INTERFACE™

MICROCOMPUTING FOR HOME AND SMALL BUSINESS VOL. 2, ISSUE 5, APRIL 1977 $1.50

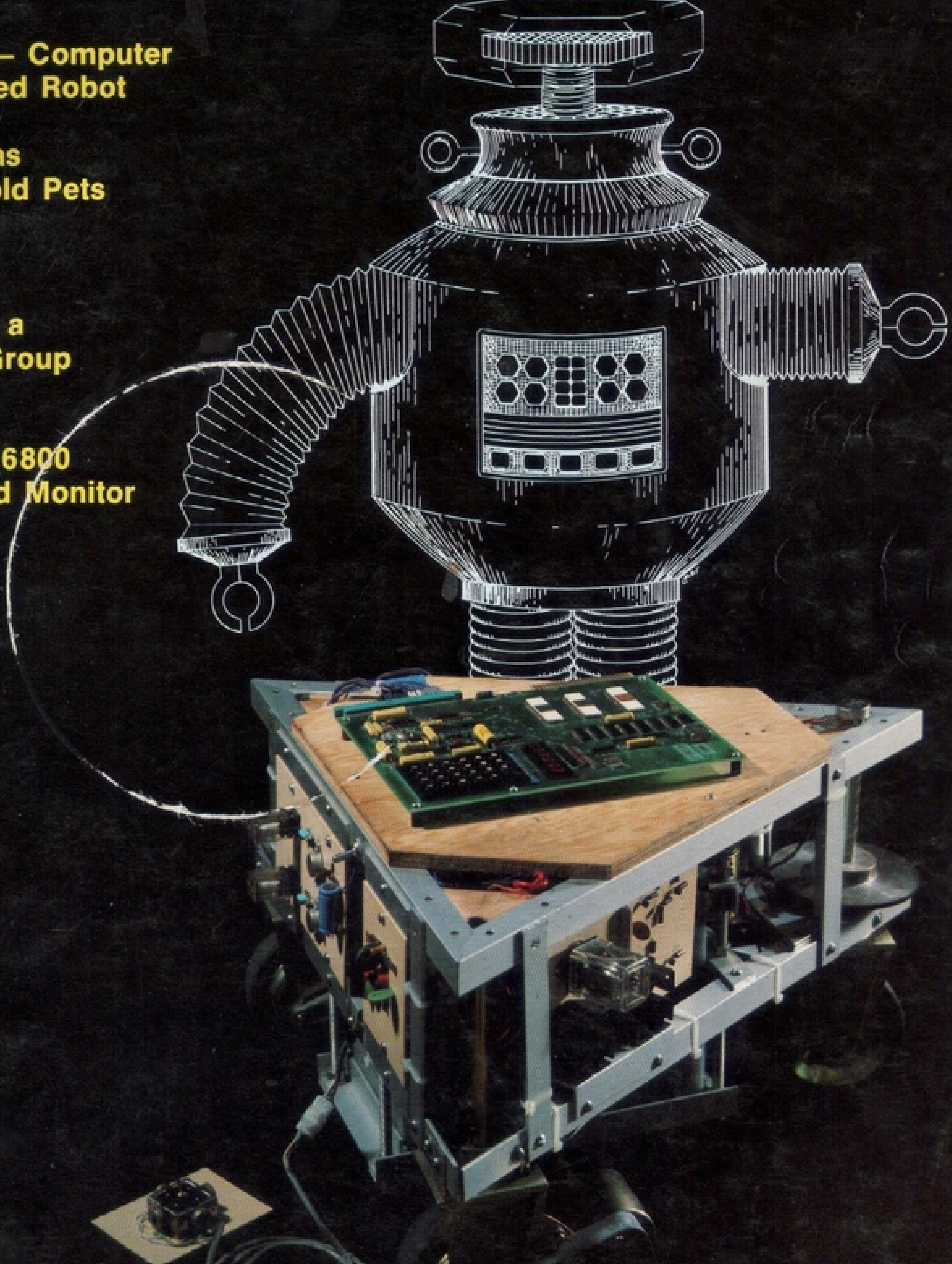
"Mike" — Computer Controlled Robot

Robots as Household Pets

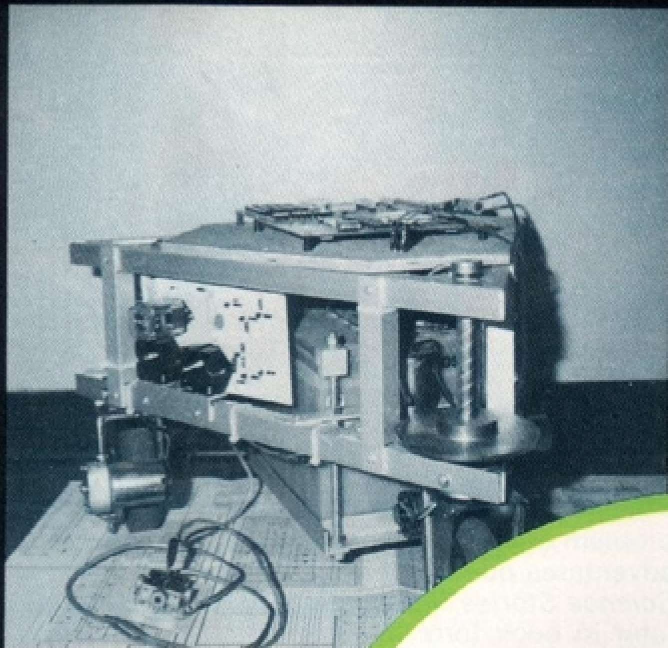
AROK

Building a Digital Group System

EXMON 6800 Extended Monitor System

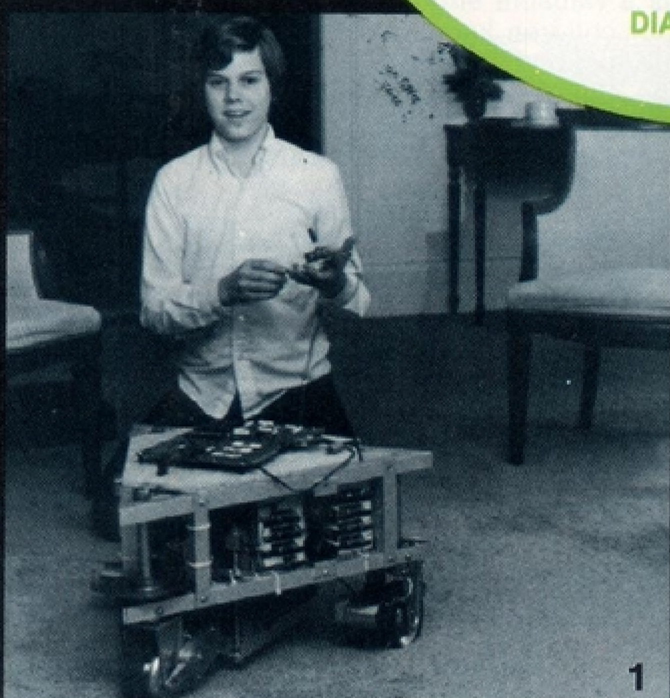


ROBOTS — OUR FUTURE FRIENDS

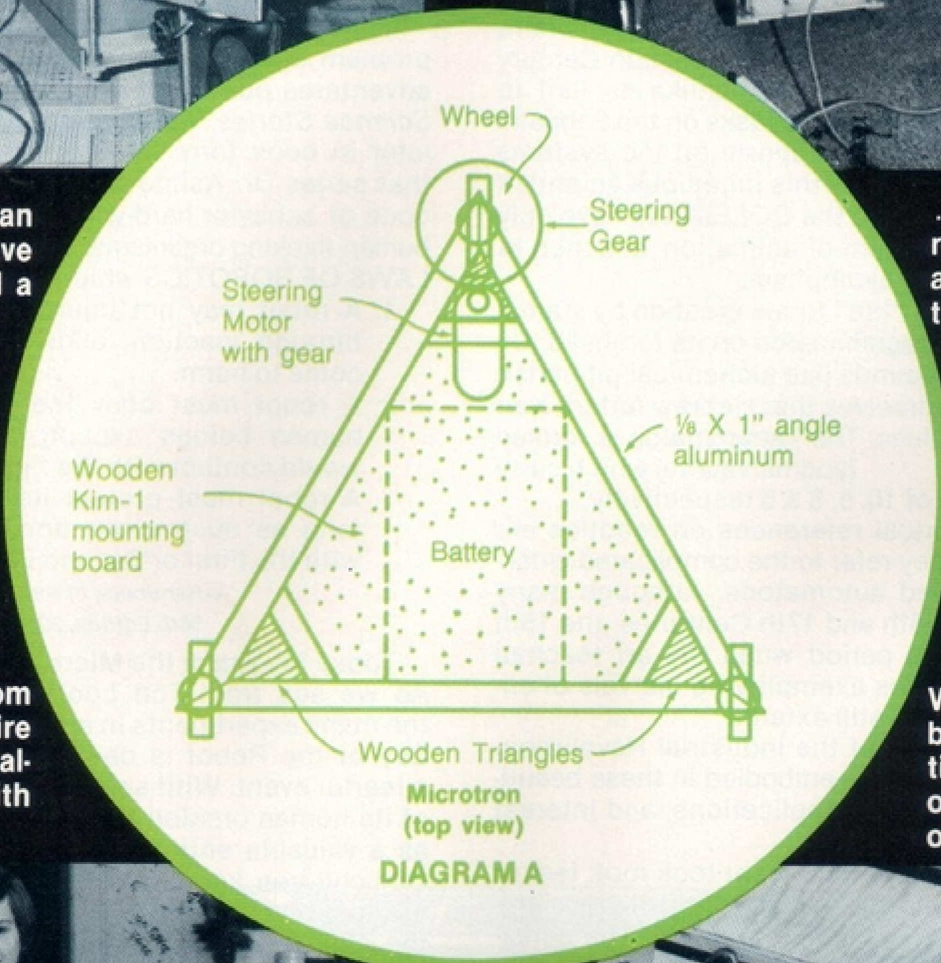


As long as I can remember I have wanted to build a robot.

I don't know from where this desire came, but it has always been with me.

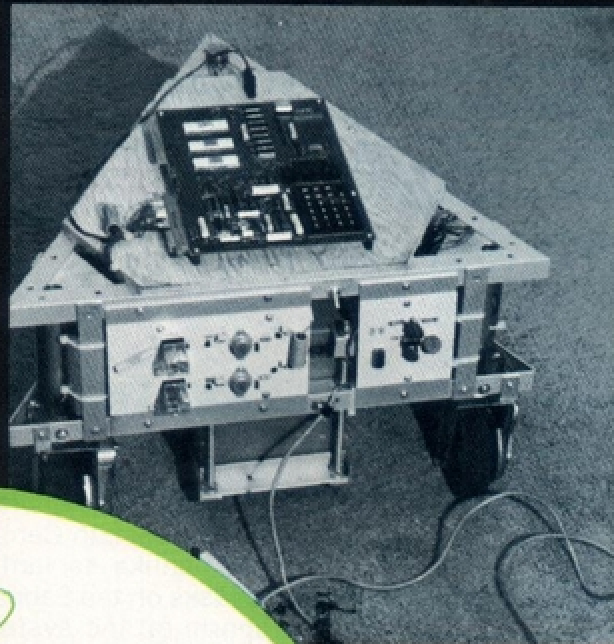


1



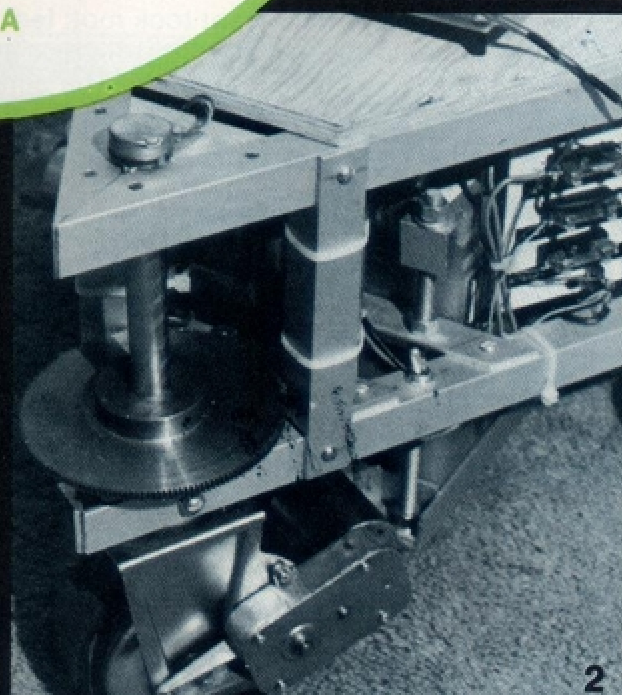
Microtron (top view)

DIAGRAM A



........how the robot should move around. I discarded the idea of legs.

Wheels seemed to be my best alternative. I decided upon a triangle with one steerable wheel.



2

A Computer Controlled Robot

by Tod Loofbourrow

Member of the Amateur Computer Group of New Jersey

Imagine the shock to some innocent passerby as he turns the corner in his morning jog and comes face-to-face with a speeding metal object. The object in question is Microtron, a triangular shaped robot that I built last summer when I was fourteen.[1] It is constructed of 1/8 x 1" angle aluminum and measures 15" in height and 23" per side.[A] Power for the robot is provided by a standard twelve volt car battery, channeled to three motorized wheels.

As long as I can remember I have wanted to build a robot. I don't know from where this desire came, but it has always been with me. I believe my first exposure to robots was at Montreal's Expo '67 where I saw a display of robots. Ever since then I have been fascinated by the idea of building such a device. Several times before I have tried and failed, the best of my efforts producing only a small empty metal box. Then last summer I came across a book about building a robot. Using the book as a guideline, I developed a mental picture of how I wanted the robot to look and then, I began to build.

My first decision to consider was how the robot should move around. I immediately discarded the idea of legs as being far too complicated, and not very fast. For a while thereafter, I was at a loss as to what to use. Wheels seemed to be my best alternative. I considered using electric window motors from automobiles and commercial wheels. The cost, however, would be high. I then came across a completely built motorized wheel in an electronics catalog.[2] Each reversible wheel was 4½" in diameter and 1" wide, and ran on 6 or 12 VDC. Individually, the wheels could carry a load of 200 lbs at walking speed on smooth, level surfaces. Stalled, these wheels had a pulling force of 20 lbs each. The current drain of one wheel was 2 amps with no load, and 8 amps stalled. These wheels appeared to be ideally suited for my concept of the robot, and so I designed it around them.

Requiring 12 volts to run the wheels at the specifications stated, I began to design a frame which could contain a 12 volt car battery. After trying many designs, I finally settled on a triangle both for stability and strength. A triangle also provided a good base for anything I might add, and it needed one less wheel than a rectangle or a square. The front wheel was to be the only steerable wheel, rotated by a separate motor, while the back two wheels were locked in place. A motor was, therefore, required to turn the front wheel. It happened that the same company that advertised the motorized wheels also had geared-down reversible motors. I ordered one of these at the same time as I ordered the motorized wheels.

When these materials arrived, I was able to construct a full-sized diagram of the finished frame and lay the parts out on it. I found that it had to be some 20" long. I immediately set to work building it, and in no time at all had the three wheels mounted in a sturdy triangular frame. It was then that I noticed that the wheels were not firmly anchored to the frame and could wobble. The reason for this problem was that the shafts coming up from the wheels were each set in a circular disk,

With the new microprocessors on the market, an inexpensive and more versatile alternative to hardware was available.

so that the wheels could rotate. The holes in the disks were slightly larger than the outside diameter of the shafts, giving a little play to the shafts. I had mounted the triangular frame on the outsides of the disks so that the wheels could rotate, hence the shafts wobbled too freely. The only solution I could see to this roadblock was to build a second triangle above the first and mount small wooden triangles in its corners. The shafts could then pass through holes in the wood. I built this second triangle and attached it rigidly to the first with pieces of angle aluminum. The wheels were then firmly mounted and, although they could still rotate, they were prevented from wobbling. The two triangles were separated from each other by five inches, and the entire framework was solidly bolted together. This new frame provided good support for anything I might add and was much sturdier than it was before the second triangle was added.

With the wobbling problem solved, I was able to mount my steering motor on the frame. Unfortunately, I ran into another problem. The steering motor, although geared down, was far too fast. It rotated the front wheel much faster than was necessary. This problem was easily solved by mounting a large gear on the shaft of the front wheel and a

1—See photo #1
2—See photo #2
A—See Diagram A

small one on the motor shaft.[3] The ideal gear ratio appeared to be ten to one. In other words, the larger gear had ten times as many prongs as the small one. When I first got the gears the hole in the larger one was only 5/8 of an inch in diameter. This was not a large enough opening for the 3/4" wide shaft to pass through. I, therefore, had to have the hole drilled bigger in a machine shop before I could mount it on the shaft. Once the hole was enlarged, I was able to mount the gear and hook up the steering motor.[A]

Now that the wheels and the steering motor were locked in place in the frame, I needed some way to contain the battery that powered them. My next step was, consequently, to design a battery cage. The first thing I did was lay a cross piece across the lower triangle of the frame 11" from the back. I bolted it in place. Second, I constructed a rectangle out of angle aluminum 12" long and 7" wide. The two shorter sides were turned so that the angle faced outward. The longer sides faced inward so that the battery could rest on them. All of it was held together by four 9" lengths of 3/8" screw rod, bolted in the corners of the rectangle. This screw rod was then bolted to the back of the frame and to the cross piece. It continued to extend above the top of the battery where small pieces of angle aluminum were bolted. The pieces of aluminum were to keep the battery from tipping. I then positioned the entire assembly so that it rode 2" above the ground. Once the battery cage was finished the main ribwork was complete and I could move on to the electronics of the robot.

As I built the framework of the robot, I had, at the same time, been building its circuitry. At first three main circuits were required to allow the robot to move and turn. The first of these circuits was a power supply.[B] This circuit took in +12 volts and by means of a regulator put out +5 volts. The +5 volts could be used for all other circuitry and for the computer which was to be added later. In addition, the power supply contained a series of fuses; it included a 7 amp fuse for the steering motor supply, a 20 amp fuse for the motorized wheels, a 1 amp logic supply fuse, a 7 amp regulator supply fuse, a 1½ amp fuse for the 5 volt supply to the computer, and a 1 amp fuse to supply 12 volts to the computer. The second circuit that I built was the motorized wheel control.[C] This circuit took in +5 volts and +12 volts and released +12 volts to go to the motor. It took in two TTL logic inputs, one for forward and reverse and the other to turn the motor on or off. The third circuit that Mike needed was a steering motor control.[C] Building the steering motor circuit actually consisted of constructing two identical circuits, one for turning right and one for turning left. Input for the steering control was +5 and +12 volts as was the input to the wheel control. The output was +12 volts. Two TTL logic inputs were required in addition, one to steer right and the other to steer left.

3—See photo #2
A—See diagram A
B—See diagram B
C—See diagram C

All three of the circuits described above were bolted to the outside of the triangular frame, with all heat sinks snugly fitted to the aluminum. Altogether, the parts for the circuits cost under one hundred dollars. This fact got me to thinking about how much circuitry I wanted to add, and whether it would be cheaper to use a computer.

As soon as the inverter was added and the program perfected, Mike became a working robot.

With the new microprocessors on the market, an inexpensive and more versatile alternative to hardware was available. After due consideration I decided that a microprocessor would be my best choice, and in so doing opened up a whole new world. I looked around at several of the microprocessors and decided on the Kim-1. There were a number of factors that influenced my decision. One advantage was that the Kim was already built. Also it was lightweight and small, which was crucial since it was to be mounted on the robot. In addition, it was one of the least expensive microprocessors on the market, costing only $245.00. Of all the advantages, the one that was probably most important to me was that the Kim-1 came with its own built-in keyboard and display for loading programs. This fact meant that I didn't have to purchase an additional video or printer to load programs into the Kim.

I made up my mind to order a Kim-1. It took about two weeks to arrive and in the meantime I developed a mounting system for it. I cut a triangle out of ½" plywood 23" on a side. I then cut the ends off of the triangle five inches from each corner. This formed a rough hexagon that screwed down on top of the aluminum triangular frame. When the Kim-1 arrived I was able to mount it on ½" spacers and anchor it into the wood with screws.[4]

The first thing I did once the Kim was mounted was to load in and play a series of games. In spite of my fascination with these programs, I settled down to work. I read over the Kim programming language and with help from my father I began to face the problem of controlling the robot. The first step was to write a program. The best program appeared to be a program loop.* The computer would constantly be going through a series of instructions to monitor any commands given and then execute them. After getting the general program written, I found that three comparators were required to compare the input from the command pots with the actual position or speed of the wheels.[D] I tested the program with the comparators and then modified and perfected it. The final program loop could be subdivided into four parts for explana-

4—See photo #3
*—See flow chart and program
D—See diagram D

tion purposes. The first part was the initialization, which was to get the computer ready to go through the main loop. The second section was the steering control, in which the computer compared the digital reading from the steering command potentiometer with a pot mounted above the shaft of the steerable front wheel. It then made the two numbers equal by rotating the front wheel right or left. A limit switch was written into the program so that the wheels can turn no more than 60 degrees in either direction.

The third part of the program was the speed control. The digital readout from the speed command pot was given to the Kim via a very simple analog to digital circuit. This number caused the computer to turn the motors on a specific number of time units out of ten, and off the remaining number of time units. The fourth part of the program was the speed and direction determination. This part of the program actually occurred before the speed control. The computer determined whether a command from the command pot was for "forward" or for "reverse" and then figured out at what speed. After the speed and direction determination part of the program figured out the number of "motor on" cycles out of ten, and the number of "motor off" cycles, the command was executed by the speed control section of the program.

Having perfected the program, I prepared to test it and, while doing so, I found a peculiarity with the Kim-1. When I hit the RESET button, instead of putting out logic zero on the output lines, as I had expected, the Kim-1 put out all logic 1's. This caused Mike to go into full speed forward, and if he hadn't been on a testing block he would have crashed headlong into the wall. This incident prompted me to add an inverter to Mike's circuitry so that the experience would not be repeated.[B] As soon as the inverter was added and the program perfected, Mike became a working robot. I wrote a few short programs so that when I was not controlling Mike, he would be moving about in a preplanned pattern. He moved in a clover leaf pattern for one of these programs, and for the other, executed a simple back-and-forth pattern, turning slightly left everytime it goes forward. These programs were used primarily for demonstration purposes and were not usually stored in memory.

After getting Mike working I controlled him at first with two potentiometers and a forward/reverse switch. One pot controlled the speed of the robot, while the other turned him right or left. Later I replaced this arrangement with a joystick.[5] All together, Mike had progressed from a whimsical idea in my head to a complex computer-controlled robot in less than three months.

Although Mike is a working operator-controlled robot, he is far more complete. He represents only the beginning of a complex *independent* unit. The next phase of construction will be to add some type of sensors to Mike's outer hull. First I will be wanting to add a framework over the triangle. I believe it will be in the shape of a nonequilateral octagon. On each side some type of sensor will

B—See diagram c
5—See photo #3

be mounted, to detect wall or objects in Mike's path. I have experimented with the possibility of using proximity switches, but I have found them to be ineffective against wood and other non-conducting obstructions. Ultrasonics and infrared light offer two unrealistic possibilities because I do not have the knowledge nor the funds to employ either one. My most practical possibility is the use of bumpers. These bumpers could consist of metal plates mounted on buttons and springs. They could detect when Mike bumps into an object and make him respond according to which bumper was triggered. Any thoughts or suggestions about any other type of sensors would be more than welcome.

Besides sensors, there are many other additions I plan for Mike in the future. For one thing, when his battery gets weak, he will hunt out his charger and plug himself into it to recharge his battery. There is also the possibility that I will build up from his triangular base to give Mike two arms and a head. I may give him a voice or the ability to respond to certain voice commands. The possibilities are limited only by my imagination. Who knows? The next person you see walking down the street may, in fact, be something else. . . .

INITIALIZATION PROGRAM

0290	A9 1F	INIT	LDA #$1F	Set DDRA PA0-4 = OUTPUT
0292	8D 0117		STA $1701	PA 5-7 INPUT
0295	A9 00		LDA #$00	SET DDRB PB0-7 INPUT
0297	8D 0317		STA $1703	
029A	A9 00		LDA #$00	LOAD A WITH 0
0296	85 02		STA $02	STORE IN SPEED COUNT
029E	85 03		STA $03	STORE IN 'ON' TIME
02A0	A9 0A		LDA #$ 0A	SET OFF TIME
02A2	85 04		STA $04	STORE IN 'OFF' TIME
02A4	A9 24		LDA #$24	SET STEERING TO CENTER
02A6	85 01		STA 01	
02A8	A9 FF		LDA FF	
22AA	8D 0017		STA 1700	SET PA TO ALL 1's.
02AD	A9 00		LDA #$00	SET BUMPER CYCLE
02AF	85 06		STA $06 =	COUNT TO 0
02B1	4C 35 02		JMP SCAN 3	START SCAN

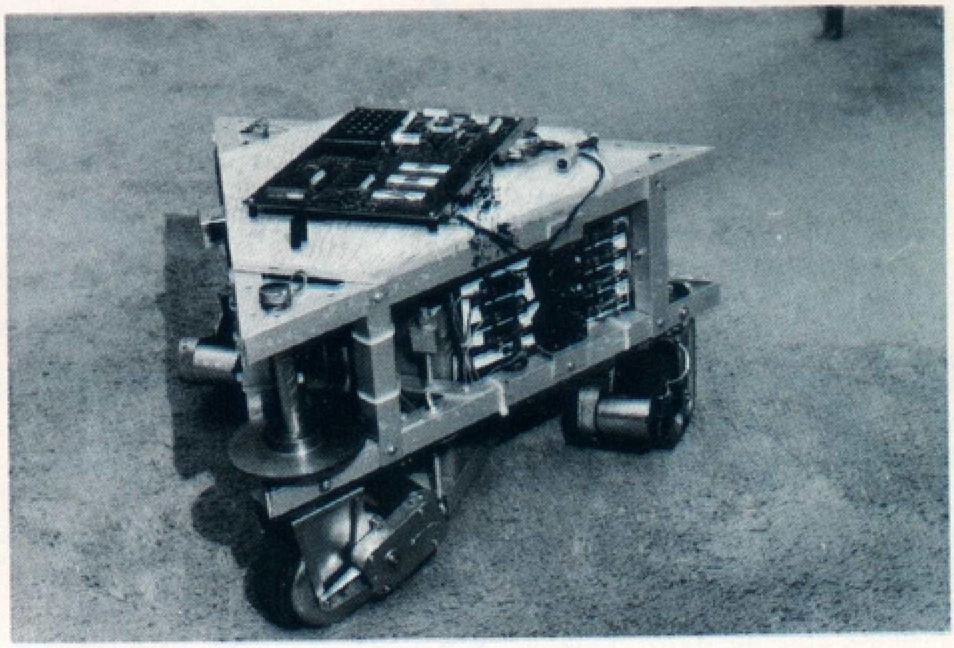
THIS ROUTINE HAS A MAIN PROGRAM WHICH IS PART OF THE MAIN SCAN ROUTINE. IT CONTROLS THE SPEED OF THE MAIN MOTORS BASED ON THE NUMBER OF TIME THROUGH THE SCAN ROUTINE.

SPEED CONTROL PROGRAM

02C0	A5 02	SCAN	LDA $02	GET SPEED COUNT
02C2	0A		ASL	SHIFT BIT 7 TO CARY (ON-OFF INDICATOR)
02C3	F0 05		BEQ CHNG	Branch if Speed Count = 0
02C5	C6 02		DEC $02	DECRIMENT SPEED COUNT
02C7	4C 0003		JMP SCAN1	Jump to Scan Continuation
02CA	B0 13	CHNG	BCS 0n	Branch if Carry Set ('ON' Cycle)
02CC	A5 03	OFF	LDA, $03	Get Value of 'ON' Time
02CE	F0 0F		BEQ ON	BRANCH IF VALUE = 0
02D0	09 80		ORA #$80	Add the "ON" Cycle Indicator
02D2	85 02		STA $02	PUT IN SPEED COUNT
02D4	AD 0017		LDA $1700	READ PA
02D7	29 F7		AND #$F7	SET PA3 TO '0' (Speed Control)
02D9	8D 0017		STA $1700	SEND PA3 = 0 (Speed Control)
02DC	4C 0003		JMP SCAN1	Jump to Scan Continuation
*CARRY SET—END OF ON CYCLE				
02DF	A5 04	ON	LDA $04	GET VALUE OF OFF TIME
02E1	F0 E9		BEQ OFF	BRANCH IF VALUE = 0
02E3	85 02		STA $02	STORE IN SPEED COUNT
02E5	AD 0017		LDA 1700	READ PA
02E8	09 08		ORA #$08	SET PA3 TO '1' (SPEED CONTROL = OFF)
02EA	8D 0017		STA 1700	SEND PA3 = 1
02ED	4C 0003		JMP SCAN1	Jump to Scan Continuation

MANUAL CONTROL TABLE

0010	OF 0A	TABLE	FAST	
0012	12 55		MEDIUM FAST	
0014	17 73		MEDIUM	REV
0016	1D 82		MEDIUM SLOW	
0018	23 91		SLOW	
001A	26 A0		OFF	
001C	29 91		SLOW	
001E	2B 82		MED SLOW	
0020	2D 73		MED	FORWARD
0024	30 0A		FAST	
0026				



PROGRAM FOR JOYSTICK CONTROL
ANALOG-TO-DIGITAL 2 INPUTS RELAY OUTPUTS
START PROGRAM AT LOCATION 0290 HEX

Address	CODE	LABEL	MNEMONIC	COMMENTS
0200	20 IF 02	ATODI	JSR START	
0203	2E 00 17	L00PI	BIT 1700	TEST PA FOR INPUT
0206	10 FB		BPL L00PI	CHECK BIT 7 FOR '1'
0208	AD 00 17	STOP	LDA $1700	READ PA
020B	29 FE		AND #$FE	SET BIT 0 TO 0
020P	8D 00 17		STA $1700	OUTPUT '0' ON PA 0
0210	AD 04 17		LDA $1704	GET TIMER COUNT
0213	60		RTS	
0214	20 1F 02	ATOD2	JSR START	
0217	2C 00 17	LOOP2	BIT 1700	TEST PA FOR INPUT
021A	50 FB		BVC L00P2	CHECK BIT 6 FOR '1'
021C	4C 08 02		JMP STOP	
021F	A9 42	START	LDA #$42	STARTING COUNT
0221	8D 05 17		STA $1705	STOP COUNT IN -8 TIMER
0224	AD 00 17		LDA $1700	READ PA
0227	09 01		ORA #$01	SET PA0 TO '1'
0229	8D 00 17		STA $1700	OUTPUT A'1' ON PA0
022E	60		RTS	
022D	A9 1F	MAIN	LDA #$1F	Set DDR PA0-4 OUTPUT PA5-7 INPUT INITIALRE
022F	8D 01 17		STA $1701	
0232	4C 58 02		JMP FQ	
0235	20 00 02	Scan3	JSR ATOP1	GET INPUT 1(Steering Pot)
0238	85 00		STA #$00	Store Result in Save Area
023A	20 7D 02		JSR WAIT	Delay for Capacitor to Discarge
023D	4C 5C 03		JMP PATCH1	Go To Steering Limit Patch
0240	85 01		STA #$01	Store Result in Save Area
0242	20 7D 02		JSR WAIT	Delay for Capacitor to Discharge
0245	A5 00		LDA #$00	PUT LOC 0 IN ALL
0247	E5 01		SBC $01	Sub #IN LOC 1
0249	F0 0F		BEQ EQ	
0248	10 18		BPL PL	
024D	AD 00 17		LDA 1700	READ PA
0250	09 04		ORA #$04	Set PA2 TO '1' (Steer Left)
0252	29 FD		AND #$FD	Set PA1 to '0' (No Steer Right)
0254	8D 00 17		STA 1700	OUTPUT PA1 = '1' PA2 = '0'
0257	4C 72 02		JMP CONT	CONTINUE SCAN ROUTINE
025A	AD 00 17	EQ	LDA $1700	READ PA
025D	09 06		ORA #$06	Set PA1, PA2, to '1' (Left, Right Stay)
025F	8D 00 17		STA $1700	SEND PA1, PA2 = 0
0262	4C 72 02		JMP CONT	CONTINUE SCAN ROUTINE
0265	AD 00 17	PL	LDA 1700	READ PA
0268	09 02		ORA #$02	SET PA1 to '1' (STEER RIGHT)
026A	29 FB		AND #$FB	SET PA2 TO '0' (No Steer left)
026C	8D 00 17		STA 1700	OUTPUT PA2 = 1, PA1 = 0
026F	AC 72 02		JMP CONT	CONTINUE SCAN ROUTINE
0272	4C C0 02	CONT	JMP SCAN	NORMAL LOOP BACK
0275				
			FOR TESTING ONLY	
0272	A5 01	CONT	LDA 01	FOR TESTING ONLY
0274	85 FA		STA FA	
0276	A5 00		LDA 00	
0278	85 FB		STA FB	FB
027A	4C C0 02		JMP SCAN	
027D	A9 28	WAIT	LDA #$28	Starting Timer Count (Check)
027F	8D 05 17		STA $1705	STORE IN -8 TIMER
0282	A9 00		LDA #$00	
0284	CD 0417	Check	CMP 1704	
0287	D0 FB		8NE Check	
0289	60		RTS	

MANUAL CONTROL PROGRAM

```
Address  Code      Label    Mnemonic         Comments

02F0     20 1F 02  A to D3  JSR START        START A/D 3 COUNT
02F3     A9 20     LOOP3    LDA #$20         SET MASX FOR PA5
02FS     2D 00 17           AND $1700
02F8     F0 F9              BEQ L00P3        BRANCH IF PAS = 1
02FA     20 08 02           JSR STOP         Stop A/D 3 Conversion A = Value
02FD     60                 RTS
02FE
0300     20 F0 02  Scan1    JSR A TO D3      ON RETURN A = VALUE
0303     85 F9              STA $F9          PUT BYTE IN DISPLAY AREA
0305     C9 26              CMP #$26         IF VALUE >= 26 THEN
0307     10 2E              BPL REV          GO TO FORWARD ROUTINE
0309     AD 00 17           LDA $1700
030C     09 10              ORA #$10
030E     8D 00 17           STA $1700        SET PA4 = 1 (REVERSE)
0311     A5 F9     SCTAB    LDA $F9          Retrieve Byte From A/D 3 Conv.
0313     A2 00              LDX #$00         SET INDEX REG = 0
0315     D5 10 EA  Again    CMP Table, X     Get Table "Speed" Value
0318     30 06              BMI FOUND        Branch if Hit in Table
031A     E8                 INX
031B     E8                 INX              INCREMENT TO NEXT VALUE
031C     E0 12              CPX #$12         CHECK FOR END OF TABLE
031E     D0 F5              BNE AGAIN
0320     E8        Found    INX              Move Pointer to "Action" Value
0321     B5 10 EA           LDA TABLE, X     Get Action Value Into A
0324     4A                 LSR              SHIFT LEFT 4 BITS
0325     4A                 LSR              TO RIGHT—FILL IN
0326     4A                 LSR              ON LEFT WITH 0's
0327     4A                 LSR
0328     85 04              STA $OFF         PUT VALUE IN OFF TIME
032A     B5 10 EA           LDA TABLE, X     GET ACTION VALUE AGAIN
032D     29 0F              AND #$0F         MASK OUT LEFT 4 BITS
032F     85 03              STA $ON          STORE IN "ON" TIME
0331     20 7D 02           JSR WAIT         Delay for Capacitor Discharge
0334     4C 35 02           JMP SCAN3
0337     AD 00 17  REV      LDA $1700
033A     29 EF              AND #$EF
033C     8D 00 17           STA $1700        Set PA4 = 0 (Forward)
033F     4C 11 03           JMP SCTAB        NOW SEARCH TABLE
```

STEERING TABLE PATCH

```
035C     20 14 02  Patch1   JSR ATOD2        Get Count (Patch from 923D
035F     A2 00              LDX #$00         SET INDEX = 0
0361     DD 73 03  Again2   CMP TABLE 2, X   GET STEERING VALUE
0364     30 06              BMIFOUND2        BRANCH IF HIT IN TABLE
0366     E8                 INX
0367     E8                 INX              Increment to next Value
0368     E0 14              CPX #$14         CHECK FOR END
036A     D0 F5              BNE AGAIN2
036C     E8        Found2   INX              Move Pointer to Steering Val
036D     BD 73 03           LDA Table2,X     Get Steering Value into A
0370     4C 40 02           JMP 0240         RETURN FROM PATCH
0373     0E 26     Table2   0E 26            Table of Steering Values
0375     13 26              13 26            Left Byte is From
0377     18 27              18 27            A → D Conversion Right
0379     1D 28              1D 28            Byte is Value Used
037B     22 29              22 29            To Determine Steering
037D     27 2A              27 2A            Angle.
037F     2A 2B              2A 2B
0381     2C 2C              2C 2C
0383     2E 2D              2E 2D
0385     3Q 2E              3Q 2E
0387     31 2F              21 2F
```

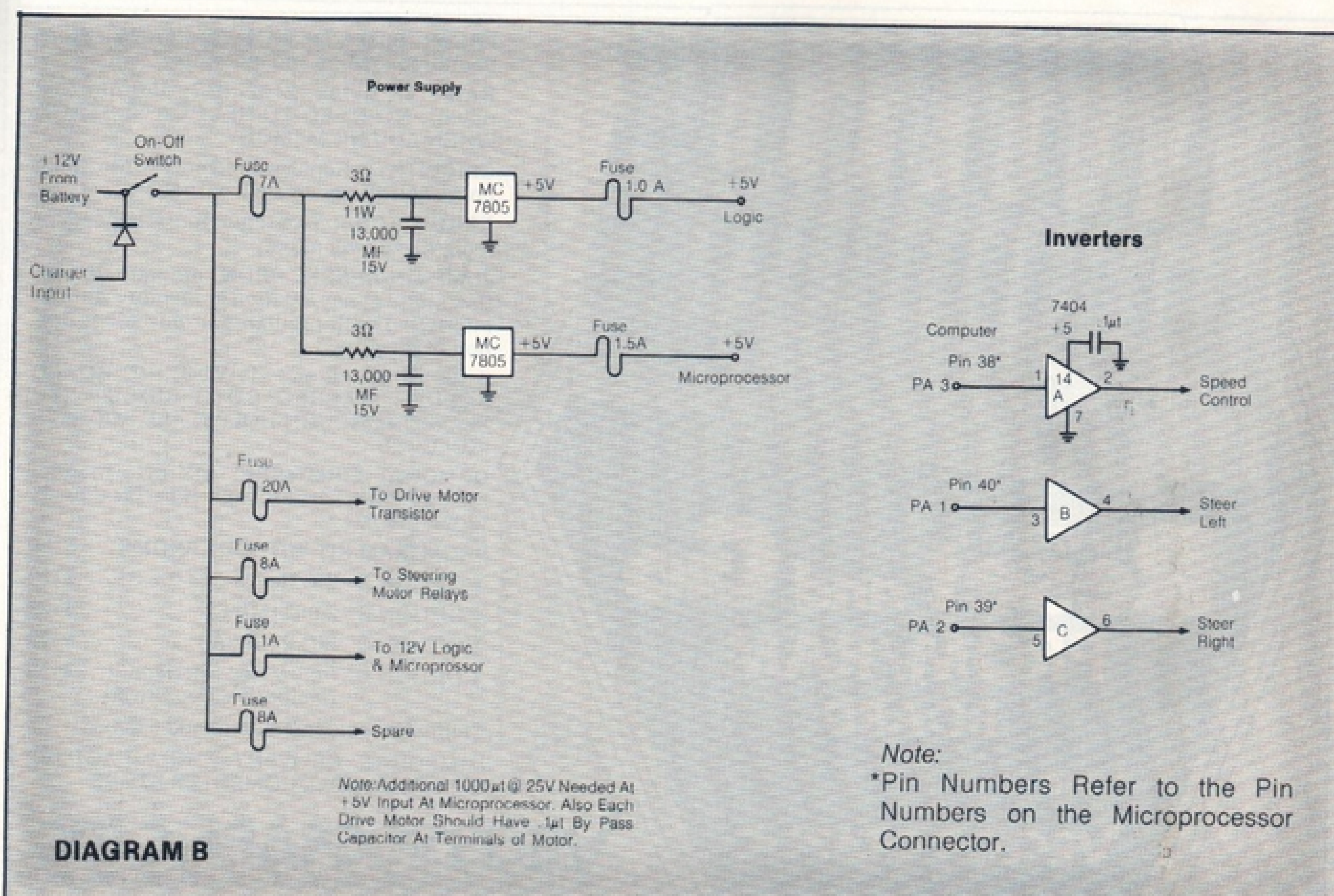


DIAGRAM B

Motorized Wheel Control

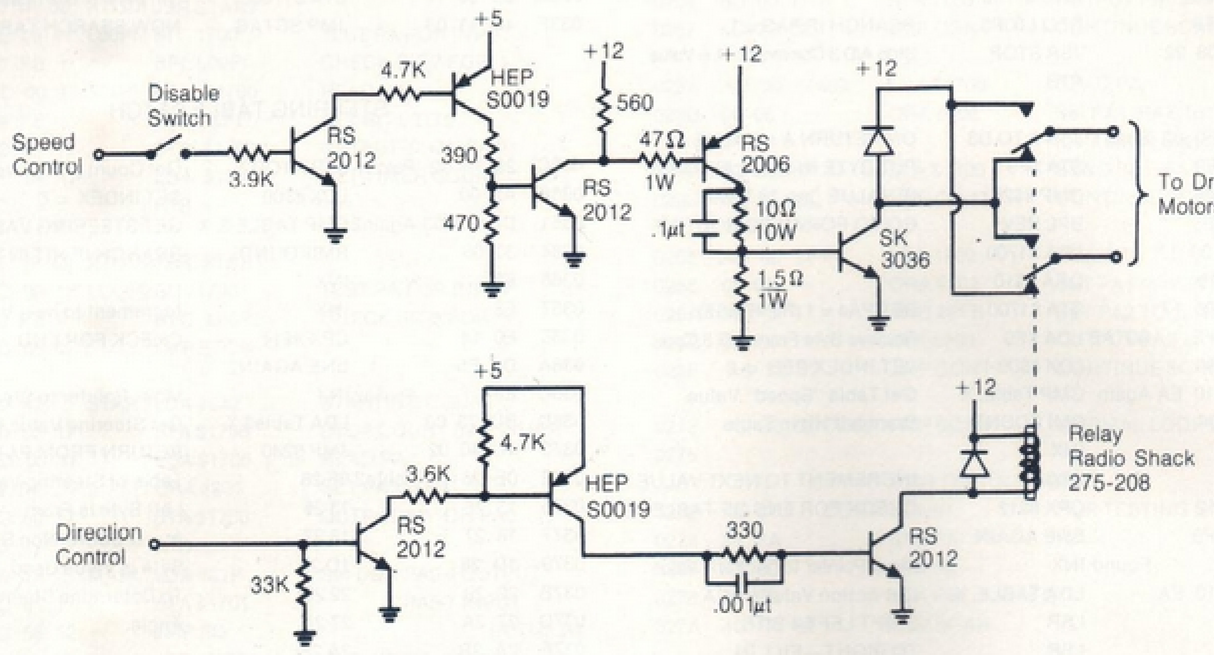
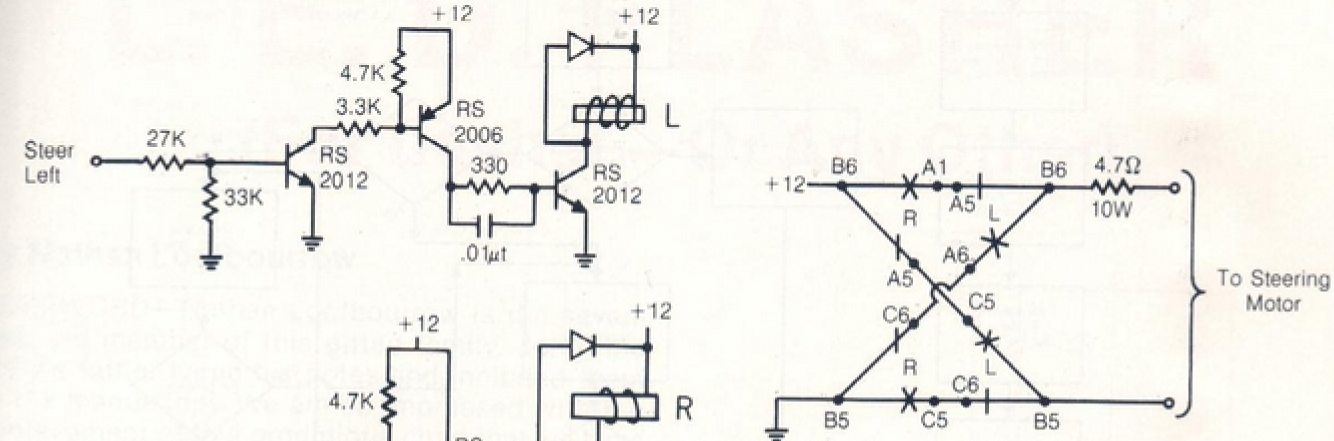
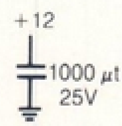


DIAGRAM C

DIAGRAM C continued

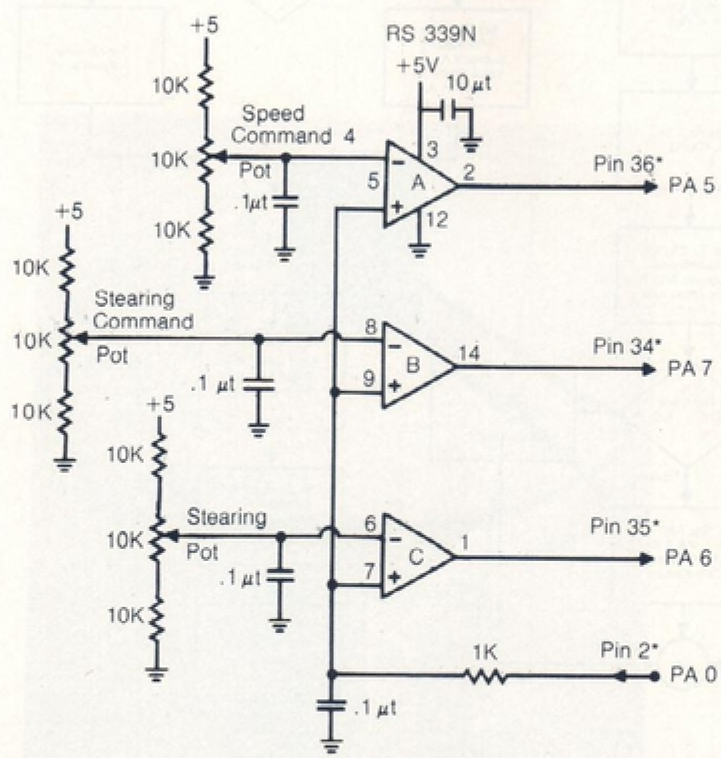
Steering Motor Control





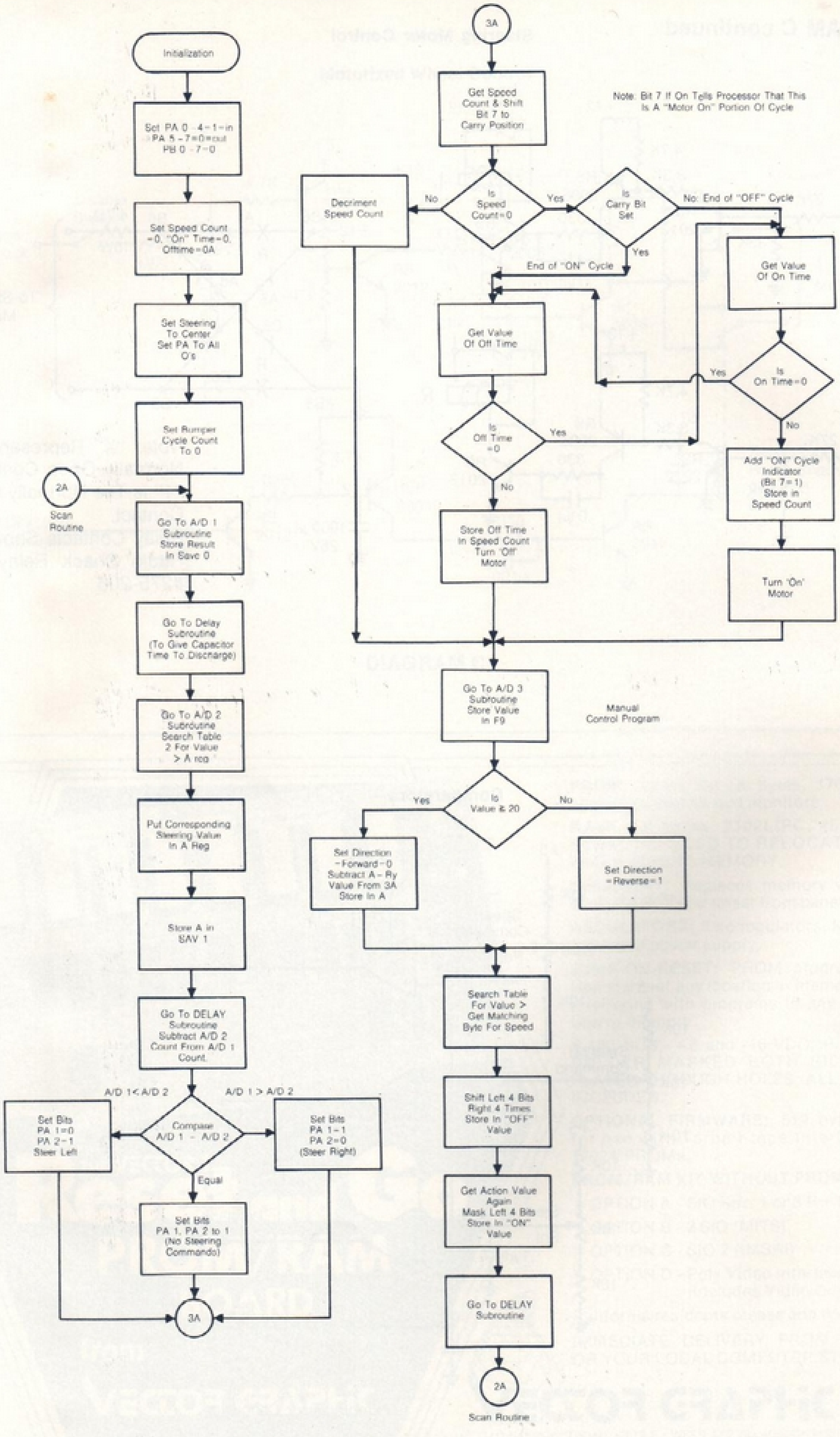
Note: "X" Represents The Normally Open Contacts & "1" Is The Normally Closed Contact.
Relay Contacts Shown For Radio Shack Relays Part #275-206

Comparators



* Microprocessor Connector

DIAGRAM D



Flow Chart for Joystick Control Program