEPRINT
DISPLAY"
  40 PRINT TAB 6;"2. VOICEPRINT
FILE"
  45 PRINT TAB 6;"3. RECOGNITION
"
  50 PRINT TAB 6;"4. CLEAR FILES
"
  55 PRINT TAB 6;"5. DISPLAY STR
ING FILE"
  60 PRINT TAB 6;"6. STOP"
  62 PRINT ,,,,,,TAB 8;"INPUT SE
LECTION"
  65 FAST
  70 INPUT S
  75 IF S<=6 THEN GOTO S*200
  80 GOTO 70
 199 REM **VOICE PRINT DISPLAY**
 200 RAND USR 16520
 205 LET K=22528
 210 CLS
 215 POKE 16577,INT (K/256)
 220 POKE 16576,K-256*INT (K/256
)
 225 RAND USR 16575
 230 PRINT AT 2,20;"AGAIN? (Y/N)
"
 235 SLOW
 240 IF INKEY$="" THEN GOTO 240
 245 FAST
 250 LET B$=INKEY$
 255 IF B$="N" THEN GOTO 20
 260 IF B$="Y" THEN GOTO 200
 265 IF B$="5" AND K<=22719 THEN
LET K=K+1
 270 IF B$="8" AND K>=22529 THEN
LET K=K-1
```

Table 10-2 (Continued)

```
 275 GOTO 210
 399 REM ***VOICEPRINT FILE***
 400 CLS
 405 PRINT AT 10,1;"ENTER STRING
 TO BE RECOGNIZED"
 410 INPUT Z$
 415 PRINT AT 12,1;"ENTER FILE P
OSITION"
 420 INPUT R
 425 LET T$(R)=Z$
 430 PRINT AT 14,1;"PRESS ENTER
TO BEGIN"
 435 INPUT F$
 440 CLS
 445 FOR I=0 TO 7
 450 RAND USR 16520
 455 POKE 25997,I
 460 RAND USR 16615
 465 PRINT AT 10,16;I+1
 470 SLOW
 475 PAUSE 30
 480 FAST
 485 NEXT I
 490 POKE 25996,R
 495 RAND USR 16641
 500 PRINT AT 12,13;"AGAIN? (Y/N
)"
 505 INPUT B$
 510 IF B$="Y" THEN GOTO 400
 515 IF B$="N" THEN GOTO 20
 520 GOTO 505
 599 REM ***RECOGNITION***
 600 RAND USR 16520
 605 RAND USR 16707
 610 CLS
 615 PRINT AT 12,10;T$(PEEK 2599
9+1)
 620 SLOW
 625 PAUSE 60
 630 FAST
 635 IF INKEY$<>"" THEN GOTO 20
 640 GOTO 600
 799 REM ***CLEAR FILES***
 800 FOR I=1 TO 10
 805 LET T$(I)=""
 810 NEXT I
 815 FOR I=26000 TO 26640
 820 POKE I,0
 825 NEXT I
 830 GOTO 20
 999 REM **DISPLAY STRING FILE**
1000 CLS
1005 FOR I=1 TO 10
1010 PRINT AT (5+I),10;I;". ";T$
(I)
1015 NEXT I
1020 PRINT ,,,,"     PRESS ANY KE
Y TO CONTINUE"
1025 SLOW
1030 IF INKEY$="" THEN GOTO 1030
1035 FAST
1040 GOTO 20
1200 STOP
```

Table 10-3
MACHINE CODE ENTRY ROUTINE

```
10 FAST
20 LET K = 16520
30 PRINT AT 10,10;K
40 INPUT I
50 POKE K,I
60 LET K = K+1
70 GOTO 30
```

process for the next word you want to store. Up to eight words can be stored. 3. starts the recognition routine and waits for you to say a word. The best match is displayed. Press any key except "BREAK" to exit this routine. 4. clears all entries in both the voice print and word string files. 5. displays everything stored in the string file. 6. will STOP the program.

If you want to SAVE the voice print files on tape you first must change line 10 of the BASIC program to "DIM C (1412)". Then RUN the program and all voice print files you insert can be SAVEd on tape afterwards. After LOADing a program which has voice print files already stored you must not RUN the program or the files will be lost. In this case you can only RUN the program by entering a "GOTO 20" statement.

ROBOT APPLICATION

As noted in the beginning of the chapter, the above speech recognition routine is not the one that is included as part of the 13K robot software listing (Table 5-10). If you want to include the voice recognition routine as included in the 3K listing, you must use the software program that is available from G. Russell Electronics, RD 1, Box 539, Center Hall, PA 16828. It is available on cassette, with documentation, for $9.95 postpaid. The assembled amplifier with the cassette and documentation has been available for $34.95. If you use the software on this cassette, you can use one of two methods for building the program. You can either load the voice recognition program first and then enter the rest of the robot listing via the keyboard. Or, if you have already built the rest of the robot program listing, you will have to use special software to merge the two programs. More can be found on merging programs in Chapter 12.

In any case, the machine code used to run the robot (the RX81 machine code) will be displaced by the voice recognition machine code and must be moved. This is easily accomplished by starting the entry of the RX81 machine code at a different address location. Instead of starting the load of the decimal code at address 16514 (Table 3-3) it starts at address location 17858. This means the last location for the machine code will be at address 17881 (vice 16537). This area in the G. Russell Electronics casssette software first REM statement has been reserved for just such applications.

Don't forget that the documentation in Table 5-11 (Chapter 5) includes references to all voice recognition routines, for easy location if you want more details on how it works in the

Table 10-4
SPEECH RECOGNITION MACHINE CODE AND ADDRESSES

16520	33	16582	205	16645	168	16708	144
16521	0	16583	207	16646	97	16709	101
16522	88	16584	64	16647	205	16710	1
16523	6	16585	35	16648	38	16711	255
16524	255	16586	12	16649	65	16712	0
16525	54	16587	29	16650	123	16713	197
16526	0	16588	32	16651	2	16714	17
16527	5	16589	248	16652	3	16715	0
16528	35	16590	201	16653	62	16716	88
16529	194	16591	126	16654	104	16717	14
16530	141	16592	167	16655	185	16718	0
16531	64	16593	200	16656	32	16719	6
16532	6	16594	254	16657	245	16720	64
16533	255	16595	43	16658	58	16721	26
16534	33	16596	56	16659	140	16722	150
16535	0	16597	2	16660	101	16723	48
16536	88	16598	62	16661	1	16724	2
16537	14	16599	43	16662	64	16725	47
16538	254	16600	71	16663	0	16726	60
16539	237	16601	197	16664	33	16727	129
16540	120	16602	213	16665	80	16728	48
16541	242	16603	229	16666	101	16729	2
16542	155	16604	205	16667	9	16730	62
16543	64	16605	178	16668	61	16731	255
16544	44	16606	11	16669	32	16732	79
16545	237	16607	225	16670	252	16733	35
16546	120	16608	209	16671	17	16734	19
16547	250	16609	193	16672	40	16735	5
16548	160	16610	5	16673	100	16736	32
16549	64	16611	194	16674	235	16737	239
16550	52	16612	217	16675	237	16738	121
16551	5	16613	64	16676	176	16739	193
16552	200	16614	201	16677	201	16740	185
16553	237	16615	6	16678	197	16741	50
16554	120	16616	64	16679	62	16742	216
16555	250	16617	33	16680	8	16743	89
16556	169	16618	160	16681	1	16744	48
16557	64	16619	97	16682	0	16745	5
16558	46	16620	58	16683	0	16746	79
16559	0	16621	141	16684	17	16747	120
16560	17	16622	101	16685	0	16748	50
16561	220	16623	133	16686	0	16749	143
16562	255	16624	111	16687	78	16750	101
16563	27	16625	17	16688	235	16751	4
16564	122	16626	0	16689	9	16752	62
16565	179	16627	88	16690	235	16753	10
16566	200	16628	213	16691	61	16754	184
16567	237	16629	17	16692	35	16755	32
16568	120	16630	8	16693	32	16756	212
16569	242	16631	0	16694	248	16757	201
16570	179	16632	25	16695	6	16758	33
16571	64	16633	209	16696	3	16759	136

(each column is continued on next page)

Table 10-4 (Continued)

16572	195	16634	26	16697	151	16760	64
16573	160	16635	119	16698	203	16761	151
16574	64	16636	19	16699	26	16762	6
16575	33	16637	5	16700	203	16763	238
16576	0	16638	32	16701	27	16764	134
16577	88	16639	244	16702	5	16765	35
16578	14	16640	201	16703	32	16766	5
16579	0	16641	1	16704	248	16767	32
16580	30	16642	40	16705	193	16768	251
16581	64	16643	100	16706	201	16769	79
		16644	33	16707	33	16770	201

Table 10-5
MACHINE CODE ERROR FINDING ROUTINE

```
10 LET K = 16520
20 CLS
30 FOR I = 0 TO 19
40 PRINT K+I;" ";PEEK (I+K),K+I+20;" ";PEEK(I+K+20)
50 NEXT I
60 INPUT J
70 LET K = K+40
80 GOTO 20
```

13K robot software. For example, routine R3 as documented in Table 5-11, waits for voice input after stopping the robot. The commands the robot is waiting for are either a "continue" or an "arms" voice input. If "arms" is recognized, then the robot goes into a demonstration of its arm, hand and upper body movements. If a "continue" voice command is given and successfully recognized, the robot continues its forward motion, exploring its envoirnment. Built into the routine is the provision to also continue its forward motion after two successive one minute waiting periods is no voice command is received. A close study of the software also reveals that the robot uses its voice to ask for the amp (amplifier) to be turned on before waiting for a voice input. It also responds to voice input, whether it is a properly recognized input or not, with its own voice response. There is also a provision to confirm or deny the voice input that the robot has correctly or incorrectly recognized with a "yes", "no" or "stop" voice input. If the voice input confirmation is not a "yes", then the recognition routine starts over again at the beginning in an attempt to get the input command correct.

THIS PAGE INTENTIONALLY BLANK

Chapter 11
ULTRASONIC RANGING SENSOR

The first major expansion of H.E.N.R.Y.'s input system consisting of bumper switches was the addition of another type of sensor, an ultrasonic range measuring system. The addition of such a sensor raises the robot's level of artificial intelligence by upgrading its navigation techniques. For example, if you observed someone navigating down a hallway by alternately bumping into walls and then changing direction (that's how the robot described in previous chapters does it) you would probably deduce that the person was either blind (lacked one of the senses) or was maybe lacking in intelligence. With the addition of a range measuring device like the ultrasonic rangefinder, the robot's navigation capabilities can be advanced to a goal seeking ability. The control software can be modified so that the robot will seek open areas to navigate through. The robot can then move down a hallway seeking a doorway opening instead of stumbling into it.

This chapter will contain enough information to allow you to interface the ultrasonic rangefinder with the ZX81/TS1000 computers or the TS2068 computer to either add the sensor capability to the robot, build a burglar alarm or any other range measurement application you come up with.

CIRCUIT REQUIREMENTS

Although there are a number ultrasonic rangefinders available, I selected the OEM kit from Polaroid, available for $99. Figure 11-1 shows you what you get from Polaroid for your $99. It consists of two complete sets of transducers, cables and ranging circuits. The documentation that comes with the kit will give you a lot of good ideas. There are six color coded wires coming from the ranging circuit board which you will have to identify and understand in order to construct an interface. The red wire provides power to the circuit in order to produce an ultrasonic pulse and is labeled as VSW. The blue wire is supposed to indicate that the circuit has transmitted an ultrasonic pulse and is labeled as XLG. I could not detect the proper signal on XLG and therefore did not use it in the interface I constructed. Instead I used the application of power (VSW) to begin timing at the beginning of each ultrasonic pulse. According to the documentation this method is not accurately repeatable but I found it accurate within less than 10% and therefore suitable for the these applications. The green wire provides the detected echo signal and is labeled as FLG. The remaining wires are connected to a power source to provide a

Figure 11-1. Polaroid ultrasonic ranging OEM kit.

ground (Brown) and +5 volts @ 1 amp (Orange & Yellow).

The ultrasonic circuit board will be interfaced to the computer using an RX81 I/O board and a small circuit you must construct in order to provide the +5 volt power (an LM323K voltage regulator); switch the power on/off (a 2N2907 transistor and 7404 logic inverter); and, an echo detection interface (a 470 ohm resistor). These circuits are illustrated in Figure 11-2.

The following is an explanation of how the circuit functions (Figure 11-2). First, a dedicated +5 volt regulator is recommended because of the high (1 amp), short duration power required to produce an ultrasonic pulse. When the computer sends an OUT signal through the RX81 (wired for data line, D0) to the transistor, power is applied to the ultrasonic circuit over line VSW causing an ultrasonic pulse to be sent to the transducer. Note: Never activate the circuit without a transducer connected, or damage may result. When the ultrasonic sound is reflected off an object and returned to the transducer it is detected by the circuit and a signal is produced on the FLG line. This signal is in turn read by the RX81 as an IN on data line D2 (or data line D1, depending on the particular version(s) described later in this chapter). The circuit is as simple as it sounds. The software is now all that is required to make it work.

SOFTWARE DETAILS

Because of the speed at which the ultrasonic pulses travel (the speed of sound) the software is required to be written in machine code. Four versions of the software, with an additional variation of one version provided for use in the robot, are provided. In case you are wondering what happened to version 1, that was the first version I wrote intending to use the XLG transmission signal. As I already explained, that I couldn't use that signal so version 1 was scrapped.

Because the method of entering machine code has been covered in detail in previous chapters, I will not repeat those procedures again here.

Version 2 software (Table 11-1) is designed for use on the ZX81/TS1000 with the RX81 I/O board wired as IN 1 and OUT 7. The machine code is fully documented with explanations. The technique previously explained in other applications whereby the IN is read and confirmed to eliminate false input readings is used here and in all versions. The BASIC language routine for version 2 prints (displays on the screen) the actual number of counts that the machine code routine has registered between ultrasonic pulse transmission and receipt of the return echo.

Version 3 software (Table 11-2) is similar to version 2 except that the machine code is designed to run with a circuit where the RX81 I/O board is wired as IN 3 and OUT 5. Also note that, unlike the circuit in Figure 11-2, the input is wired to data line D1 instead of D2. All remaining software also is designed for use with this wiring. The BASIC routine for version 3 converts the machine code count into a range in feet and displays it on the screen.

Table 11-3 contains a variation of version 3 which can be inserted directly into the robot software. The machine code functions are the same as those in Table 11-2, version 3, but

Figure 11-2
Interface and power circuits between
RX81 and ultrasonic circuit boards

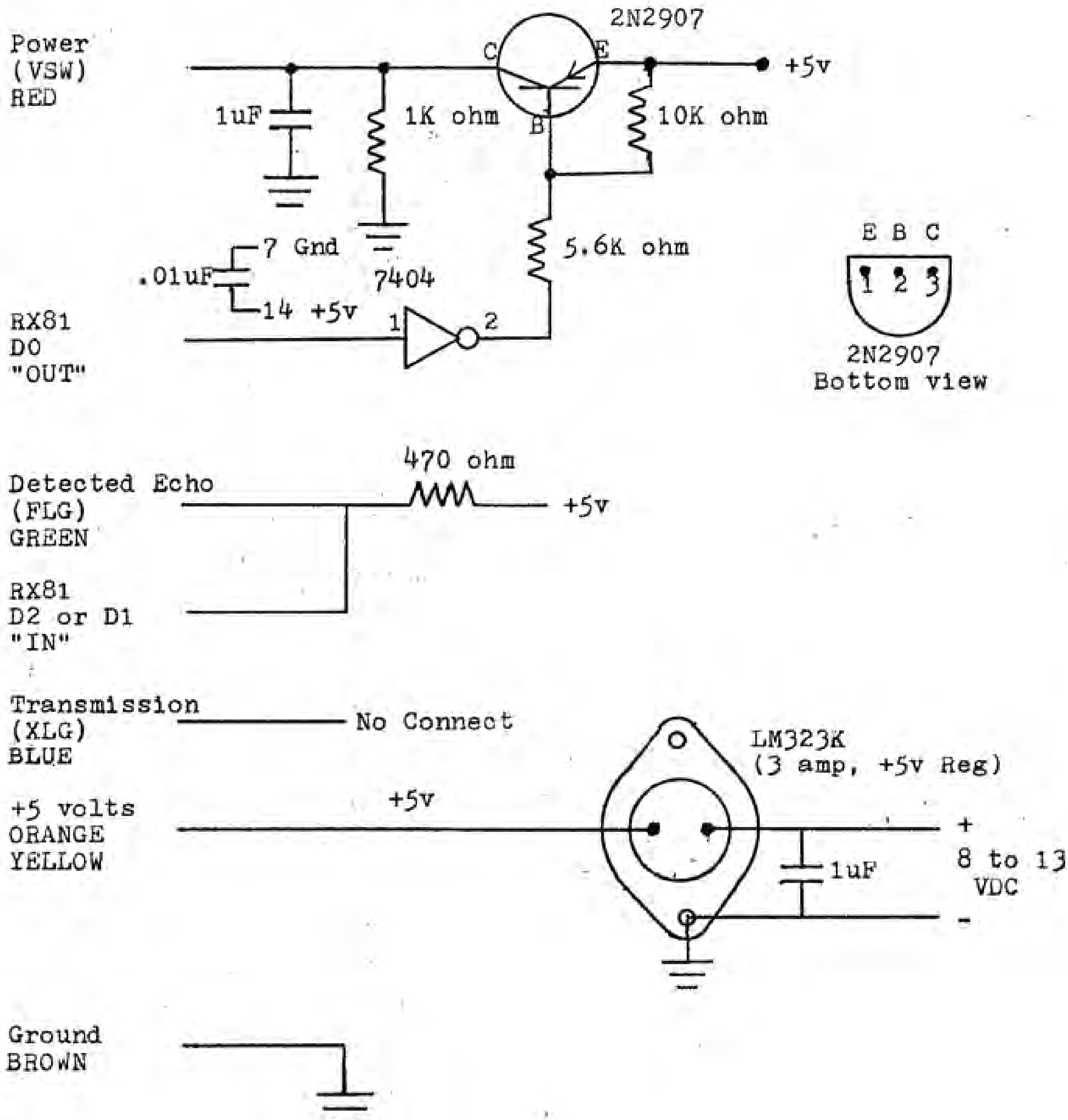


Table 11-1
Version 2 Software

RX81 MACHINE CODE DOCUMENTATION FOR ULTRASONIC RANGING

ZX81/TS1000 address (Sequence)	Code	Hex	Z80 Assembler	(Function)
(power on)				
16514	62		ld a,N	
16515	1			(output select, D0=1)
16516	211		OUT N,a	(select I/O board wired as
16517	7			OUT 7)
(FLG pulse in)				
16518	219		in a,N	(select I/O board wired as
16519	1			IN 1)
16520	47		cpl	
16521	230		AND N	(mask for FLG input at D2)
16522	4			(D2=4)
16523	71		ld b,a	(load register b for later comparison with register a)
(count to measure time until FLG is received)				
16524	42		ld HL,(NN)	(reserve address to store count)
16525	172	AC		(pointer for address 16556
16526	64	40		16556 = 40 AC hex)
16527	17		ld DE,NN	(load the number 01 to start count)
16528	1			
15629	0			
16530	25		ADD HL,DE	(add count of 1 each time routine is entered)
16531	34		ld (NN),HL	(store new one up count at
16532	172	AC		address 16556 = 40 AC)
16533	64	40		
(compare FLG input pulse)				
16534	14		ld c,N	(pulse return default count
16535	12			12 x 256 = 3072 pulses or @ 60 feet)
16536	47		cpl	
16537	230		AND N	(mask 12 count)
16538	12			
16539	188		CP h	(compare with pulse count if
16540	40		JR Z,N	same jump forward to 16550
16541	10			if not, continue)
16542	219		in a,N	(select I/O board wired as
16543	1			IN 1)
16544	47		cpl	
16545	230		AND N	(mask for FLG input at D2)
16546	4			(D2=4)

Table 11-1 (Continued)

16547	184	CP b	(compare with first pulse in
16548	40	JR Z,N	if the same pulse continue,
16549	230	-26	if not, jump back to 16524)

(FLG pulse at D2 has now been confirmed)

(power off)

16550	62	ld a,N	
16551	0		(output select 0 = off)
16552	211	out N,a	(select output board wired
16553	7		as OUT 7)
16554	201	ret	(return to BASIC routine)

(area for storage of count)

16555	0		
16556	0		(hex address = 40 AC)
16557	0		(hex address = 40 AD)
16558	0		
16559	0		

BASIC PROGRAM FOR ULTRASONIC RANGING (version 2)

```
 1 REM
10 FAST
20 POKE 16556,0
30 POKE 16557,0
40 RAND USR 16514
50 PRINT PEEK 16557*256+PEEK 16556
60 PAUSE 200
70 GOTO 20
```

(NOTE: This version is written for an RX81 I/O board wired as IN 1 and OUT 7. Also note that Input is at D2 and Output is at D0)

Table 11-2
Version 3 Software

RX81 MACHINE CODE DOCUMENTATION FOR ULTRASONIC RANGING

ZX81/TS1000 address (Sequence)	Code	Hex	Z80 Assembler	(Function)
(power on)				
16514	62		ld a,N	
16515	1			(output select, D0=1)
16516	211		OUT N,a	(select I/O board wired as
16517	5			OUT 5)
(FLG pulse in)				
16518	219		in a,N	(select I/O board wired as
16519	3			IN 3)
16520	47		cpl	
16521	230		AND N	(mask for FLG input at D2)
16522	2			(D1=2)
16523	71		ld b,a	(load register b for later comparison with register a)
(count to measure time until FLG is received)				
16524	42		ld HL,(NN)	(reserve address to store count)
16525	172	AC		(pointer for address 16556
16526	64	40		16556 = 40 AC hex)
16527	17		ld DE,NN	(load the number 01 to start count)
16528	1			
15629	0			
16530	25		ADD HL,DE	(add count of 1 each time routine is entered)
16531	34		ld (NN),HL	(store new one up count at
16532	172	AC		address 16556 = 40 AC)
16533	64	40		
(compare FLG input pulse)				
16534	14		ld c,N	(pulse return default count
16535	12			12 x 256 = 3072 pulses or @ 60 feet)
16536	47		cpl	
16537	230		AND N	(mask 12 count)
16538	12			
16539	188		CP h	(compare with pulse count if
16540	40		JR Z,N	same jump forward to 16550
16541	10			if not, continue)
16542	219		in a,N	(select I/O board wired as
16543	3			IN 3)
16544	47		cpl	
16545	230		AND N	(mask for FLG input at D2)
16546	2			(D1=2)

16547	184	CP b	(compare with first pulse in
16548	40	JR Z,N	if the same pulse continue,
16549	230	-26	if not, jump back to 16524)

(FLG pulse at D1 has now been confirmed)

(power off)

16550	62	ld a,N	
16551	0		(output select 0 = off)
16552	211	out N,a	(select output board wired
16553	5		as OUT 5)
16554	201	ret	(return to BASIC routine)

(area for storage of count)

16555	0		
16556	0		(hex address = 40 AC)
16557	0		(hex address = 40 AD)
16558	0		
16559	0		

BASIC PROGRAM FOR ULTRASONIC RANGING (version 3)

```
  1 REM
 10 FAST
 20 POKE 16556,0
 30 POKE 16557,0
 40 RAND USR 16514
 50 LET R=((PEEK 16557*256+PEEK
16548)-179)/46.5
 60 PRINT INT (R+0.05);" FEET"
 70 PAUSE 10
 80 CLS
 90 GOTO 20
```

(NOTE: This version is written for an RX81 I/O board wired as IN 3 and OUT 5. Also note that Input is at D1, previous version had Input at D2, and Output is at D0)

Table 11-3
Version 3 Software for Robot

RX81 MACHINE CODE DOCUMENTATION FOR ULTRASONIC RANGING WITH ADDRESSES MODIFIED FOR INSERTION IN ROBOT CONTROL PROGRAM

ZX81/TS1000 address (Sequence)	Code	Hex	Z80 Assembler	(Function)
(power on)				
17934	62		ld a,N	
17935	1			(output select, D0=1)
17936	211		OUT N,a	(select I/O board wired as
17937	5			OUT 5)
(FLG pulse in)				
17938	219		in a,N	(select I/O board wired as
17939	3			IN 3)
17940	47		cpl	
17941	230		AND N	(mask for FLG input at D1)
17942	2			(D1=2)
17943	71		ld b,a	(load register b for later comparison with register a)
(count to measure time until FLG is received)				
17944	42		ld HL,(NN)	(reserve address to store count)
17945	56	38		(pointer for address 17976
17946	70	46		17976 = 46 38 hex)
17947	17		ld DE,NN	(load the number 01 to start count)
17948	1			
17949	0			
17950	25		ADD HL,DE	(add count of 1 each time routine is entered)
17951	34		ld (NN),HL	(store new one up count at
17952	56	38		address 17976 = 46 38)
17953	70	46		
(compare FLG input pulse)				
17954	14	ld c,N		(pulse return default count
17955	12			12 x 256 = 3072 pulses or @ 60 feet)
17956	47		cpl	
17957	230		AND N	(mask 12 count)
17958	12			
17959	188		CP h	(compare with pulse count if
17960	40		JR Z,N	same jump forward to 17970
17961	10			if not, continue)
17962	219		in a,N	(select I/O board wired as
17963	3			IN 3)
17964	47		cpl	
17965	230		AND N	(mask for FLG input at D1)

Table 11-3 (Continued)

17966	2		(D1=2)
17967	184	CP b	(compare with first pulse in
17968	40	JR Z,N	if the same pulse continue,
17969	230	-26	if not, jump back to 17944)

(FLG pulse at D1 has now been confirmed)

(power off)

17970	62	ld a,N	
17971	0		(output select 0 = off)
17972	211	out N,a	(select output board wired
17973	5		as OUT 5)
17974	201	ret	(return to BASIC routine)

(area for storage of count)

17975	0	
17976	0	(hex address = 46 38)
17977	0	(hex address = 46 39)
17978	0	
17979	0	

BASIC PROGRAM CHANGES/INSERTIONS ULTRASONIC RANGING FOR ROBOT CONTROL PROGRAM (version 3)

		line #
U1	Ultrasonic ranging routine	4000 - 4030
U2	Ultrasonic pulse routine	4500 - 4550
U3	Ultrasonic range right and left subroutine	4100 - 4380
U4	Ultrasonic intruder detection routine	4700 - 4790
V22	Alarm on announcement	1500 - 1670

NOTE: BASIC listings for above routines are listed below.

ADDITIONAL COMMANDS REQUIRED

```
 127  IF Z < .5 THEN GOTO 132
2110  IF Z < .67 THEN GOTO 9185
4705  GOSUB 1500
9062  LET S = 17976
9063  LET T = 17977
9068  LET U = 17934
9163  POKE 16434, PEEK T
9164  POKE 16435, PEEK S
9165  LET Z = RND
9265  IF IN = 64 THEN GOTO 4700
9292  GOTO 4000
9295  IF TN = 7 THEN POKE N,42
9300  IF TN = 40 THEN GOTO 2100
```

Table 11-3 (Continued)

VARIABLES

U Ultrasonic ranger - One pulse measurement

P,Q,X Range measurements

S,T Range stored in hex

U1 ULTRASONIC RANGING ROUTINE

```
4000  REM U1
4010  GOSUB 4500
4020  IF Q < 2 THEN GOTO 4100
4030  GOTO 9293
```

U2 ULTRASONIC PULSE ROUTINE

```
4500  REM U2
4510  POKE S,0
4520  POKE T,0
4530  RAND USR U
4540  LET Q = ((PEEK T * 256 + PEEK S) - 179)/46.5
4550  RETURN
```

U3 ULTRASONIC RANGE RIGHT AND LEFT SUBROUTINE

```
4100  REM U3
4110  SLOW
4120  POKE A,32                 (output selected, body
4130  GOSUB G                    rotates right 22 degrees)
4140  PAUSE 60
4150  POKE A,0                  (stop body rotation)
4160  GOSUB G
4170  FAST
4180  GOSUG 4500                (pulse)
4190  LET P = Q                 (store range)
4200  SLOW
4210  POKE A,16                 (body left 45 degrees)
4220  GOSUB G
4230  PAUSE 120
4240  POKE A,0                  (stop body rotation)
4250  GOSUB G
4260  FAST
4270  GOSUB 4500                (pulse)
4280  SLOW
4290  POKE A,32                 (body rotates returning to
4300  GOSUB G                    straight ahead position)
4310  PAUSE 60
4320  POKE A,0
4340  GOSUB G
4350  FAST
```

Table 11-3 (Continued)

```
4360  IF Q < 2 AND P < 2 THEN GOTO 100
4370  IF Q < P THEN GOTO 300
4380  GOTO 200                    (in these last three BASIC
                                   lines the decision is made
                                   to move toward the open
                                   space: right, left or fwd)
```

U4 ULTRASONIC INTRUDER DETECTION ROUTINE

```
4700  REM U4
4710  LET X = 6
4720  FOR W = 1 TO 10                  (measures range)
4730  GOSUB 4500
4740  PAUSE 20
4750  IF W = 10 THEN GOTO 4780
4760  LET X = Q
4770  NEXT W
4780  IF (Q + 1.4) < X THEN GOTO 700      (activates voice
4790  GOTO 4730                            warning if range
                                           decreases)*

                                          * Voice warning
                                            routine is not
                                            provided here.
```

V22 ALARM ON ANNOUNCEMENT

```
1500  REM V22
1510  POKE M,1
1520  POKE N,8                   "attention"
1530  PAUSE 30
1540  POKE M,0
1550  POKE N,120                 "please"
1560  PAUSE 50
1570  POKE M,1
1580  POKE N,63                  "intruder"
1590  PAUSE 30
1600  POKE N,3                   "alarm"
1610  PAUSE 30
1620  POKE M,0
1630  POKE N,96                  "is"
1640  PAUSE 20
1650  POKE N,115                 "on"
1660  PAUSE 20
1670  RETURN
```

the addresses have been adjusted to prevent interference with the motor control, bumper switch, and voice recognition machine code routines already in the robot software. The BASIC routines include avoidance software in subroutines U1, U2 and U3 and also a buglar alarm routine in subroutine U4 with voice warning in subroutine V22. For a full understanding of these routines the documentation can be integrated with the robot software and documentation found in Tables 5-10 and 5-11.

Table 11-4 contains the software required to operate the circuit with the TS2068 computer. The BASIC software is provide for two applications. One version is a range measuring routine and the other version is an intrusion or range changing routine.

As you can see from the different applications suggested in the software presented here, there are endless numbers of uses for the ultrasonic rangefinder circuit. The robot can perform as a mobile burglar alarm, distances can be measured, etc. Even though the measurements made by this system are crude, the addition of these "eyes" to your computer opens up many possibilities. To give you some idea of the specifications and/or limitations of this system, Figure 11-3 is provided. The angular coverage of the transducer is approximately 25 degrees. The figure should be helpful in deciding transducer placement on your robot or other application.

Table 11-4
Versions 4 & 5 Software

RX81 MACHINE CODE DOCUMENTATION FOR ULTRASONIC RANGING

TS2068 address (Sequence)	Code	Hex	Z80 Assembler	(Function)
(power on)				
65254	62		ld a,N	
65255	1			(output select, D0=1)
65256	211		OUT N,a	(select I/O board wired as
65257	5			OUT 5)
(FLG pulse in)				
65258	219		in a,N	(select I/O board wired as
65259	3			IN 3)
65260	47		cpl	
65261	230		AND N	(mask for FLG input at D1)
65262	2			(D1=2)
65263	71		ld b,a	(load register b for later comparison with register a)
(count to measure time until FLG is received)				
65264	42		ld HL,(NN)	(reserve address to store count)
65265	16	10		(pointer for address 65296
65266	255	FF		65296 = FF 10 hex)
65267	17		ld DE,NN	(load the number 01 to start count)
65268	1			
65269	0			
65270	25		ADD HL,DE	(add count of 1 each time routine is entered)
65271	34		ld (NN),HL	(store new one up count at
65272	16	10		address 65296 = FF 10)
65273	255	FF		
(compare FLG input pulse)				
65274	14		ld c,N	(pulse return default count
65275	12			12 x 256 = 3072 pulses or @ 60 feet)
65276	47		cpl	
65277	230		AND N	(mask 12 count)
65278	12			
65279	188		CP h	(compare with pulse count if
65280	40		JR Z,N	same jump forward to 65290
65281	10			if not, continue)
65282	219		in a,N	(select I/O board wired as
65283	3			IN 3)
65284	47		cpl	
65285	230		AND N	(mask for FLG input at D1)
65286	2			(D1=2)

65287	184	CP b	(compare with first pulse in
65288	40	JR Z,N	if the same pulse continue,
65289	230	-26	if not, jump back to 65264)
(FLG pulse at D1 has now been confirmed)			
(power off)			
65290	62	ld a,N	
65291	0		(output select 0 = off)
65292	211	out N,a	(select output board wired
65293	5		as OUT 5)
65294	201	ret	(return to BASIC routine)
(area for storage of count)			
65295	0		
65296	0		(hex address = FF 10)
65297	0		(hex address = FF 11)
65298	0		
65299	0		

BASIC PROGRAM FOR ULTRASONIC RANGING (VERSION 4)

```
 5 CLEAR 65253: FOR n=65254 TO 65299
10 READ d: POKE n,d: NEXT n
15 DATA 62,1,211,5,219,3,47,230,2,71,42,16,255,17,1,0,25,
34,16,255,14,12,47,230,12,188,40,10,219,3,47,230,2,184,40,
230,62,0,211,5,201,0,0,0,0,0
20 POKE 65296,0
30 POKE 65297,0
40 RANDOMIZE USR 65254
50 LET r=((PEEK 65297*256+PEEK
65296)-192.5)/3.87
60 PRINT INT r;" Inches"
70 PAUSE 10
80 CLS
90 GOTO 20
```

BASIC PROGRAM FOR ULTRASONIC RANGING INTRUSION ALARM (VERSION 5)

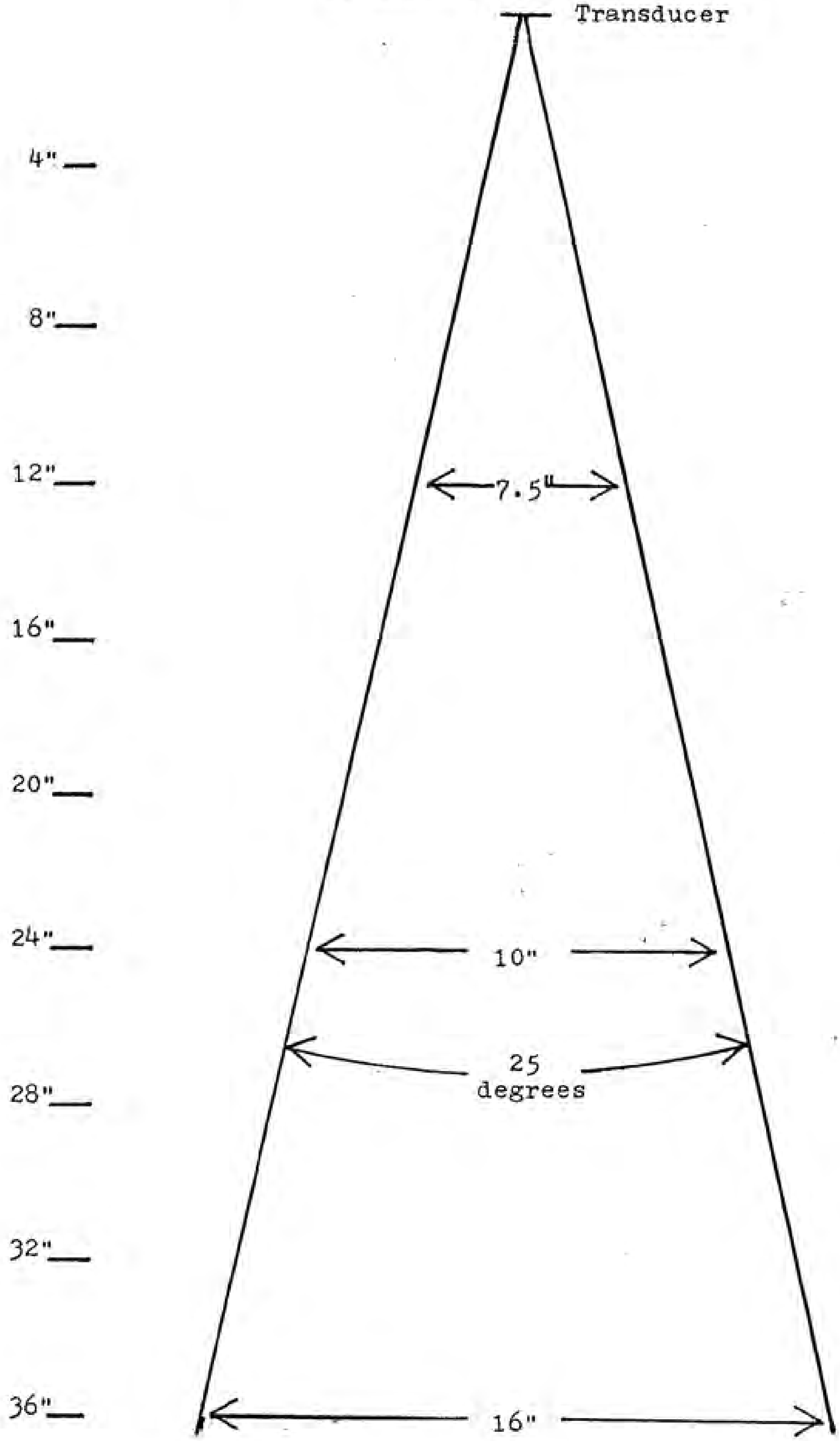
```
 5 CLEAR 65253: FOR n=65254 TO 65299
10 READ d: POKE n,d: NEXT n
15 DATA 62,1,211,5,219,3,47,230,2,71,42,16,255,17,1,0,25,
34,16,255,14,12,47,230,12,188,40,10,219,3,47,230,2,184,40,
230,62,0,211,5,201,0,0,0,0,0
17 LET ra=6
18 FOR w=1 TO 10
20 POKE 65296,0
30 POKE 65297,0
40 RANDOMIZE USR 65254
```

Table 11-4 (Continued)

```
 50 LET r=((PEEK 65297*256+PEEK
65296)-192.5)/46.5
 70 PAUSE 20
 80 IF w=10 THEN GO TO 100
 85 LET ra=r
 90 NEXT w
100 IF (r+1.4) <ra THEN GO TO 200
110 GO TO 20
200 PRINT "Intruder detected"
210 BEEP 1,0: BEEP 1,12
220 GO TO 210
```

(NOTE: These versions are written for an RX81 I/O board wired as IN 3 and OUT 5. Also note that Input is at D1 and Output is at D0)

Figure 11-3
Transducer field of view



THIS PAGE INTENTIONALLY BLANK

Chapter 12
HARDWARE AND SOFTWARE TIPS

This chapter is a collection of miscellaneous hardware and software tips which may be of use to you as you tinker away with your computer control projects. For hardware they range from tips on how to actually modify your equipment as would be required to drive a video monitor to aligning the head on your tape recorder to prevent LOAD and SAVE problems.

TAPE LOAD/SAVE PROBLEMS

One of the drawbacks of the Sinclair ZX81, TS1000 and TS1500 computers is the potential for LOAD and SAVE problems related to your tape recorder. But as I always say "What do you expect from a $40 computer?" The problems are caused by the fact that the computer requires an input signal with a very narrow envelope, so there is little tolerance for error. The odds are very great the sooner or later you will probably encounter LOADing problems with your computer. And after talking to many people with problems, I think you may run into problems sooner than later. I went for nearly a year of operation with my ZX81 before I had serious problems and I learned a great deal from the experience. But, don't be discouraged, the vast majority of problems can be solved very easily.

First of all, some tape recorders do not work at all with your computer but work fine with voice or music. It may LOAD a pre-recorded tape but will not SAVE a program due to insufficient recording level. This is caused by the automatic record level circuitry in some recorders. I have no solution for this problem other than not using such a recorder.

The most talked about problem is the playback volume for loading. Much has been written and many devices sold (LED volume level indicators) to solve this problem.

A far more critical problem than volume level is the alignment of the record/playback head on the tape recorder. Although many commercial program tape suppliers guaranty that their tapes will load, if your tape head (or the tape supplier's) is out of alignment significantly, the tape will not load. A bad alignment causes the tape to produce a signal or stream of data with a different range of frequencies than the computer originally produced. In other words, the stream of data sent to the recorder during the SAVE operation contains signals with a certain range of frequencies. These signals are recorded on the tape. Then if another recorder is used to LOAD (playback) this tape and either of the recorders had a head that was out of alignment, the signals would be played back at a range of frequencies incompatable with the computer, and would not LOAD successfully.

There is another interesting aspect to this alignment problem. If your recorder head is out of alignment, tapes SAVEd on it can probably be LOADed successfully using the same recorder. This is true only if your head stays at the same "out of alignment" position. However, your tapes SAVEd on your recorder will not LOAD using another recorder.

A serious alignment problem can slowly develop over a long period of tape recorder use. With this problem the tape head slowly but continually moves further out of alignment, over time, with use.

To explore this problem, let me describe what physically happens to the tape head during alignment. By the way, even the most expensive commercial disc and tape drives experience these same alignment problems. Anyway, your small tape recorder has only one head which serves for both recording and playback. To take a closer look at the head, open the cassette access lid and push the "play" button. The "record" button will not push in unless a cassette has been inserted, in which case you could not see the head move into place. Notice that the head (a shiny rectangular device with a rounded surface facing the tape) is fastened in place with two screws. One screw is on the tape feed end and one is toward the takeup end. One of the screws (probably the one on the feed end) Will have a spot of paint covering part of it. If you can see under the head mounting on this end, you should see a small compression spring under the screw or head mounting. This is the head adjustment or alignment screw. Turning it up or down (clockwise = down and counterclockwise = up) changes the angle of the head in relation to the tape and thus changes the alignment. Unless your system shows the symptoms of a head out of alignment, I do not recommend you proceed with this alignment procedure. After all, why throw your head out of alignment if it is OK?

Although you can have your recorder aligned by an expert (for a price) using sophisticated signal producing and measuring equipment, you can also do it yourself using a very sophisticated piece of nature's equipment, your ear. Set up one of your program tapes and adjust the sound (remove the ear plug so you can hear the sound over the speaker) so that it produces the highest pitched (frequency) sound as follows. When the head is in the playback position, the adjusting screw will be visible through a small hole in the top of the case. You will need a small screw driver that will fit through the hole. Turn the screw clockwise first, as the screw will have turned counterclockwise if it has loosened and come out of adjustment. As little as an eighth of a turn can be enough to cause or correct problems. Once you have passed through the highest pitched sound, back of the setting and pass through it several times to zero in on the best setting. The best setting is the highest pitch while remaining loudest. One caution. If you just recently started experiencing the problem, do not use a tape for this procedure that you recently recorded (SAVEd). Instead, use a tape you recorded (SAVEd) before the problem appeared. In other words, use an older tape that always LOADed properly in the past, but won't LOAD now. Otherwise you will not get the alignment back to its original position. Once you have made the proper adjustment, note the position of the screw head because unless you can secure the screw head with some model paint or finger nail polish, it will eventually move back out of alignment again.

Before you try securing the screw head, you may want to recover some of the programs you either SAVEd while the head was out of alignment, or a tape you bought or got from someone which you had never been able to LOAD before. Here is how you do it. Play the tape and adjust the head for the highest sounding pitch as described above, for the tape you want to LOAD. Then LOAD the program. But, before you SAVE the program, return the head adjustment screw to the new proper adjustment setting you had previously determined. Once you have recorded all the old tapes you desire, lock the adjustment screw with a dab of nail polish or model paint. Be careful not to get any on the tape head.

Finally, another problem that can give you similar problems, but is not as common, is incorrect tape drive speed. This can be tested by making a test tape using an old unneeded cassette, as long as the tape transport will work. The tape speed is supposed to be 1 7/8 (1.875) inches per second. Measure out 56.25 (30 x 1.875) inches of tape on the cassette by pulling the tape out of the front access opening. Then starting with the tape in the rewound position, place a piece of tape splicing or other thin tape over the recording surface of the tape at the 56.25 inch mark. Now you can start your recorder and a stop watch at the same time. With the volume set at maximum, listen for the dead spot on the tape. It should show up 30 seconds into the tape. If the speed is off and the batteries are fresh, you can try to find the motor speed adjustment which is a variable resistor inside the recorder. I would only recommend this procedure for someone experienced in electro-mechanical systems. Any personal robot builder should be qualified! On my recorder, the adjustment can be made through an access from the battery compartment, but other recorders may be different.

May all your LOADs be successful!

ZX81/TS1000/TS1500 DIRECT VIDEO OUTPUT

When I decided that it was time to replace my jittery 12 year old television I had been using for my ZX81 display, I figured it was time to convert to a first class video monitor. I had Mike Lord's suggested monitor circuit modification in his "The Explorers Guide to the ZX81", so I ordered a Zenith 12 inch video monitor (model ZVM-121). When the monitor arrived, I made the modification to my ZX81 (Figure 12-1) and powered up. I got a good sharp picture but shortly found it was not perfect under all conditions. When I displayed more than a few reverse video characters in a single line, that line would tear and in some cases severly distort the picture. After much experimenting with this circuit and others, I found that only a slight modification to Mr. Lord's circuit circuit fixed the problem completely. The circuit is shown in Figure 12-2. The improvement to the circuit is the removal of one of the two diodes from between the transistor emitter and the resistors, leaving only one diode there and the addition of two other

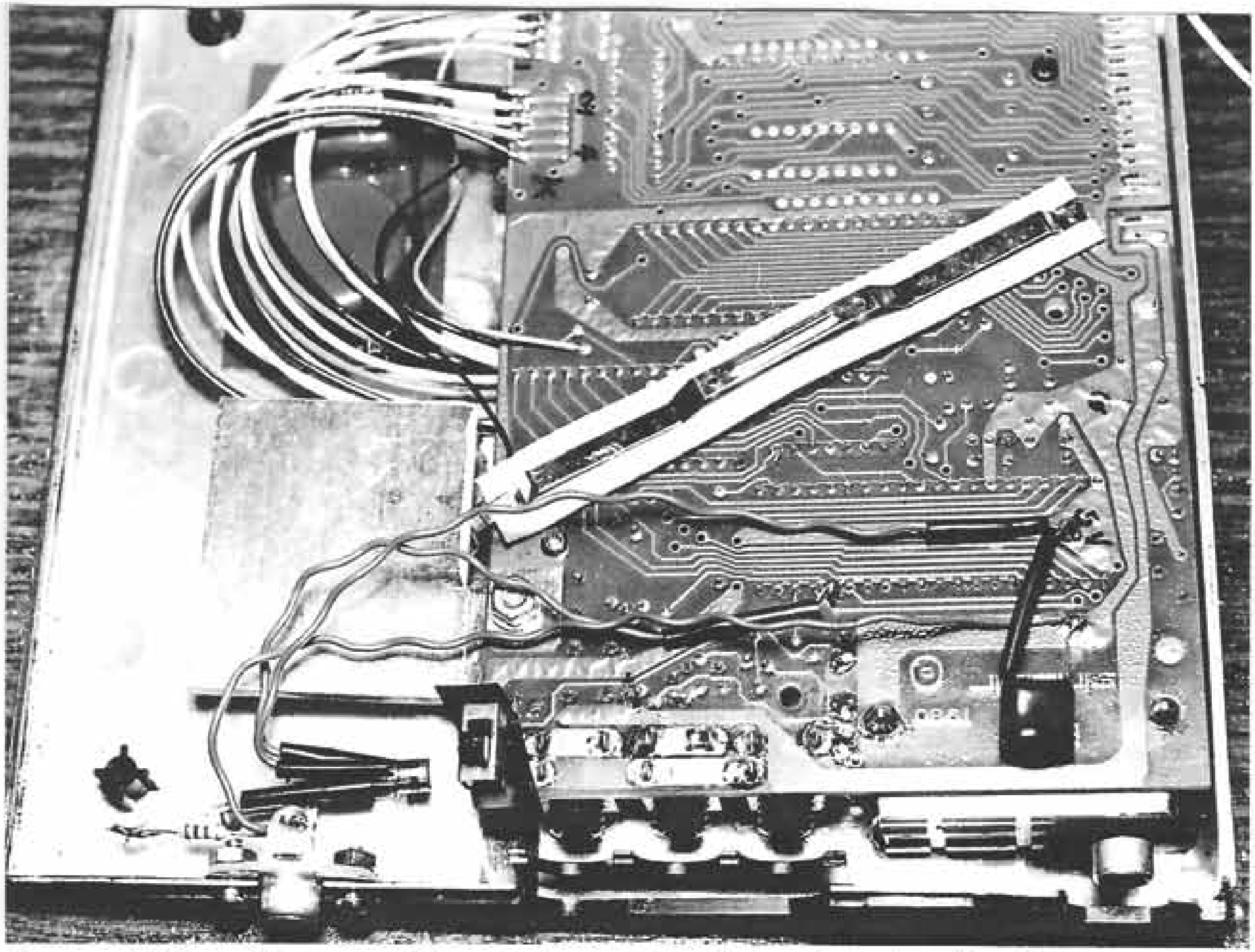


Figure 12-1. Video monitor driver circuit hookup inside the computer.

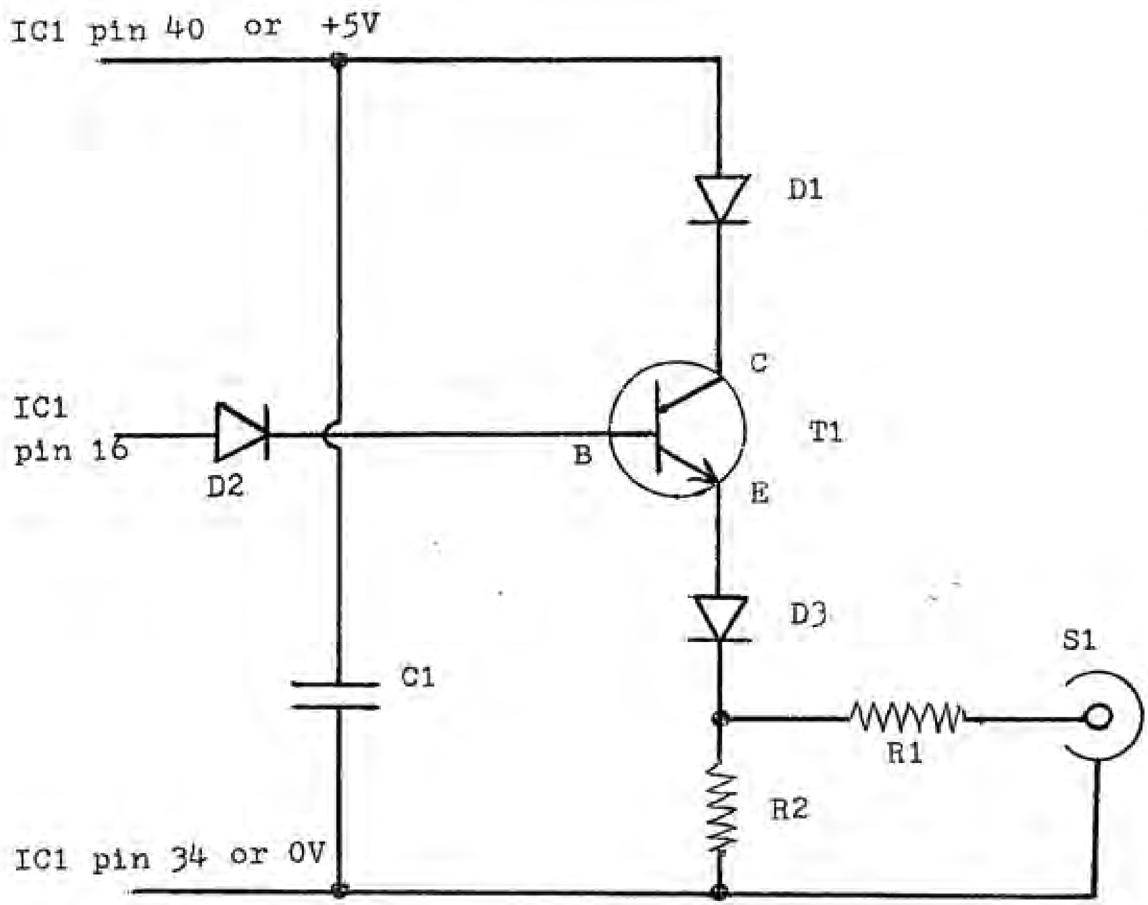
diodes. One other is placed between IC1 pin 16 and the transistor base and one is placed between the +5 volt supply and the collector of the transistor. It works perfectly now, just like the expensive conputers. By the way, with this modification you can leave the modulator hooked up and in fact you can drive both a video monitor display and a television display at the same time. I have found this handy in letting others see what's happening on the computer when you have an audience.

ZX PRINTER AND THE TS2068

The only thing preventing the operation of the Sinclair ZX Printer with the TS2068 is the lack of a 9 volt supply. All other pins on the ZX Printer are compatible with the TS2068 peripherial connector on the rear of the computer. I made the printer work by connecting a power socket to the ground pin and the 9 volt pin of the printer. I did this by soldering the leads of the power socket to an extender board inserted between the computer peripherial connector and the printer edge connector. The same modification can also be made by removing the cover from the printer edge connector and soldering the power socket leads to the edge connector.

Here is a step by step description of the printer edge connector modification. Turn the connector upside down, exposing the screw heads. Remove both screws, noting that the

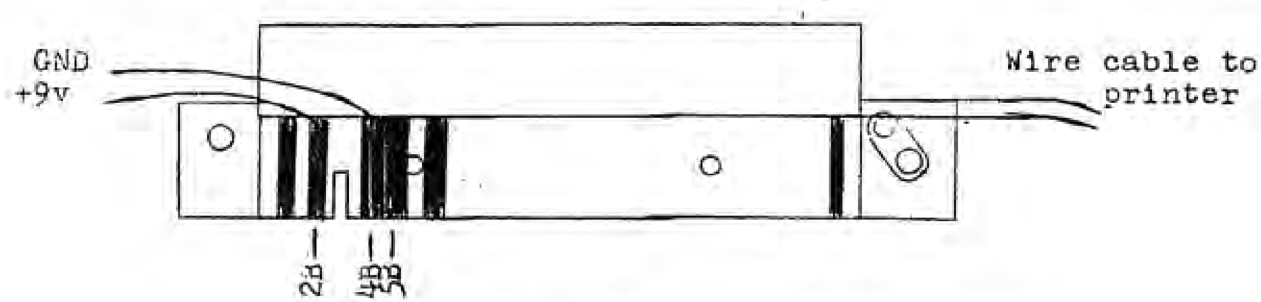
Figure 12-2. Video monitor circuit schematic.



T1	2N2221 transistor
D1-D3	1N4148 diodes
C1	.1 uf capacitor
R1	33 ohm ¼ watt resistor
R2	100 ohm ¼ watt resistor
S1	RCA type phono socket

shorter screw came from the wire cable end of the assembly. Prepare two, three inch pieces of hookup wire by striping 1/8 inch of insulation from each end. Referring to Figure 12-3 for the following steps, solder one wire to the foil strip on the jumper board, close to where the light blue wire is soldered to the foil. This is the foil strip next to the connector key-way slot, on the side of the slot furthest from where the wire cable enters the connector. This corresponds to connector finger 2B as described in your ZX81 or TS1000 user manual, or the 9 volt line. Solder the other wire to the double wide foil strip, on the other side of the slot, close to where the green wire is solder. This is finger 4B + 5B or the ground line. Route both wires out the closest opened end of the connector assembly, being careful not to pinch the wires between the two halves of the connector cover. Remember which wire was the 9 volt line and which was the ground line. Now put the two halves of the cover back over the connector and tighten the screws. Note that the metal lug with the hole has to be aligned so that the short screw goes through the center of the hole. Lastly, solder the 9 volt wire to the center terminal of a 1/8 inch jack (Radio Shack 274-333) and solder the ground wire to the outside or ground terminal of the jack.

Once the above modification has been made, turn off the TS2068 power; Plug the printer edge connector into the computer; turn on the computer; and, plug the original ZX81 or TS1500 power supply into the power socket jack you soldered to the printer. The full capabilities of the TS2068, including single dot printing, are now available through your good old trusty ZX Printer.



View from Bottom or screw head side with Bottom Half of Cover Removed.

Figure 12-3. ZX Printer connector modification for use with the TS2068.

In the remainder of this chapter, I will try to expand upon some of the software procedures and techniques that were mentioned or used earlier in the book. Most of the discussion will be devoted to machine code, which may be difficult for you to master if you depend on digging the answers out of the reference material, like I had to do. There have been many books written to help you understand and use machine code, but unfortunately, much that has been written is difficult or impossible to use if you don't already have some understanding of the subject. Recommended references are listed in Appendix A of this book.

First, let me clear up one possible confusion in the terminology of machine code. Throughout this book I refer to "machine code" when technically I really mean Z80 assembler or Z80 assembly language. I use the term machine code because I picked it up in the Sinclair literature and continue to use it. The basis for the term comes from the fact that you enter Z80 assembler instructions into program lines in the computer by entering a decimal number code for each instruction. You POKE a code, for each instruction, into an address location which holds it for program execution.

MACHINE CODE IN THE REM STATEMENT

In the ZX81/TS1000/TS1500 computers, the machine code is held in a specific address location which corresponds to the beginning of your BASIC program listing area. The address area begins at address 16514. When you POKE code into the computer starting at this address location (16514) it will appear in the first REM statement, if the REM statement is the first BASIC statement of your program. Remember in Chapters 3, 5, 6, 8, 10 and 11 when you made space for the machine code by starting your listing with a REM statement and filling it with spaces? If you didn't realize what you were doing, fire up your computer and follow me through the demonstration that follows. Do not RUN any or these listings, as they may cause the computer to destroy the listing and you will have to turn off the computer to get it to reset.

First, enter a "1 REM" statement to begin the listing. Then ENTER "POKE 16514,1". Now LIST the program and see what happened. The screen should have several strange symbols after the 1 REM statement that have generally invaded the screen. It should be obvious that you have done something wrong. What you did was to POKE a number (code) into a location that was already occupied by code that the computer was using for some BASIC housekeeping function. Turn the computer off, then turn it back on and enter the listing in Table 12-1. But, after you type "1 REM" and before you ENTER the statement, type at least ten "spaces". This will reserve room in the REM statement for the machine code.

Now, after the listing in Table 12-1 is entered, RUN the program by entering "GOTO 2". As each address location is displayed on the screen, type any number between 1 and 20, and press "ENTER". After you enter the number for the last address (16523), LIST the program. Notice that some strange looking symbols have been placed in the REM statement. Open your computer's User Manual to the first page of the Character Set

Table 12-1
MACHINE CODE ENTRY ROUTINE

```
1 REM
2 FOR N = 16514 TO 16523
3 PRINT AT 10,10;N
4 INPUT I
5 POKE N,I
6 NEXT N
```

Appendix. You were entering code into the computer and the corresponding character is what appeared in the REM statement.

This ability to hold machine code in the first REM statement is unique to the Sinclair computer operating system. The only caution you need to know about this is that the REM statement with the machine code must always be the first statement in the program listing.

To get into a machine code routine you need to use a "USR" routine in BASIC with the appropriate address. Refer to Table 3-4 (Chapter 3) for better understanding of how this works. The RX81 "OUT" routine is accessed by a "LET OUT=USR 16533" statement. This directs the program into the machine code routine at address 16533. The computer runs the routine in descending sequence until it reaches address 16537, which is a machine code "return" instruction (code 201 or character TAN). It then exits the machine code routine and returns to the next statement after the originating "USR" statement in the BASIC program listing. Study the "IN" portion of the routine in Table 3-4 to better understand the input functions of the machine code routine. This routine will be used as an example in the discussion that follows.

MACHINE CODE RELATIVE ADDRESSING

If you are not familiar with machine code programming and would like to understand more about the machine code routines used in this book, you may have studied the RX81 machine code routine that first appeared in Chapter 3 (Table 3-4). The notation in Table 3-4 is self explanatory, except for the Jump Relative (JR) activity in the "IN" routine. This routine is designed to prevent false inputs from the input switches/pulses. For example, input from a microswitch as is used on the robot in the form of bumper switches, can be read incorrectly by the computer if there is any switch bounce. To prevent this false input, the input routine reads the input three times. If the second input is not the same as the first, the routine aborts the original reading and starts all over again. Likewise if the second reading of the input agrees with the first, but the third does not agree, the routine is started again. To accomplish this, the routine compares the first input with the second and third at addresses 16521 and 16527 respectively.

If the second or third readings of the input are not the same as the first reading, the routine jumps back to the beginning from addresses 16522/16523 and 16528/16529 respectively. The routine jumps back using a negative number by counting backwards up the routine. For example, as depicted in

Table 3-4, to jump relative to the beginning of the routine from address 16523 the count is -10. If you count, starting with address 16523, you reach 10 at address 16514, the beginning of the routine.

The conversion of the -10 number into a hex or decimal number which is inserted in the address location is not quite as simple to explain. Because the eight bit computer can only count from 0 to 255 using positive numbers, an innovative solution is required. An initial look at Table 12-2 will probably not reveal the solution. The system uses the most significant binary digit (the left hand one in the table) as an indicator for the sign, positive or negative. A zero (0) in this position indicates a positive number and a one (1) indicates a negative number. That leaves only seven positions of the binary number, or bits, to count with. In other words, you can only relative address from 127 counts forward (+127) to 128 counts backward (-128). I only included the relative addressing counts from -1 to -32 in the displacement table (Table 12-2) as all examples in this book fall in that range. If you have a need to convert a number not listed in this table, refer to the Z-80 Microcomputer Handbook listed in Appendix A.

The other type of machine code jump used by the Z80 chip is a direct address jump. An address conversion table is included at Table 12-3, for your information. Note that when using this type of jump, the address is converted to hex occupying two consecutive addresses. However, when the location is inserted in the routine the two hex numbers are reversed.

USEFUL SOFTWARE PROGRAMMING TOOLS

You may have wondered how I compiled the long 13K robot control program depicted in Table 5-10 (Chapter 5), without a great deal of keyboard entry time. As I explained in Chapter 10, the voice control input machine code software was purchased on cassette from G. Russell Electronics. Did I start with that software loaded into my computer and then enter that giant 13K BASIC listing line-by-line from the keyboard? No, there is an easier way. What you need is some software which will let you merge two separate programs. The 13K program, minus the voice recognition machine code, was written over a two year period with many revisions as the robot control was made more sophisticated and more hardware peripherials were added to the system. I stopped counting revisions at number 37 about two years ago.

The software I used to merge the two programs was "Z-TOOLS" from Sinware, Box 8032, Santa Fe, NM. Although the instructions that come with the software state that it is to be used to merge BASIC programs, it can be used successfully to merge the voice recongnition machine code with the BASIC listing. The reason it works is because the machine code is contained in the "1 REM" statement, which is handled ike any other BASIC statement by the computer. You cannot however merge machine code carried in two separate REM statements.

If you merge the voice regcognition software from one cassette with the robot control software from another cassette, remember to delete the "1 REM" statement from the robot control software and use a cassette recording of such a revised program

Table 12-2
MACHINE CODE RELATIVE ADDRESSING DISPLACEMENT TABLE

Key	Displacement Value			
	Count =	Binary	Decimal	Hex
COPY	-1	11111111	255	FF
RETURN	-2	11111110	254	FE
CLEAR	-3	11111101	253	FD
UNPLOT	-4	11111100	252	FC
CLS	-5	11111011	251	FB
IF	-6	11111010	250	FA
RAND	-7	11111001	249	F9
SAVE	-8	11111000	248	F8
RUN	-9	11110111	247	F7
PLOT	-10	11110110	246	F6
PRINT	-11	11110101	245	F5
POKE	-12	11110100	244	F4
NEXT	-13	11110011	243	F3
PAUSE	-14	11110010	242	F2
LET	-15	11110001	241	F1
LIST	-16	11110000	240	F0
LOAD	-17	11101111	239	EF
INPUT	-18	11101110	238	EE
GOSUB	-19	11101101	237	ED
GOTO	-20	11101100	236	EC
FOR	-21	11101011	235	EB
REM	-22	11101010	234	EA
DIM	-23	11101001	233	E9
CONT	-24	11101000	232	E8
SCROLL	-25	11100111	231	E7
NEW	-26	11100110	230	E6
FAST	-27	11100101	229	E5
SLOW	-28	11100100	228	E4
STOP	-29	11100011	227	E3
LLIST	-30	11100010	226	E2
LPRINT	-31	11100001	225	E1
STEP	-32	11100000	224	E0

Table 12-3
ADDRESS LOCATION CONVERSION TABLE

Decimal	Hex	Insert in reverse order in MC instruction
16514	40 82	i.e. 82 40
16515	40 83	83 40
16516	40 84	etc.
16517	40 85	
16518	40 86	
16519	40 87	
16520	40 88	
16521	40 89	
16522	40 8A	
16523	40 8B	
16524	40 8C	
16525	40 8D	
16526	40 8E	
16527	40 8F	
16528	40 90	
16529	40 91	
16530	40 92	
16531	40 93	
16532	40 94	
16533	40 95	
16534	40 96	
16535	40 97	
16536	40 98	
16537	40 99	
16538	40 9A	
16539	40 9B	
16540	40 9C	
16541	40 9D	
16542	40 9E	
16543	40 9F	
16544	40 A0	
16545	40 A1	
16546	40 A2	
16547	40 A3	

in your merging operation. Also note that any remaining lines in the robot control listing that have the same line number as the voice recognition listing (2 REM, for example) must be changed before the merge operation.

After you have successfully merged the two programs, do not RUN the new consolidated program yet. You have to first put the RX81 machine code back into the program. It cannot however go into the address locations you put it into during your Chapter 3 software writing. The voice recognition machine code now occupies those address locations. The voice recognition has however reserved space within the 1 REM statement for any machine code you want to add. The reserved area is from addresses 17603 to 18056.

If you will study the end of the 1 REM statement in Table 5-10, you will see that I not only added the RX81 I/O machine code routine for robot control but I followed it with the optical encoder machine code routine. The optical encoder is not implememted however in the program. POKE the RX81 I/O machine code into the voice recognition by changing the FOR TO routine to addresses 17858 TO 17881. In other words, the addresses for the code in Table 3-3 would start at address 17858 (vice 16514) and end with address 17881 (vice 16537). The addresses for the USR statements in the 13k listing (Table 5-10) already reflect these new address locations.

Appendix A
REFERENCE PUBLICATIONS

"Build This Robot for Under $400", Gupton, Radio-Electronics magazine Special Reprint, 1981

"Build Your Own Working Robot", Heiserman, TAB Books 1976

"The Explorers Guide to the ZX81", Lord, Timedata 1982

"Linear Databook", National Semiconductor Corporation, 1982

"Logic Databook", National Semiconductor, 1981

"Master Handbook of Microprocessor Chips", Adams, TAB Books 1981

"Timex Sinclair 2068 Personal Color Computer User Manual", Durang, Timex 1983/Sinclair 1982

"Timex User Manual", Vickers, Timex 1982/Sinclair 1982

"TTL Cookbook", Lancaster, Sams 1974

"The Z-80 Microcomputer Handbook", Barden, Sams 1978

"ZX81 Basic Programming", Vickers, Sinclair 1981

THIS PAGE INTENTIONALLY BLANK

Appendix B
PARTS SOURCES

All Electronics Corp.
P.O. Box 20406
Los Angeles, CA 90006

Budget Robotics & Computing
Box 18616
Tucson, AZ 85731

DAK Industries, Inc.
8200 Remmet Ave.
Canoga Park, CA 91304

Edmund Scientific Co.
101 E. Gloucester Pike
Barrington, NJ 08007

Electronic City
801 Broadway
Tucson, AZ 85719
(602-622-1173)

Digi-Key Corp.
P.O. Box 677
Thief River Falls, MN 56701

Gledhill Electronics
P.O. Box 6884
San Francisco, CA 94101

Heath Co.
Benton Harbor, MI 49022

H & R Corporation
401 E. Erie Ave.
Philadelphia, PA 19134

JAMECO Electronics
1355 Shoreway Road
Belmont, CA 94002

Knapp of Florida, Inc
4750 96th St. N.
St. Petersburg, FL 33708

Mouser Electronics
11433 Woodside Ave.
Santee, CA 92071

Polaroid Corp.
Commercial/Battery Division
784 Memorial Drive
Cambridge, MA 02139

The Robot Works
Box D6
Washington Mills, NY 13479

Sineware
Box 8032
Santa Fe, NM

Winfred M. Berg Inc.
499 Ocean Ave.
East Rockway, L.I., NY 11518

THIS PAGE INTENTIONALLY BLANK

INDEX

THE FOLLOWING PAGES ARE SUPPLIED AS A BONUS FOR ANYONE INTERESTED IN TEACHING HIGH SCHOOL LEVEL BEGINNING ROBOTICS COURSES.

THE MATERIAL INCLUDES COURSE DESCRIPTIONS, COURSE OUTLINES, PARTS LISTS WITH SOURCES AND PLATFORM CONSTRUCTION DRAWINGS FOR TWO, ONE SEMESTER COURSES (ROBOTICS 1 AND ROBOTICS 2).

THIS MATERIAL ALONG WITH THE DETAILED INFORMATION IN THE BOOK SHOULD BE SUFFICIENT FOR AN ELECTRONICS/COMPUTER QUALIFIED INSTRUCTOR TO PREPARE FOR AND TEACH THE COURSES.

COURSE DESCRIPTION

Curriculum Unit:

Course Title: ROBOTICS 1	Grade Level(s): 10-12
Semester #s:	Prerequisites: Computers & Basic or equivalent
Quarter #s: none	
Course Type: Optional	Prerequisite for: Robotics 2

PURPOSE: To familiarize students with the fundamentals of input/output and control circuits in robotics. It is a laboratory oriented class in which students learn the electrical, mechanical and programming aspects of robotics by actually building a robot.

STUDENT LEARNING OUTCOMES -- The student will be able to:

c 1. Display an understanding of voltage, resistance and amperage by solving each of the values in the formula $V = I \times R \quad I = \frac{V}{R} \quad R = \frac{V}{I}$

c 2. Recognize DC symbols and terminology used in appropriate circuit diagrams/schematics.

c 3. Demonstrate an understanding of capacity and filtering in DC power sources including power surges, spikes, frequency interference, the use of hash chokes, and large capacitors.

c 4. Explain the application of motors in control applications and the difference between AC & DC motors.

p 5. Build an LED indicator circuit and operate it using a battery.

p 6. Combine battery, LED circuit, and motor to work together.

c 7. Identify in schematic depiction the elements which comprise a computer's central processing unit (CPU) as they relate to computer input/output (I/O).

c, a 8. Display an ability to understand binary numbers, the machine code and BASIC language required to operate computer I/O devices.

c, p 9. Follow schematic and written instructions which will lead to the successful completion of building a motor control circuit.

c 10. Demonstrate the ability to write software routines necessary to control electric motor.

c,a,p 11. Display an ability to understand basic robotic control theory as it pertains to assembling circuits and components necessary to design and build a mobile platform.

c,p 12. Demonstrate the implementation of an input circuit necessary for a robot to sense the environment.

c 13. Discuss types of mechanical arms and advanced I/O.

TO FACILITATE STUDENT LEARNING OUTCOMES -- The teacher will:

C 1. Provide and arrange for a laboratory enviornment conducive to the successful completion of laboratory and practical assignments.

D 2. Illustrate subject matter through written material, lecture, and audio visual aids.

C 3. Evaluate and measure student progress providing feedback to individual students.

I 4. Encourage further study of robotics by providing an opportunity for discussion of applications.

D,I 5. Discuss social and ethical issues relating to robots.

D 6. Present lectures on related theory, mathematics, electrical, hardware and software principles.

I 7. Review and continue to build on student outcomes of prerequisite courses.

ROBOTICS 1
EXPANDED COURSE OUTLINE

1. DC circuits: Review AC versus DC electricity emphasizing polarity.

a. Volts, resistance, amperage: E = I x R

Give examples using power sources, resistors, diodes and LEDs.

b. Reading circuit diagrams/schematics: Become familiar with symbols for terms above plus others to be used in the course.

c. Basic digital logic: Hi and Lo, +5 volts and zero volts or ground. Show transistor circuit and lead into several digital logic integrated chips and circuits.

d. Power sources: Battery, rectification of AC and regulator circuits building on logic learned above.

(1) Capacity: Using E, I and R from above explain capacities of various sources.

(2) Filtering: Explain power surges, spikes and frequency interference (using TRON example) and problems they can cause. Explain use of hash choke filters, large capacitors, etc.

Note: Have all components in classroom to show when explaining on blackboard. Make reference to how they will be used when actually building later in the course.

2. Motors & Circuit Status Indicators: Explain the application of motors in control applications (i.e. wheels, traveling screws, X and Y axis positioning, stepper motors, brushless and brush motors. Cover applications such as printers, plotters, robot arms, etc.

a. LED indicator circuit: Draw polarity/direction indicating circuit based on material previously covered. Actually build circuit on small PC board and operate it using a battery.

b. DC Motors: Basic differences between AC and DC motors.

(1) Direction: Understand polarity and gearing ramifications.

(2) Speed: Understand voltage, gearing and drive wheel size implications.

Connect battery, LED circuit and motor to demonstrate them all working together.

3. Computer Input/Output (I/O): Diagram workings of computer in combination with I/O circuit.

a. Parallel port: Diagram circuit and explain how it works starting with computer CPU and ending with I/O lines on I/O board. Explain binary numbers.

(1) Data lines: CPU addressing to I/O board.

(2) Control lines: CPU descriminators.

(3) Decoding: Sorting out signals and determining I/O lines.

b. Software: Explain machine code and cover appropriate MC and BASIC language routines. Refer back to hardware descriptions. and schematic covered above.

(1) Address location: Explain addressing required for each I/O board. Write software to be used in control of platform to be built later.

(2) Line selection: Again referring back to hardware write software required control specific I/O lines.

Note: Above is done using the ZX81/TS1000 computer and RX81 I/O board and its documentation as instructional aids.

4. Control circuit interfacing motor with computer: Continuing from I/O schematics covered above, complete output circuits required to control an external device such as the LED and motor which are going to be a part of this project.

a. Low versus High voltage circuits: Using examples like an LED and a motor, determine what type of control circuit is required. Learn why different approaches are most logical (i.e. over design vs underdesign) and what is required as a minimum.

b. Transistor switches: Medium current control circuits and how they work.

c. Relays: Mechanical and solid state types and their application and cost. Speed considerations. Learn about physical characteristics and capabilities of various types.

d. Circuit noise: Expand upon filtering previously covered and learn how to control noise to acceptable levels.

e. Software

(1) On-Off control: Learning how to control multiple tasks by time sharing.

(2) Timing: Learn timing versus input methods. Use software techniques such as PAUSE and FOR NEXT. Learn how these techniques may be effected by the way other software routines are written.

Note: Control circuit for one motor will be constructed at the end of this portion of instruction.

5. Platform/Base with one motor: Plywood base will be designed and constructed.

a. Motor versus platform speed: Capabilities of motor selected will be studied and thoroughly understood.

b. Steering: Steering methods will be studied but not applied in constructing this platform.

c. Platform size & loading: Various sizing and loading will be studied but size for platform constructed will be predetermined. Considerations for household platforms will be studied.

d. Materials: Survey of all types of material will be conducted.

Note: At this point the one motor platform with power supply, computer, I/O board, LED circuit and one motor will be completed and demonstrated.

6. Input sessor: Expansion of previous I/O instruction to include simple input hardware and software. Previous schematics will be used and expanded.

a. Mechanical switch sensors: Explain simple open/closed input. Micro switch will be used.

b. Software: Expansion of I/O software explanation to include input application. Software will be written and integrated with existing software.

Note: The following material will be covered only to make students familiar with the concepts and give them a glimpse of what to expect if they continue thier study in an advanced robotics course.

7. Advanced applications

a. Mechanical

(1) Arms

(a) Movement axis

(b) Traveling screw

(2) Multiple motors

b. Advanced I/O

(1) Optical encoder

(2) Ultrasonic ranging

(3) Voice recognition

(4) Digital voice

The following is a time allocation plan for teaching the above course as a 1 semester, 18 week course:

WEEK(s)	
1	1. DC circuits
2-4	2. Motors & circuit status indicators
5-8	3. Computer input/output (I/O)
9-11	4. Control circuit interfacing
12-15	5. Platform/base with motor
16-17	6. Input sensor
18	7. Advanced applications

COURSE DESCRIPTION
Curriculum Unit:

Course Title: ROBOTICS 2
Semester #s:
Quarter #s: none
Course Type: Optional

Grade Level(s): 10-12
Prerequisite: Robotics 1 or consent of instructor
Prerequisite for: N/A

PURPOSE: To reinforce and extend the skills learned in Robotics 1. Students will build a robot with multiple motors and sensors that can interact with its enviornment.

STUDENT LEARNING OUTCOMES -- The student will be able to:

c 1. Display an understanding of DC circuits, motors and circuit status indicators, computer input/output, and interfacing control circuits with computers learned from Robotics 1.

c 2. Demonstrate an understanding of the theory involved in the operation of buffered buss expansion circuits.

p 3. Build a buffered buss expansion board and operate it connected to the expansion slot of the computer.

c 4. Explain the application of optical encoders used in conjunction with robot arms and drive mechanisms.

c 5. Identify in schematic depiction the elements of a simple optical encoder circuit and describe the input/output control circuit required by the optical encoder.

c,a 6. Demonstrate the ability to write software routines necessary to implement an optical encoder.

c,a 7. Compare techniques used to create a computer generated voice.

c,a 8. Demonstrate an understanding of various sensors available for robotics applications with emphasis on ultrasonic ranging devices.

c,p 9. Follow schematic and written instructions to construct a computer controlled ultrasonic ranging device.

c 10. Demonstrate the ability to implement software routines necessary to control the ranging sensor.

c,a,p 11. Build a robot with sensors and two motors.

c 12. Demonstrate an understanding of Artificial Intelligence (AI) as applied to robot navigation.

c,a 13. Discuss uses of personal robots and social implications.

TO FACILITATE STUDENT LEARNING OUTCOMES -- The teacher will:

C	1.	Provide and arrange for a laboratory environment conducive to the successful completion of laboratory and practical assignments.
D	2.	Illustrate subject matter through written material, lecture, and audio visual aids.
C	3.	Evaluate and measure student progress providing feedback to individual students.
I	4.	Encourage further study of robotics by providing an opportunity for discussion of applications.
D, I	5.	Discuss career opportunities relating to robots.
D	6.	Present lectures on related theory, mathematics, electrical, hardware and software principles.
I	7.	Review and continue to build on student outcomes of prerequisite courses.

ROBOTICS II
EXPANDED COURSE OUTLINE

1. Review of Robotics I course material as it applies to this course.

a. DC circuits

b. Motors and circuit status indicators

c. Computer Input/Output

d. Interfacing control circuit with computer

e. Input circuits

Note: Have all components in classromm that were built during first semester course to refresh memories and show how they will be used in this course.

2. Requirement for expansion board in implementing complex computer control projects.

a. Explain reasons expansion board is required as listed below.

(1) Multiple input/output boards

(2) Additional power requirements

(3) Buffering input/output

b. Explain theory of buffered expansion board operation.

c. Construct buffered buss expansion board from a kit of parts.

3. Refinement of motor control through use of optical encoder on the motor shaft: Explain the use of optical encoders in applications such as robot arms and drive mechanisms. Describe the construction and implementation of a simple optical encoder circuit.

a. Describe an optical encoder constructed from an infrared Light Emitting Diode (LED) and phototransistor.

b. Discuss the input/output control circuit required for an optical encoder.

c. Describe requirements of optical encoder software.

(1) Demonstrate requirement for high speed machine code software subroutines.

(2) Demonstrate usefulness of BASIC language control software.

(3) Discuss and demonstrate the requirement to mix machine code and BASIC language software due to physical characteristics of measurements optical encoder is required to make.

4. Digital Voice output: Discuss reasons for adding human voice to robotic equipment.

a. Comparison of techniques used to create computer generated voice.

(1) Discuss the phoneme technique of voice generation including its advantages and shortfalls.

(2) Discuss the technique using entire words stored in digital form for use in voice generation including advantages and disadvantages.

b. Detailed study of example system using complete words stored on Read Only Memory (ROM) chips.

(1) Explain the function of the speech processor circuit.

(2) Describe ROM storage techniques.

(3) Study the technique of programmable input/output in controlling this voice output system.

(4) Explain and demonstrate software techniques used to produce voice output.

5. Ultrasonic transducer sensors: Discuss the application of various senses available for robotics applications with emphasis on the usefulness of ultrasonic ranging devices.

a. Building an ultrasonic system based on the Polaroid Original Equipment Manufacturers (OEM) kit.

(1) Explain the operation of the transducer circuit.

(2) Power supply: Expalin requirements and build circuit.

(3) Input/output circuit & control: Explain requirements.

b. Applications through software: Explain theory and write software for:

(1) Ranging

(2) Motion sensing

c. Integration with other sensors: Determine requirements based on overall robot platform configuration and implement required hardware and software changes/modifications.

(1) Hardware: Use various examples complicating the

problem.

(2) Software: Emphasize interaction of various components and event timing problems.

6. Mobile platform/base with two motors: Plywood base will be designed and constructed.

a. Direction control techniques explained and discussed.

(1) Steerable wheel technique will be studied but not implemented.

(2) Two wheel drive steering will be studied and implemented.

b. Construction requirements will be studied to include platform size and loading; materials selected; and, two drive motor platform will be constructed.

7. Artificial intelligence (AI) in navigation: AI applications in robot navigation will be studied and applied for platform constructed for this course as time permits. The following considerations will be included:

a. Sensor input

b. Map creation/storage

c. Avoidance versus seeking techniques

8. Usefulness of personal robots: Survey and discussion of possible uses for personal robots to include the following:

a. Security

b. Messenger

c. Environment monitor

d. Companionship

e. Education

PARTS LISTS for ROBOTICS COURSES

ROBOTICS 1

Computer (ZX-81 or TS1000) Zebra Sys TS 1000		$49.95
RX81 I/O board kit BR&C RX-81A		25.00
12 volt Brevel gear motor H&R Q5658		25.00
Battery RS23-007 ER 732 Ray-O-Vac 926		10.39
Power supply for computer		
7805 voltage regulator RS276-1770		1.59
Heat sink RS 276-1363		.79
2200 uF 35v Capacitor RS-1020		2.49
Hash chokes 100 uH 2 amps RS273-102 2 each		1.98
PC board RS276-168		1.95
Power supply for relay coils		
7805 voltage regulator RS276-1770		1.59
Heat sink RS276-1363		.79
1 MFD 30v Capacitors RS272-996 3 each (2 here 1 above)		2.37
Transistor switches/DIP relays/12 volt coil relays (2ea)		
2222A Transistors RS276-1617 Package of 15(1.98) 2 ea =		.26
Switching diodes 1N914/4148 RS276-1620 Package of 50 (1.98)	10 ea =	.40
DIP DPDT relays 5 volt coil RS275-215 (3.99 x 2)		7.98
1/4 watt 220 ohm resistors RS271-1313		.39
.01 uF ceramic capacitors RS272-431	2 ea	.59
22 uF elec. capacitors RS272-1026	.69 x 2ea	1.38
1N4003 diodes RS276-1102 2 ea		.59
PC board RS276-168		1.95
LED indicator circuit		
PC board	estimate	1.00
LEDs 2 ea	"	.20
1N4003 diodes RS276-1102 2 ea		.59
220 ohm resistors 2 ea	included above	----
1ea 4 inch diamater wheel with 1/4 inch hole	estimate	2.00
2ea 1 1/4 inch diameter caster wheels	"	2.00
1ea Microswitch	"	1.50
2 sq. ft. 1/2 inch plywood	"	1.00
Approx. 2 feet 1/2 inch "L" aluminum	"	2.00
Misc. screws	"	.50
Misc. nuts & bolts	"	2.00
Approx. 10 feet hookup wire	"	.50
Rosen core solder	"	.50
		$151.22

ROBOTICS 2

Computer (ZX-81 or TS1000) Zebra Sys TS1000	$49.95
16K Memory Zebra Sys TS1016	10.00
Expansion board kit BR&C EXP-1A	80.00
RX81 I/O board kit BR&C RX-81B 2 @ 21.00 ea.	42.00
Optical Encoder circuit	
33K ohm 1/4 watt resistor RS271-1341 5 for .39	.08
330 ohm 1/4 watt resistor RS271-1315 5 for .39	.08
Light emitting (IR) diode RS276-143	1.49
Phototransistor (IR) RS276-142	1.99
74LS14, 14 pin DIP, Hex Schmitt Triggers Elec Cty	.99
PC board estimate	1.50
Ultrasonic Ranging Circuit	
Polaroid OEM Kit (two sets for $99) Polaroid Corp.	49.50
2N2907 Transistor RS276-2023	.79
7404 DIP IC RS276-1802	.99
1 MFD capacitor RS272-996 4 ea. (2 for .79)	1.58
.01 MFD capacitor RS272-131 (2 for .59)	.30
1K ohm 1/4 watt resistor RS271-1321 5 for .39	.08
5.6K ohm 1/4 watt resistor RS271-031 2 for .19	.10
10K ohm 1/4 watt resistor RS271-1335 5 for .39	.08
470 ohm 1/4 watt resistor RS271-1317 5 for .39	.08
LM323K 3 amp 5 volt voltage regulator BR&C EXP-1R	9.00
Heat sink and mounting bolts included in above price	
10 feet hookup wire	.50
PC Board estimate	1.50
12 volt Brevel gear motors H&R Q5658 2 each	50.00
Battery RS23-007 ER 732 Ray-O-Vac 926	10.39
Power supply for expansion board	
LM317T voltage regulator RS276-1778	2.79
Heat sink RS 276-1363	.79
2200 uF 35v Capacitor RS-1020	2.49
Hash chokes 100 uH 2 amps RS273-102 2 each	1.98
1N4003 diodes RS276-1102 2 each	.59
120 Ohm 1/4 watt resistor RS271-1311 (5 for .39)	.08
5K ohm adj. pot. RS271-343	1.49
.1 MFD 25 volt Capacitor RS272-135 (2 for .39)	.20
PC board RS276-168	1.95
Power supply for relay coils	
7805 voltage regulator RS276-1770	1.59
Heat sink RS276-1363	.79
1 MFD 30 volt Capacitors RS272-996 3 @ .79 ea. 2 here 1 above	2.37
Transistor switches/DIP relays/12 volt coil relays (4ea)	
2222A Transistors RS276-1617 Package of 15(1.98) 4 ea =	.52
Switching diodes 1N914/4148 RS276-1620 Package of 50 (1.98) 20 ea =	.80
DIP DPDT relays 5 volt coil RS275-215 (3.99 x 4)	15.96
1/4 watt 220 ohm resistors RS271-1313 (5 for .39)	.78
.01 uF ceramic capacitors RS272-431 4 ea	1.18
22 uF elec. capacitors RS272-1026 .69 x 4ea	2.76
1N4003 diodes RS276-1102 (2 for .59)	1.18
PC board RS276-168	1.95

LED indicator circuits			
PC boards (2 ea)		estimate	2.00
LEDs 4 ea		"	.40
1N4003 diodes RS276-1102	(2 for .59)		1.18
220 ohm resistors	4 ea	included above	----
2ea 4 inch diamater wheel with 1/4 inch hole		estimate	4.00
1ea 1 1/4 inch diameter caster wheels		"	1.00
2ea Microswitch		"	3.00
3 sq. ft. 1/2 inch plywood		"	1.50
Approx. 4 feet of 1/2 inch "L" aluminum		"	2.00
Misc. screws		"	1.00
Misc. nuts & bolts		"	4.00
Approx. 20 feet hookup wire		"	1.00
Rosen core solder		"	.50
			$376.79

Common equipment & Tools (for ROBOTICS 1 & ROBOTICS 2)

Cassette recorder for program storage			$50.00
Printer for LISTing printouts	Zebra Sys PR32		$35.00
Television for monitor			-----
Ref. book "BUILD A MICROCOMPUTER-CONTROLLED ROBOT"		BR&C	$10.95
			$95.95

Table saw (to cut plywood)
Hasksaw (to cut "L" aluminum)
Electric drill & bits
Soldering Iron
Screw drivers
Needle nose pliers
Wire cutters

Other Misc. Consumables

Thermal printer paper RS26-1332	$3.95
Hookup wire RS278-1304	2.19
Multiconductor wire RS278-757	2.39
	$8.53

Parts Suppliers (for ROBOTICS 1 & ROBOTICS 2)

RS = Radio Shack stores

Zebra Sys = Zebra Systems, Inc.
78 - 06 Jamaica Ave.
Woodhaven, NY 11421 (718) 296-2385

H&R = H & R Corporation
401 E. Erie Ave.
Philadelphia, PA 19134 (215) 426-1708

BR&C = Budget Robotics & Computing
Box 18616
Tucson, AZ 85731 (602) 298-6095

Elec Cty = Electronic City
801 E. Broadway
Tucson, AZ (602) 622-1173

Polaroid Corp = Polaroid Corporation
Commercial/Battery Division
784 Memorial Drive
Cambridge, MA 02139

Other Hardware Sources (ZX81, TS1000, 16K RAM Pack, TS2040 or Alphacom 32 printer)

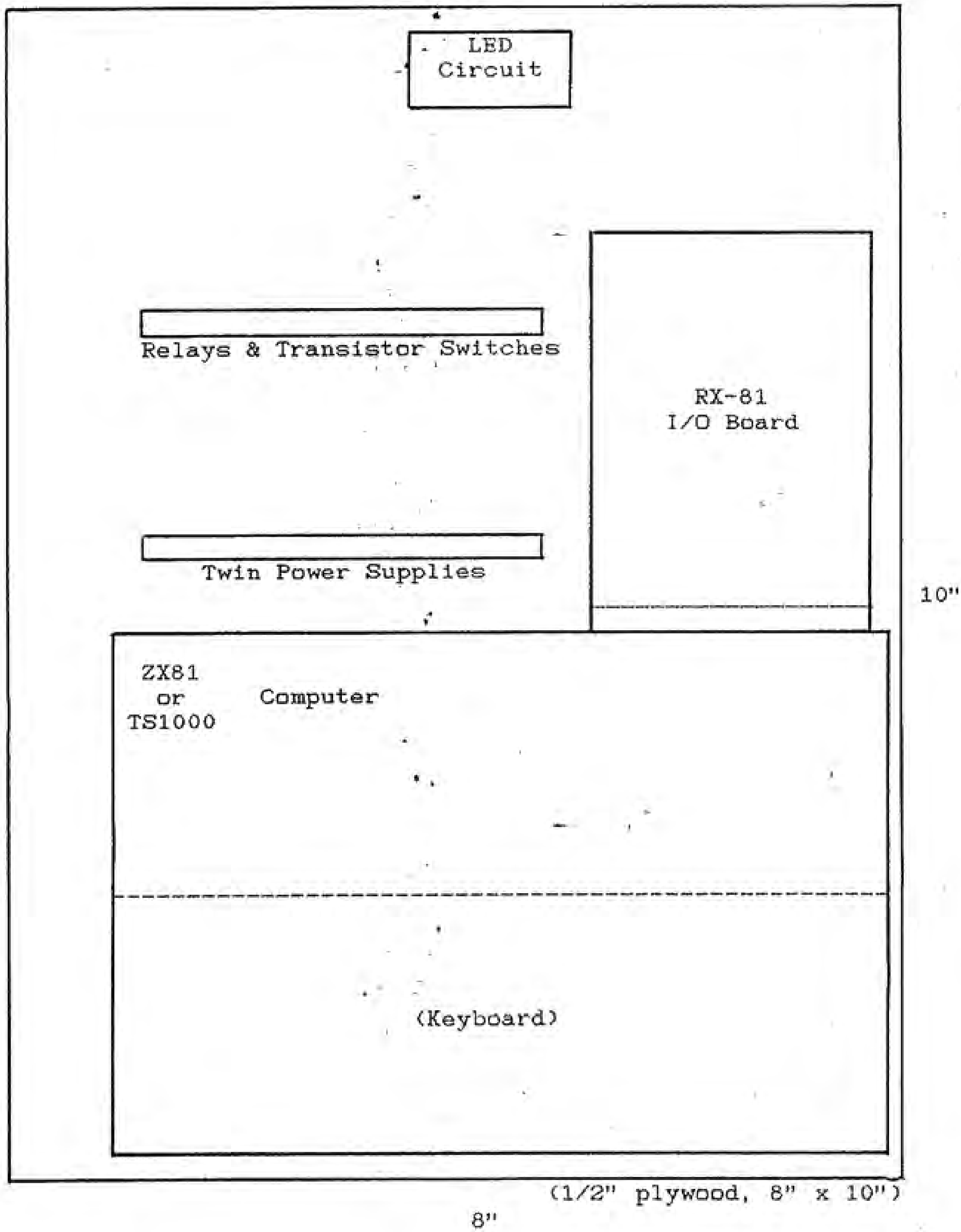
E. Arthur Brown
3404 Pawnee Dr.
Alexandria, MN 56308 (612) 762-8847
763-6393

Curry Computer
5344 W. Banff
Glendale, AZ 85306 (602) 978-2902

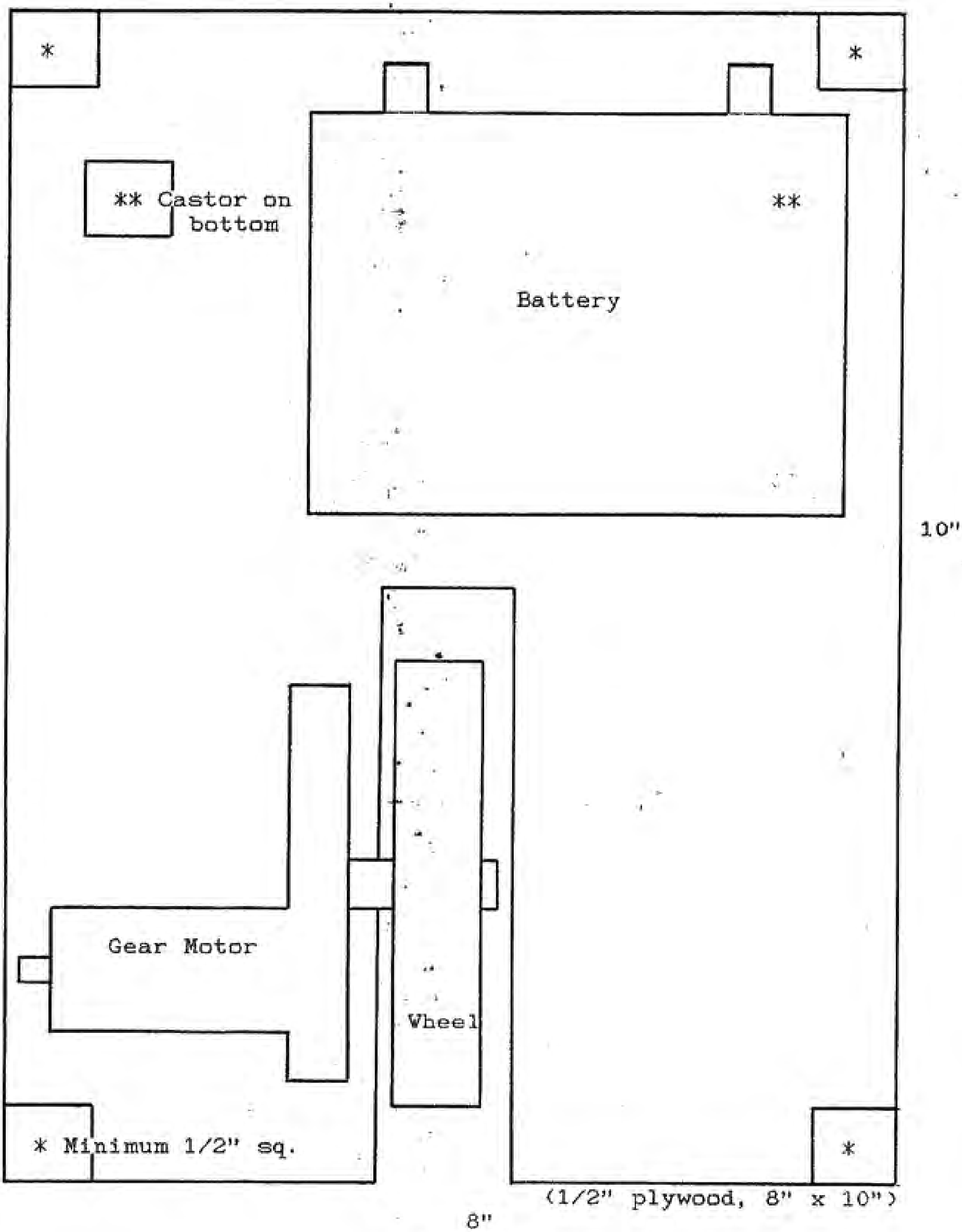
Games to Learn By, Inc.
P.O. Box 78
Collinsville, Conn. 06022 (203) 673-7089

Games to Learn By, Inc.
P.O. Box 575
Williamsburg, Mass. 01096 (413) 268-7505

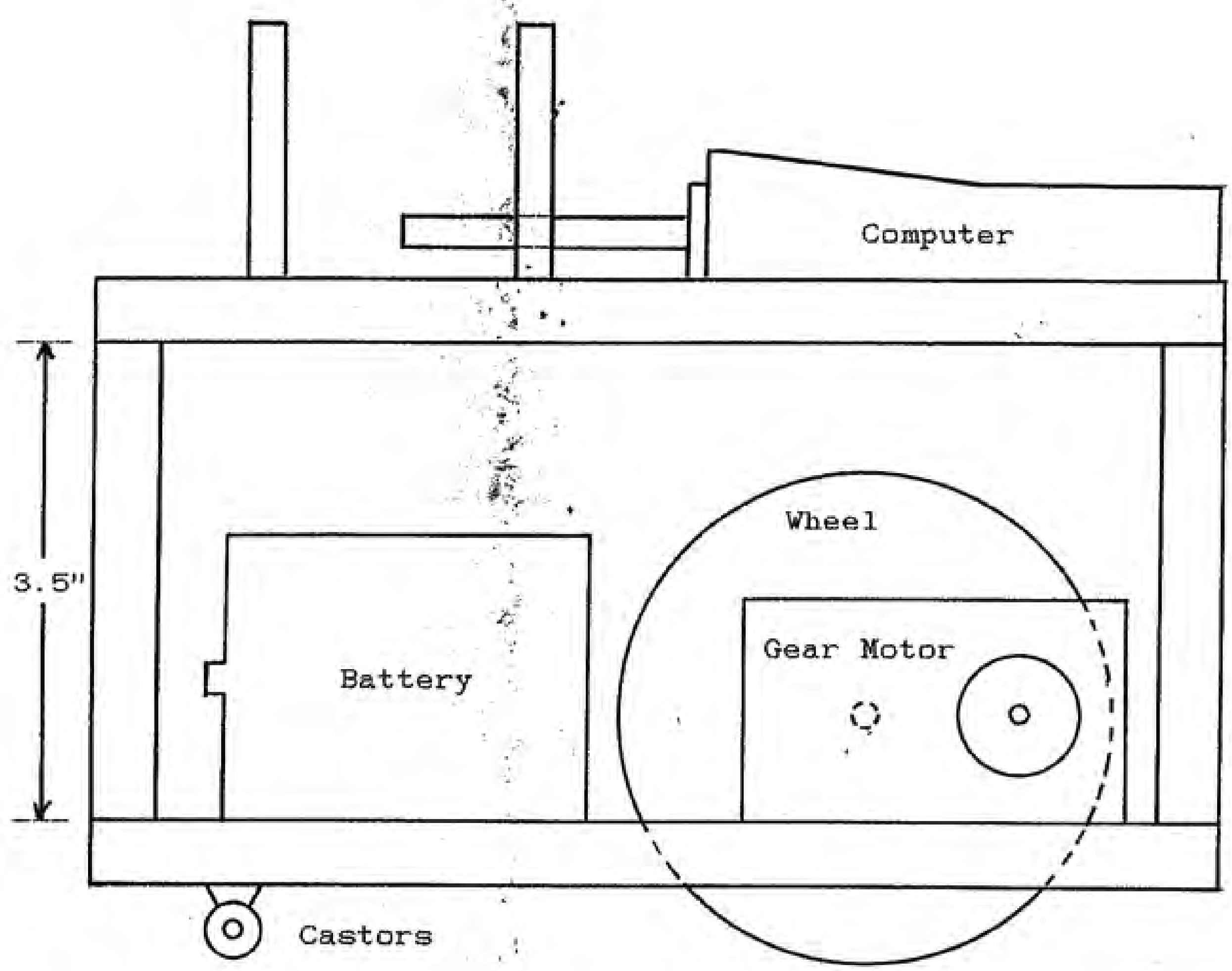
ROBOTICS 1 Mobile Platform
Upper Level Layout



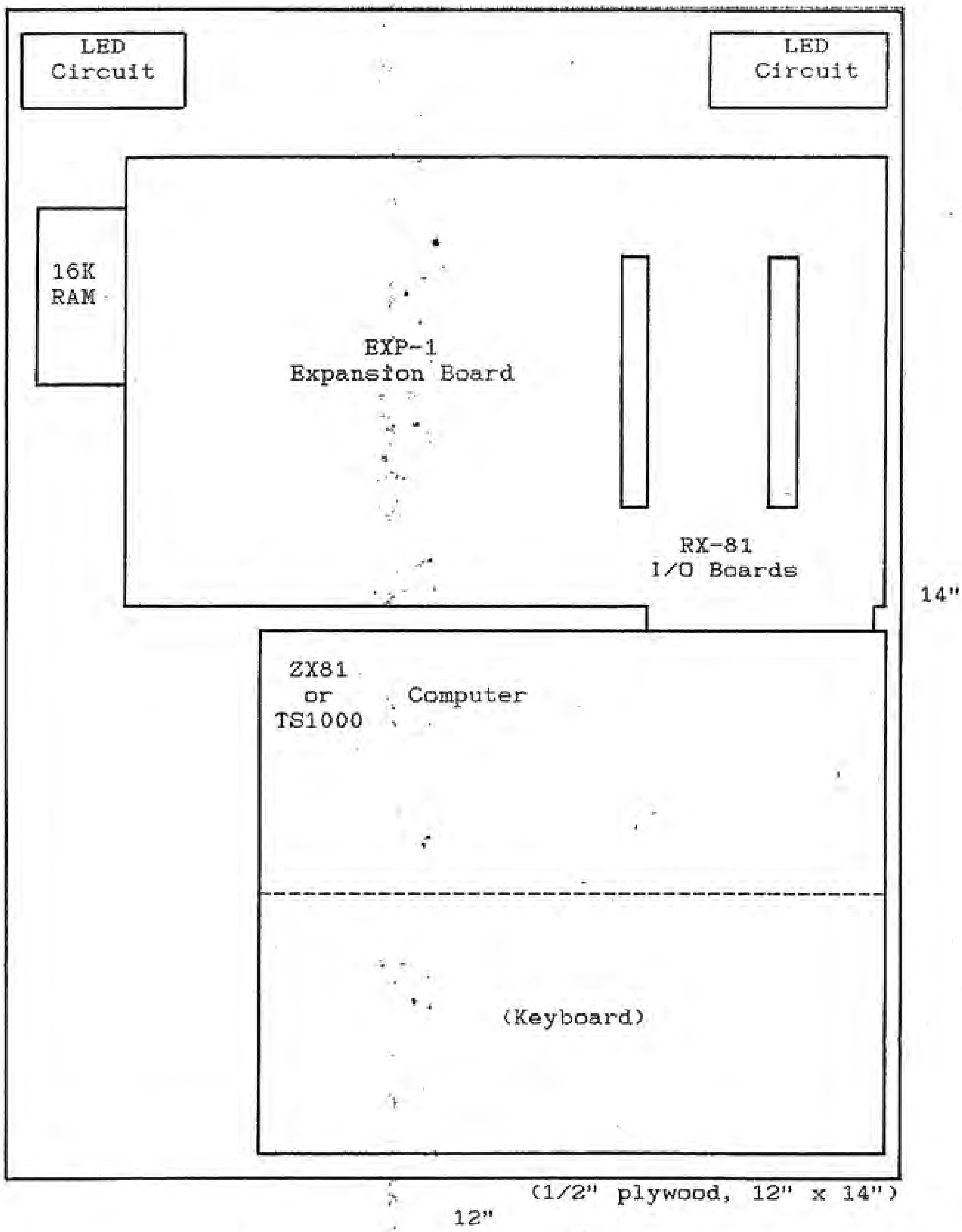
ROBOTICS 1 Mobile Platform
Lower Level Layout



ROBOTICS 1 Mobile Platform
Side View

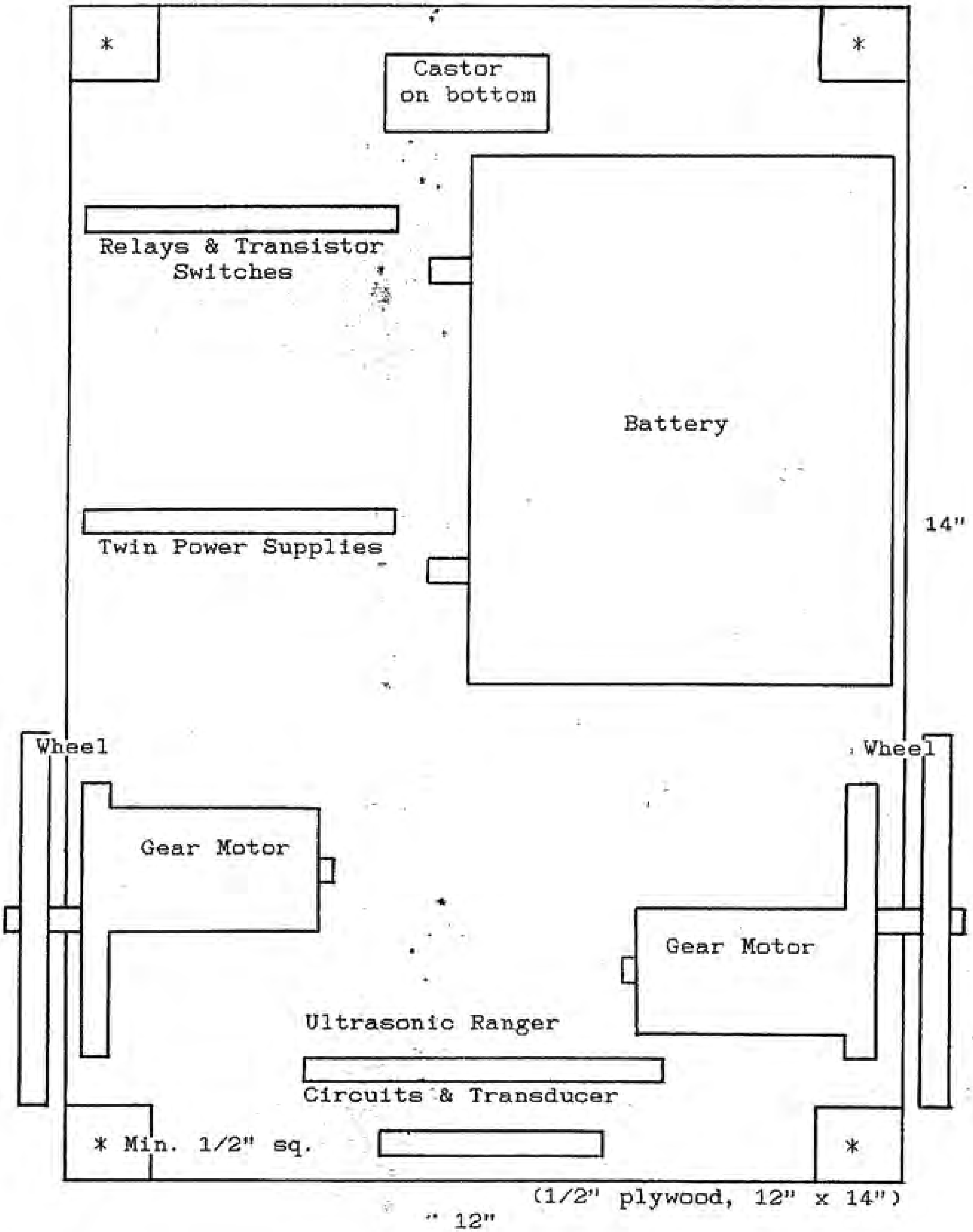


ROBOTICS 2 Mobile Platform
Upper Level Layout



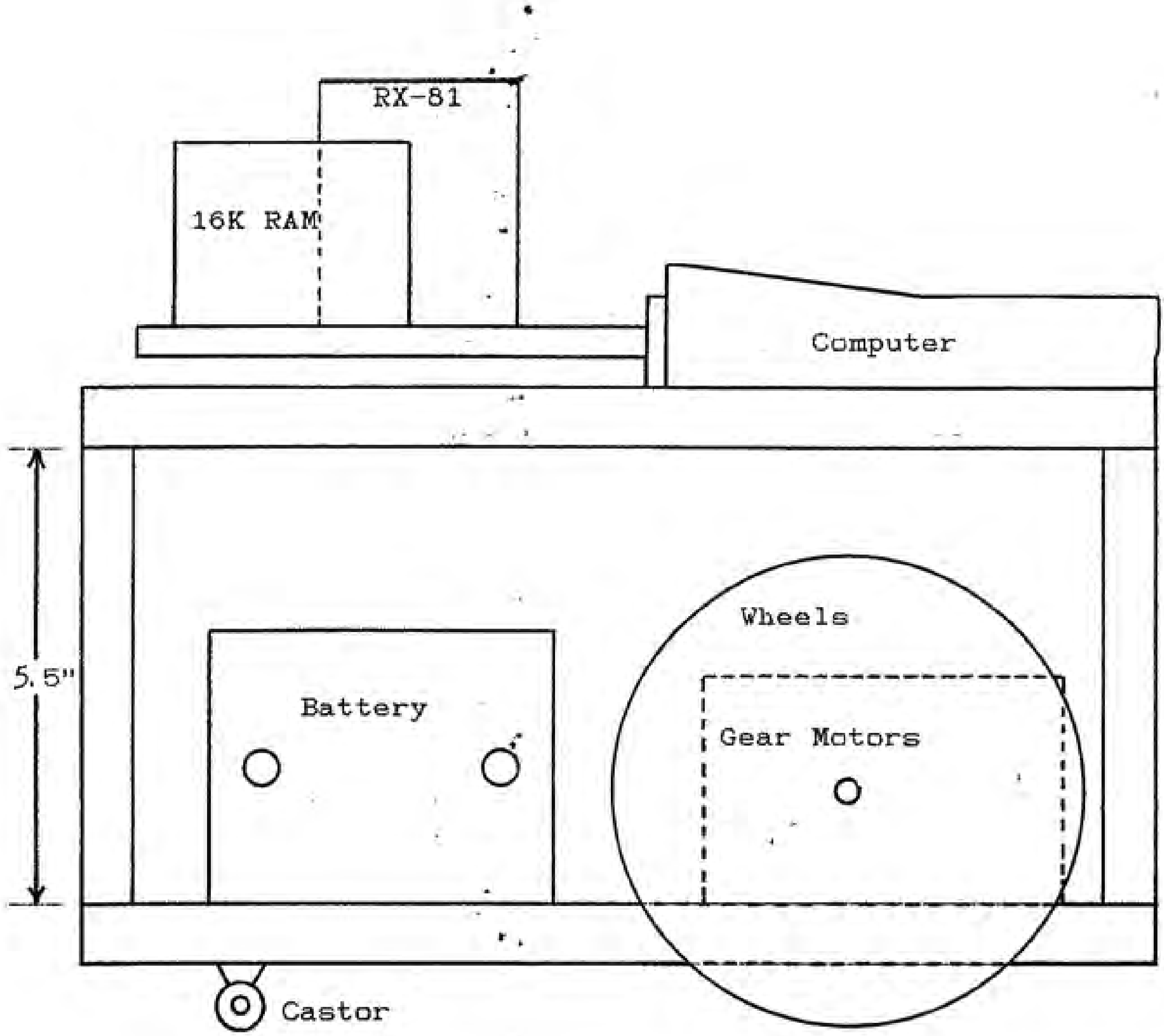
(1/2" plywood, 12" x 14")

ROBOTICS 2 Mobile Platform
Lower Level Layout

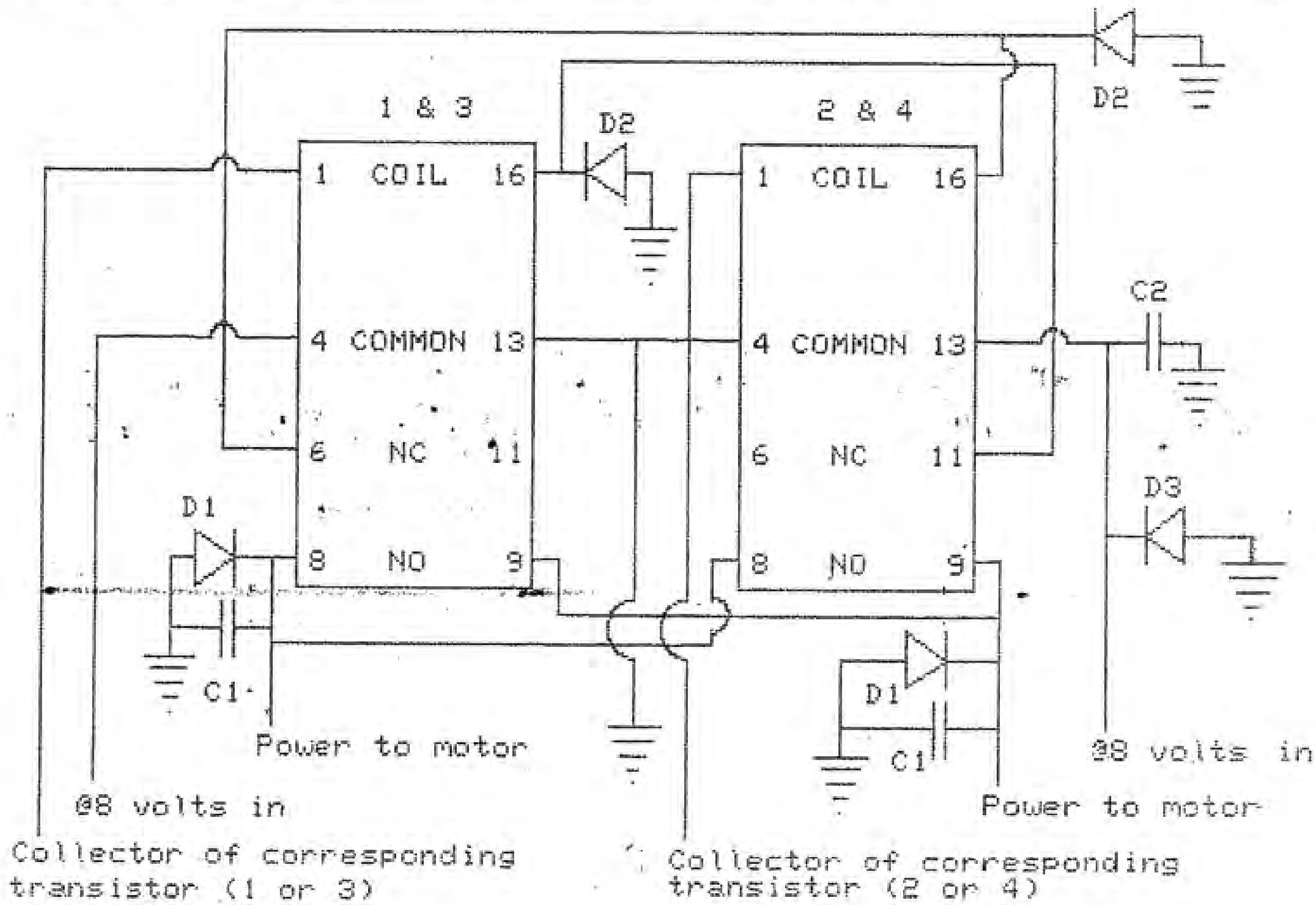


ROBOTICS 2 Mobile Platform

Side View

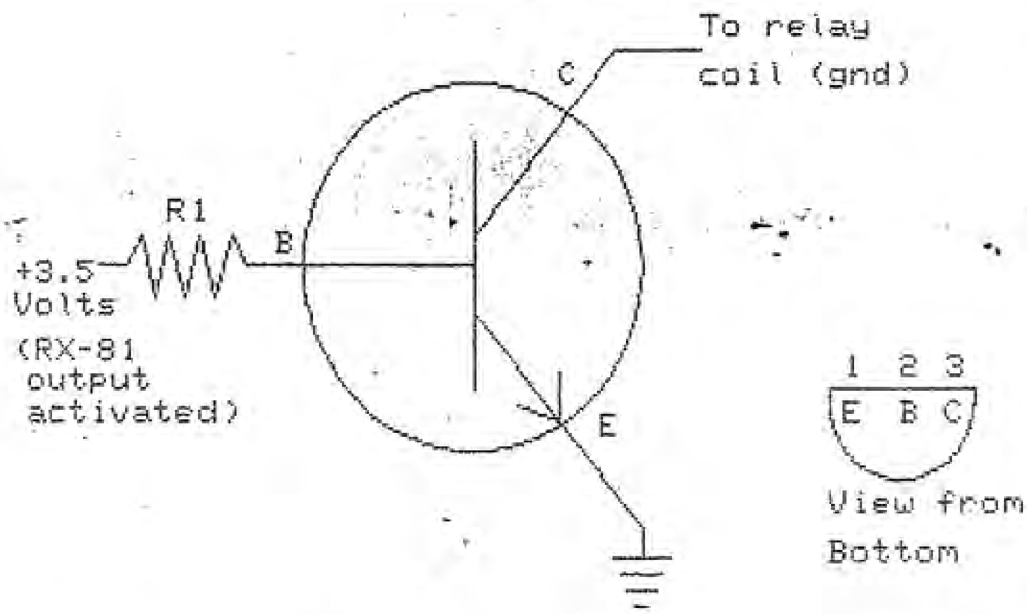


ROBOT 5 VOLT COIL DPDT MOTOR CONTROL RELAY CIRCUIT



TRANSISTOR SWITCH FOR RELAY CIRCUIT

2N2222 Transistor



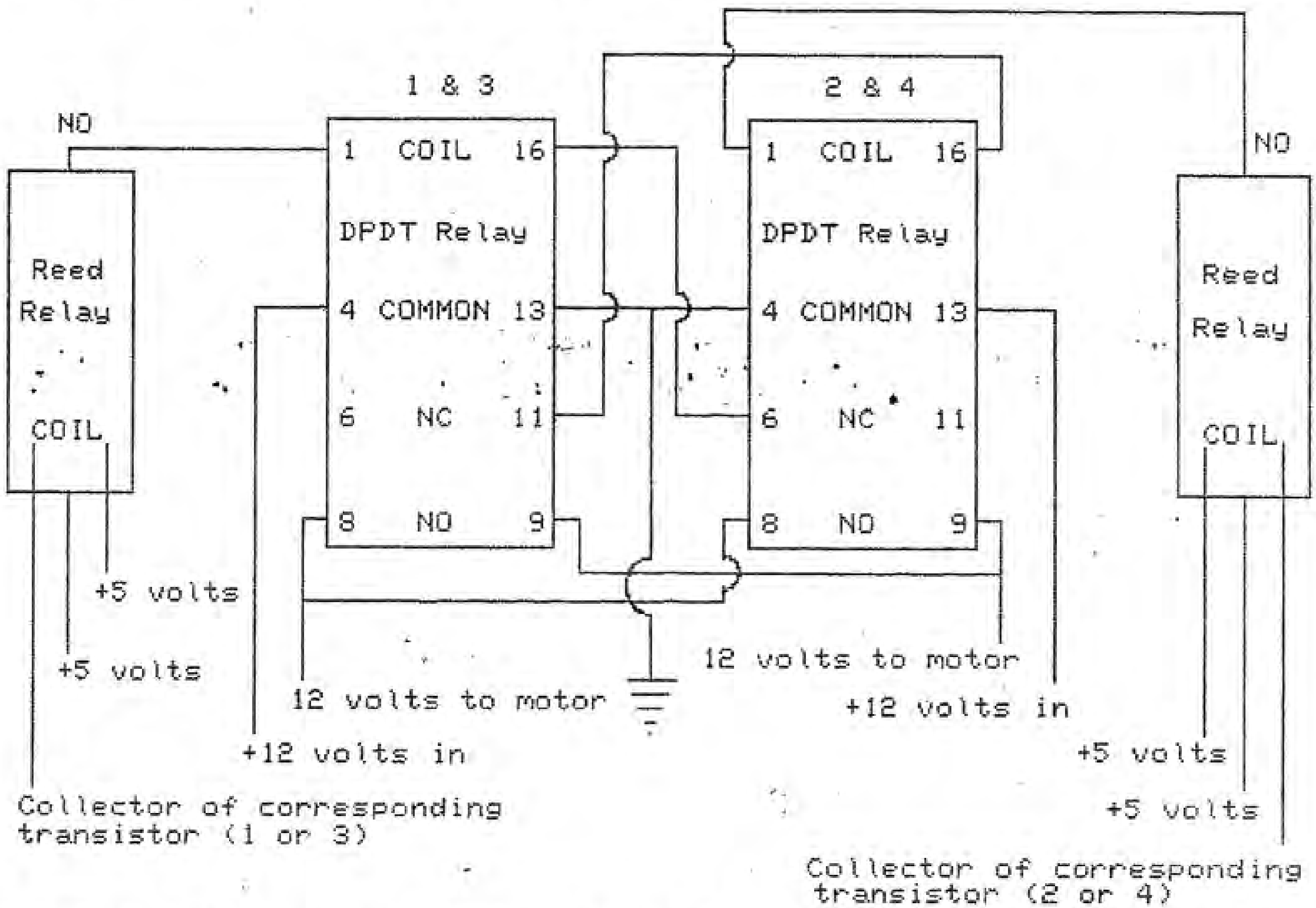
Component list for
Transistor switch
and Relay Circuit

R1	220 OHM 1/4 WATT
D1	1N4002 DIODE
D2	1N4148 DIODE
D3	1N5401 DIODE
C1	.01 MFD 25V CAPACITOR
C2	22 MFD 35V CAPACITOR

PCB CONNECTIONS

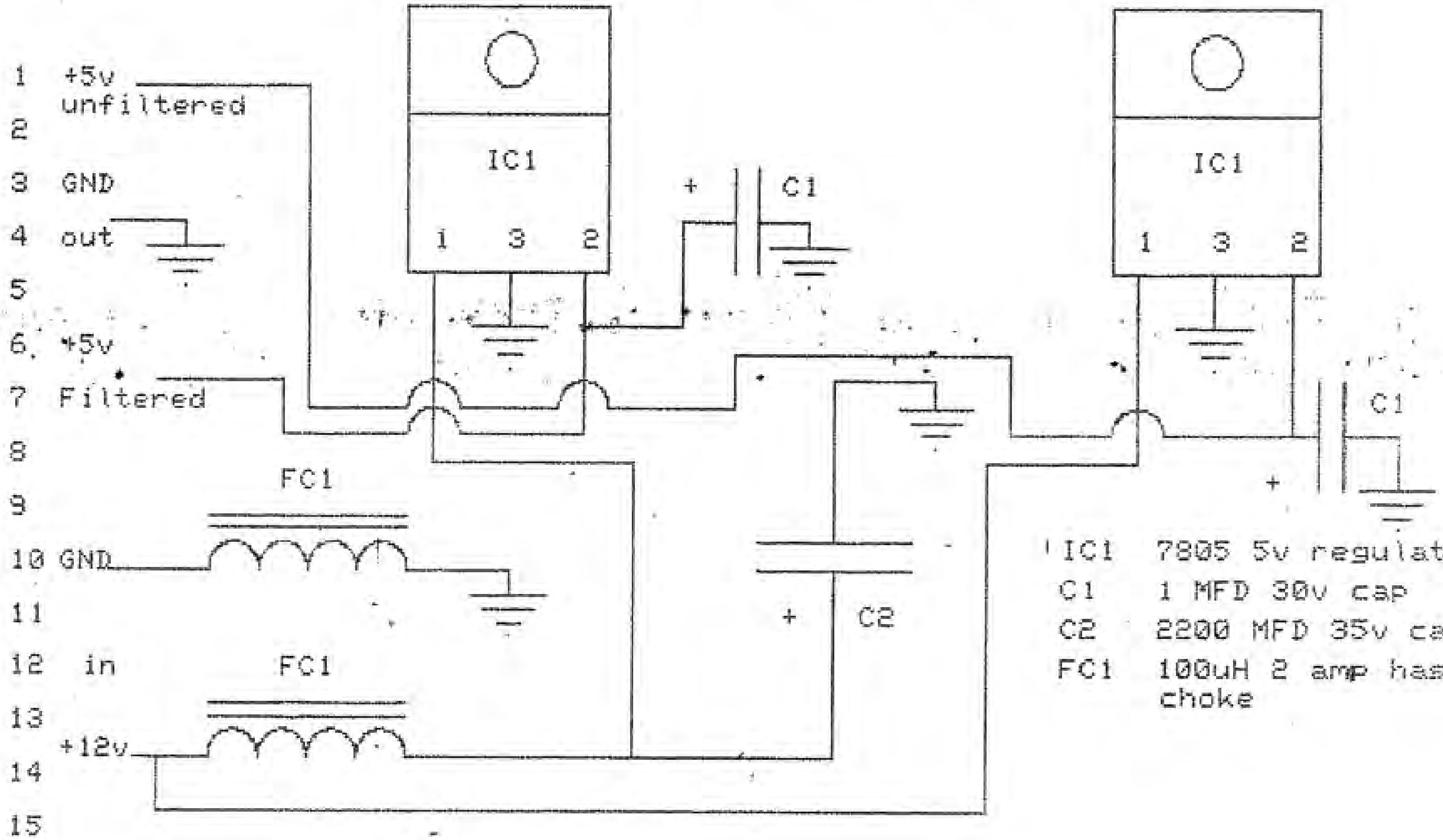
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Motor 1 out		motor 2 out				PWR in		GND					Out 2	Out 1

ROBOT MOTOR CONTROL RELAY CIRCUIT SCHEMATIC



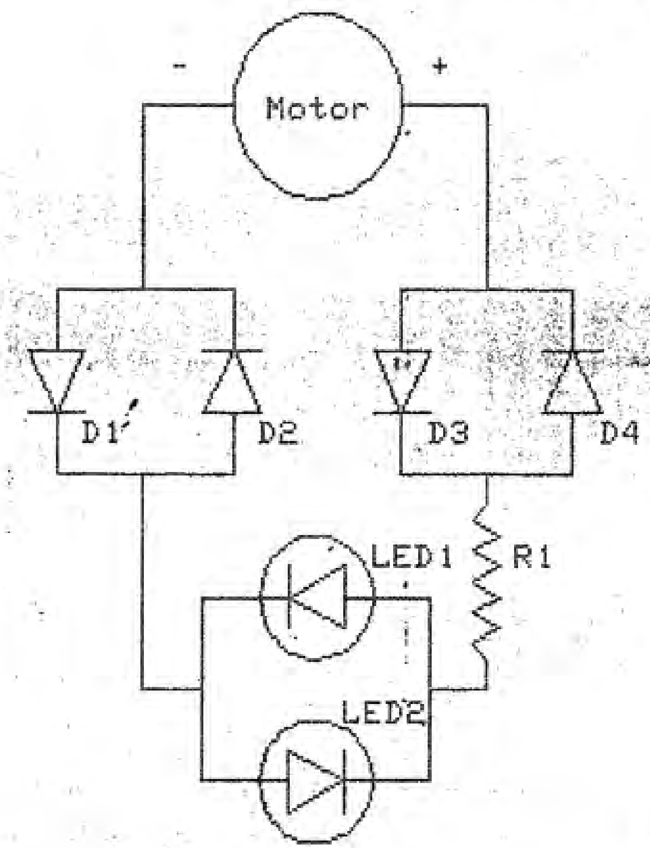
NOTE: This modification of Figs. 5-13,14 uses Digi-Key Z627ND reed relays.

5 VOLT POWER SUPPLIES

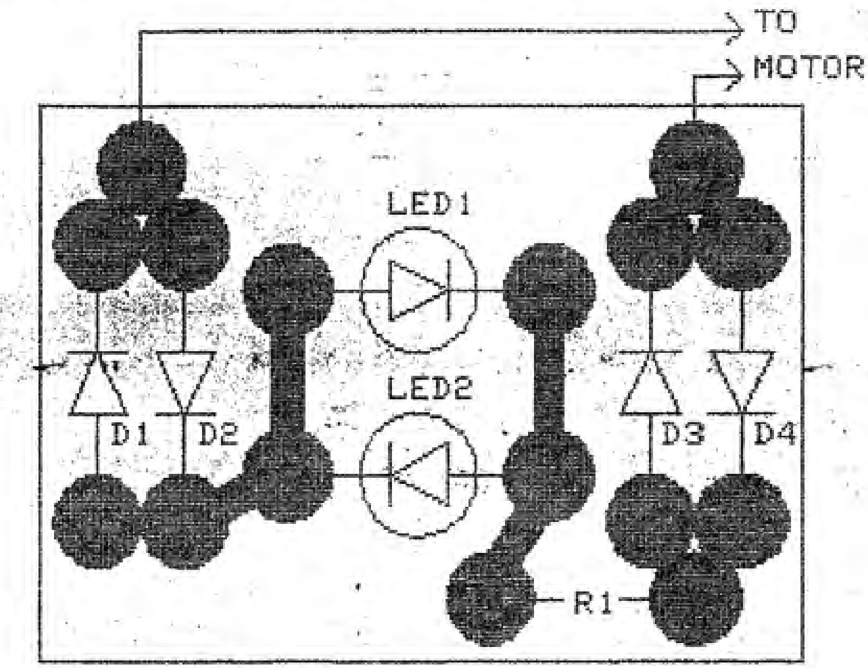


IC1	7805 5v regulator
C1	1 MFD 30v cap
C2	2200 MFD 35v cap
FC1	100uH 2 amp hash choke

Motor Direction Indicator

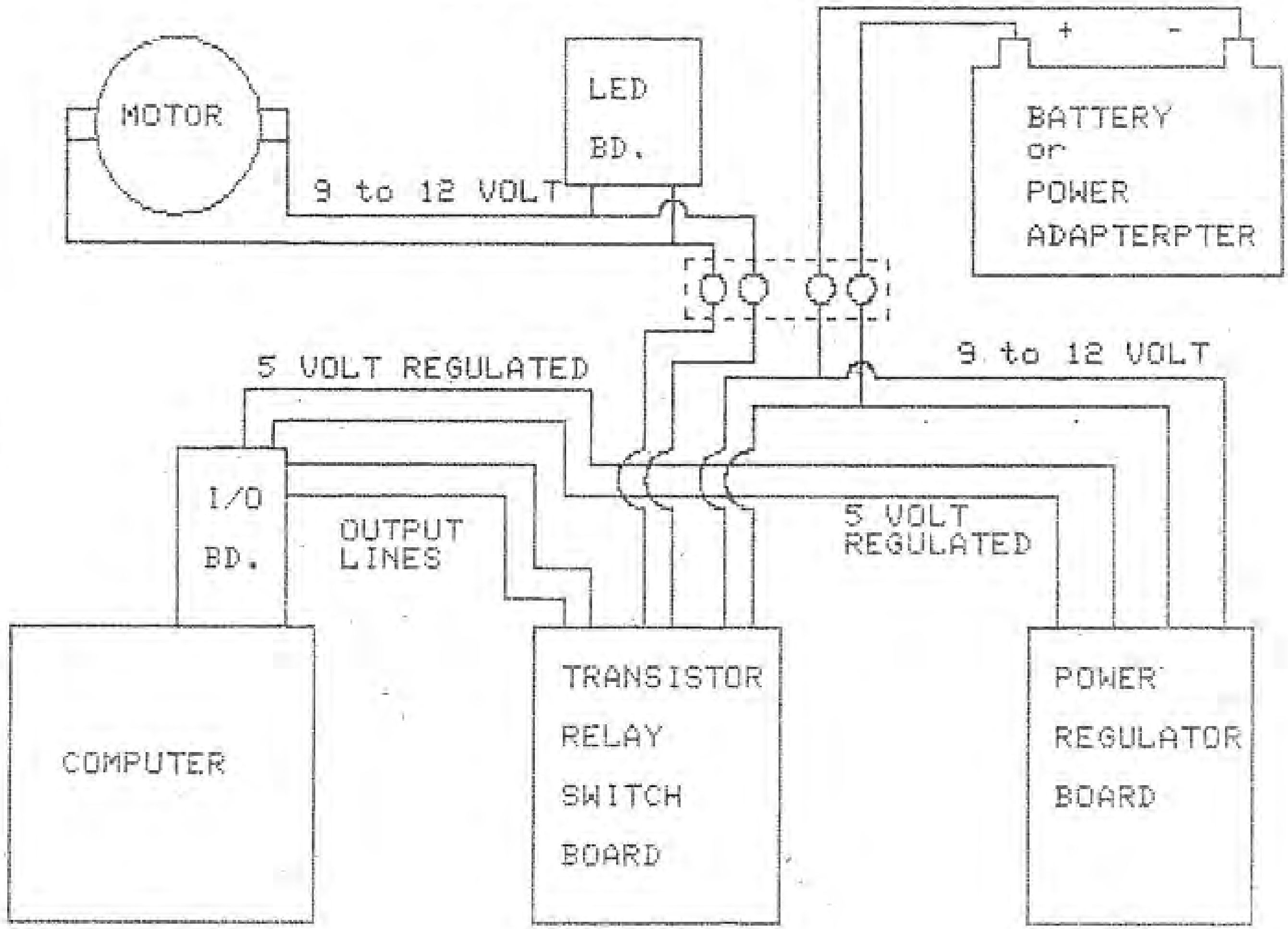


SCHEMATIC



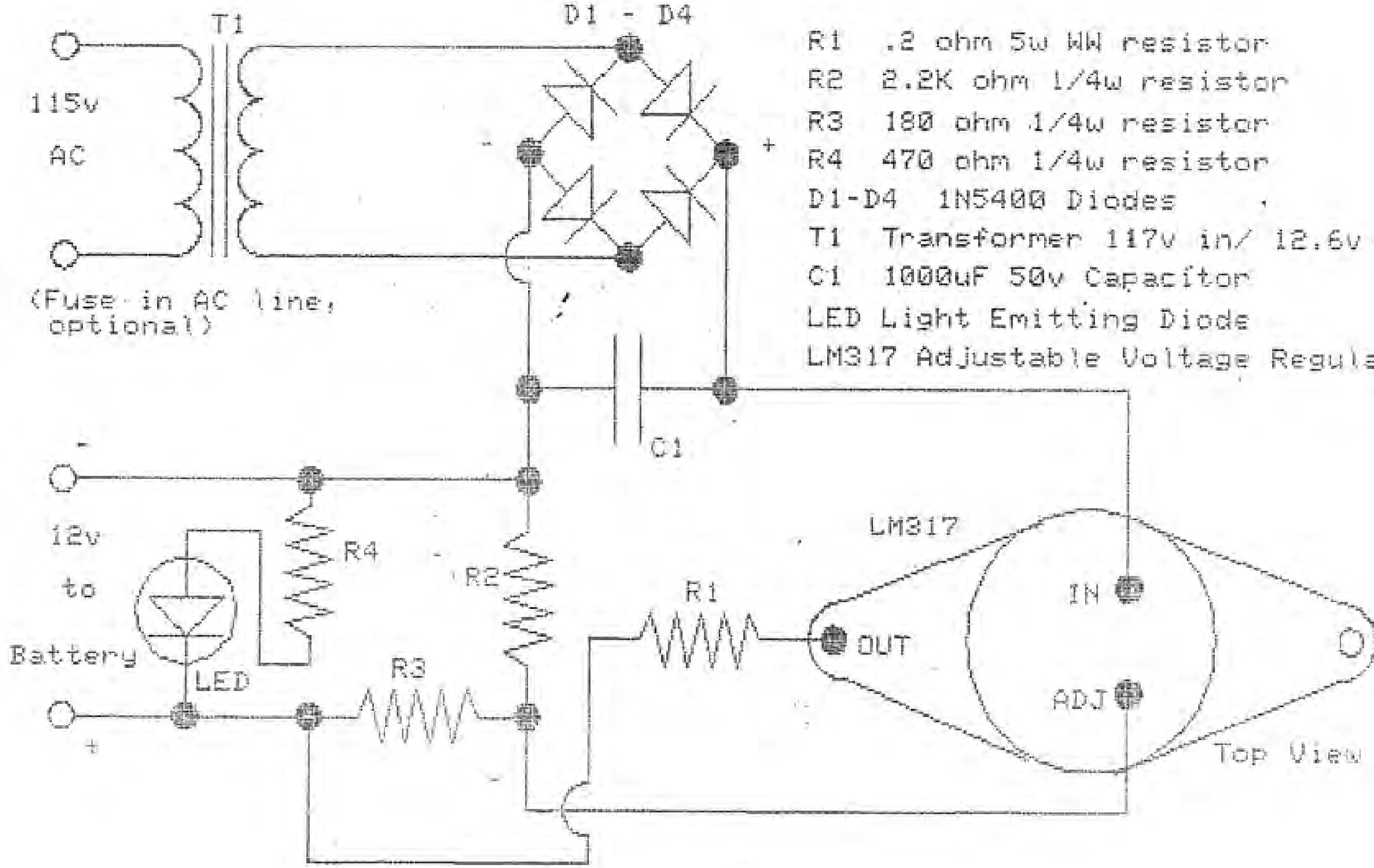
CIRCUIT LAYOUT

ROBOT WIRING

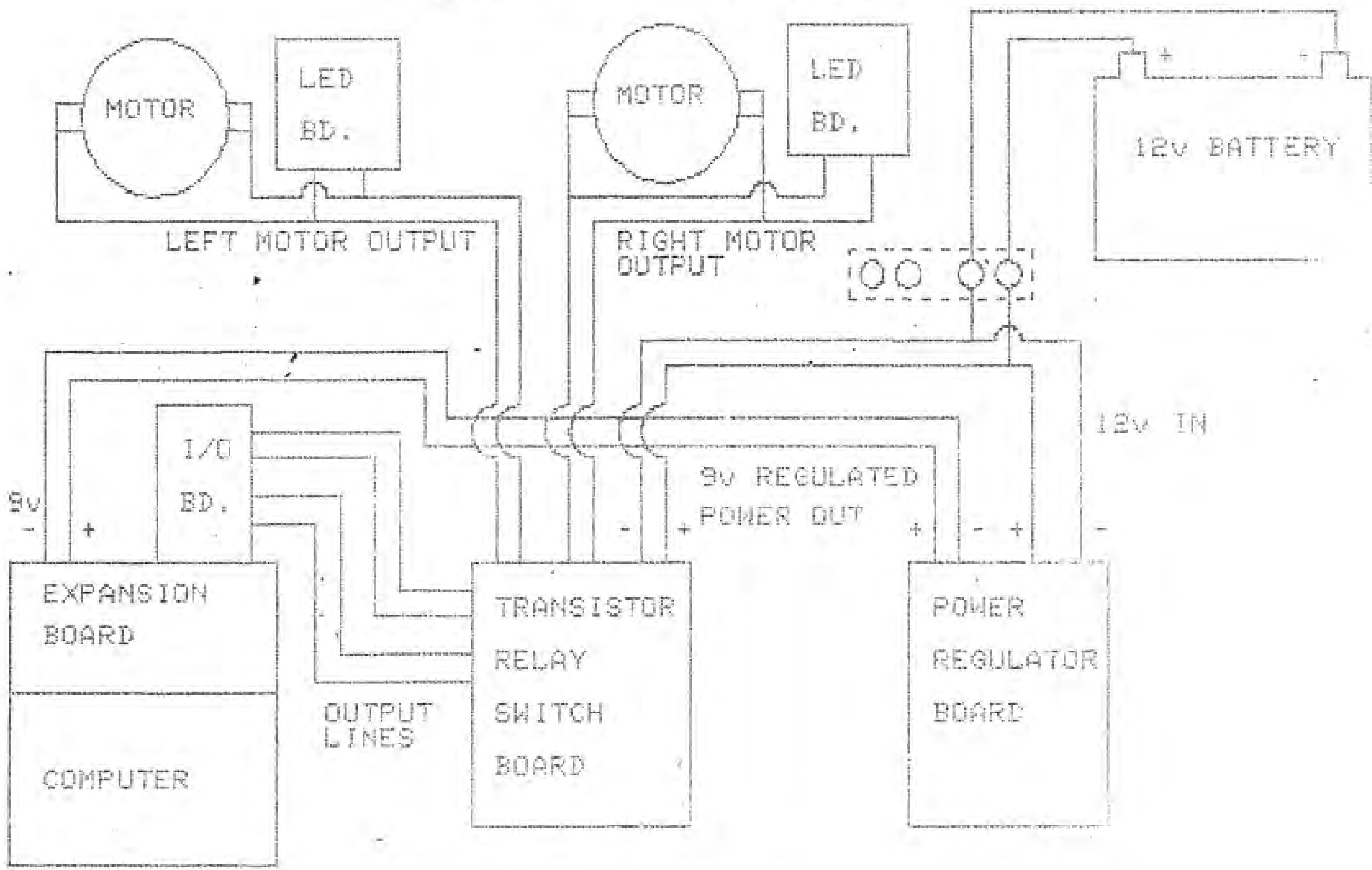


BATTERY CHARGER for 12 VOLT LEAD ACID BATTERIES

R1 .2 ohm 5w WW resistor
R2 2.2K ohm 1/4w resistor
R3 180 ohm 1/4w resistor
R4 470 ohm 1/4w resistor
D1-D4 1N5400 Diodes
T1 Transformer 117v in/ 12.6v Out
C1 1000uF 50v Capacitor
LED Light Emitting Diode
LM317 Adjustable Voltage Regulator



ROBOT WIRING



TWO MOTOR ROBOT CONTROL