Arm your Atari

by Ted Wilmot

Robotics is an exciting new field. Along with Artificial Intelligence, it's bound to create a new Industrial Revolution. Unfortunately, most people are fascinated by robots, but few get hands-on experience, due to the high cost and industrial nature of the mechanisms.

Unless you're rich or are a mechanical wizard, the robotics scene has probably drifted from your thoughts. But take heed, Atari fans, there's a new kid in town. His name is Armatron, and he wants to meet your computer.

Who is this Armatron, anyway? I first learned of him some time ago, browsing through a Radio Shack catalog. Off the page jumped the ad for their Armatron robot arm. Intrigued, I read the blurb. Before I could finish, the little kid inside me was shouting, "Get it!"

Soon I was on my way to check one out. My initial impression was so positive that, even before I'd purchased one, visions of interfacing it to my computer were percolating through my mind.

For about $30, you can be the proud owner of an Armatron. For an additional $30 and three evenings' work, you can convert the "toy" into a precision computer-controlled servo-system. The process requires a basic knowledge of electronics and simple hand tools. Here's the procedure.

The mechanics.

First, remove the arm's plastic housing. Take out the seven Phillips head screws on the bottom of the housing. Be sure to remove the housing with the arm in normal operating position (i.e., the arm on top). Remove the joysticks, along with the top of the housing, and set those parts aside.

You'll now notice the complex mechanical transmission to distribute power from the single permanent magnet motor to the rest of the arm. Figure 1 shows the internal arrangement of the various assemblies within the transmission.

Now, remove the three largest ring gears on top of the gear housing (see Figure 1). With the gears in a safe place, take out the eight Phillips head screws holding down the gear and joystick housings. *Note: one screw is somewhat hidden inside the ring gear pit.*

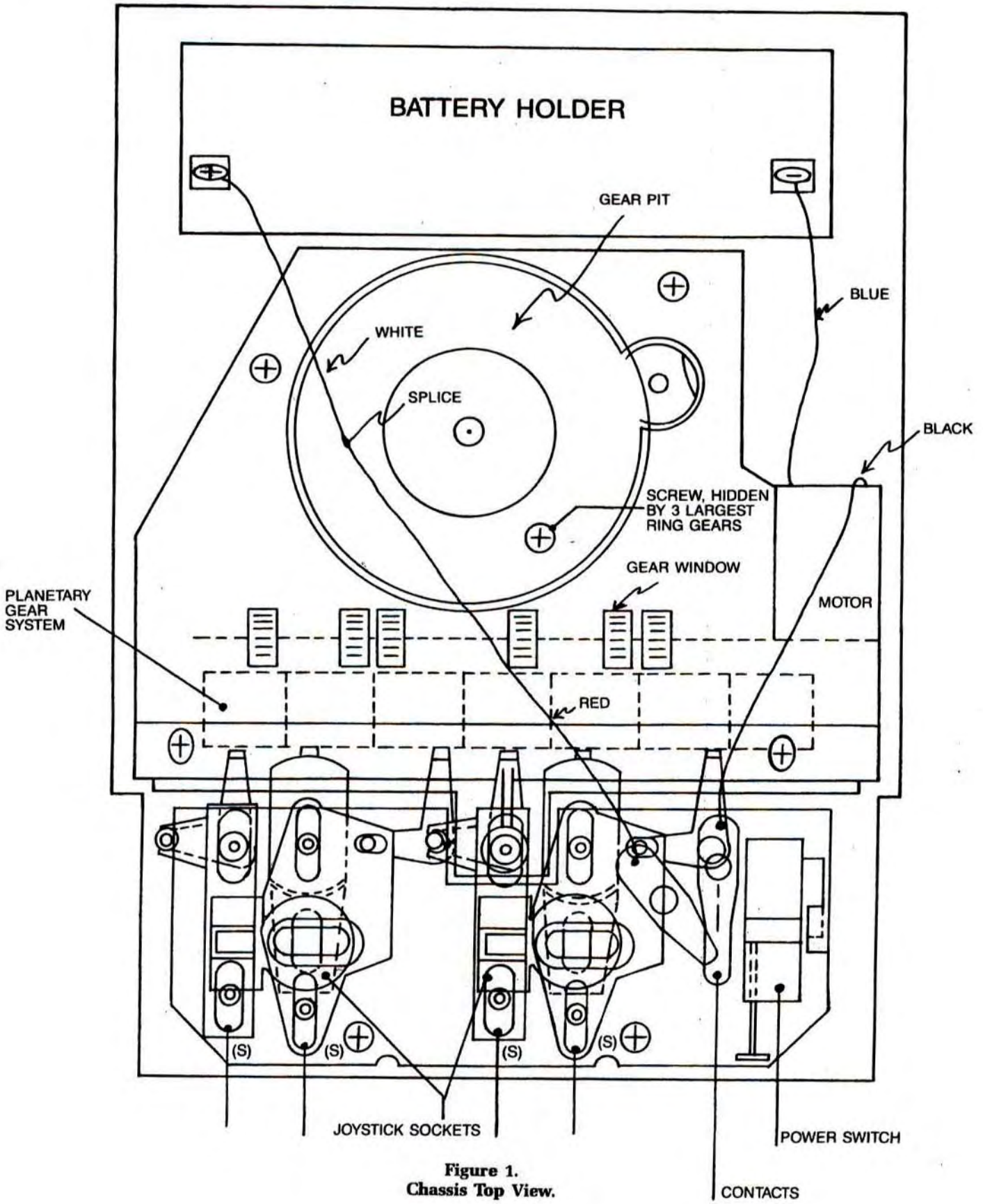


Figure 1.
Chassis Top View.

At this point, the "energy level" mechanism should be removed. Disconnect both wires at the energy level switch (two pieces of copper over the energy level indicator) and splice them together, eliminating the switch. Now, remove the energy level switch and the energy level indicator (the orange tube with some gears at one end).

With the energy level switch bypassed, remove the gear housing, then the joystick housing. Now, remove the two remaining gears, those that powered the energy level indicator (both are worm gears with regular gear ends just above the motor shaft).

Now that all the preliminary work is done, we'll begin interfacing the arm's transmission to the computer.

The arm's joysticks control a series of six plates, which connect with cams on a rotating series of planetary gears. The motor continuously rotates the planetary gear system, and when a joystick is moved, one of three plates connects with a corresponding cam on the gear system to transmit power from the motor to the arm.

To interface our electrical computer to the arm's mechanical transmission requires some sort of electrical to mechanical converter. Of course, a motor comes to mind, but it would be very tricky to install a separate motor to run each of the arm's joints. Therefore, I chose to use solenoids to simulate the action of the joysticks, instead.

This way, solenoids can be mounted outside the arm's housing and connected mechanically to the joystick plates via nylon fishing line.

A total of twenty holes must be drilled in the bottom portion of the arm's housing. Twelve holes give the fishing line a direct route from the solenoids to their corresponding plates. The remainder are to fasten the solenoids' Plexiglas supports and legs.

Figure 2 shows the location of holes to be drilled to accommodate the fishing line. The holes have numbers next to them, indicating the distance from the bottom of the chassis at which they should be drilled. For best results, use a #60 printed circuit board drill to make the holes.

NOTE

Part 1 of this article will work with any 8-bit Atari. However, Part 2 requires three joystick ports, so it's not compatible with the XL and XE series computers.

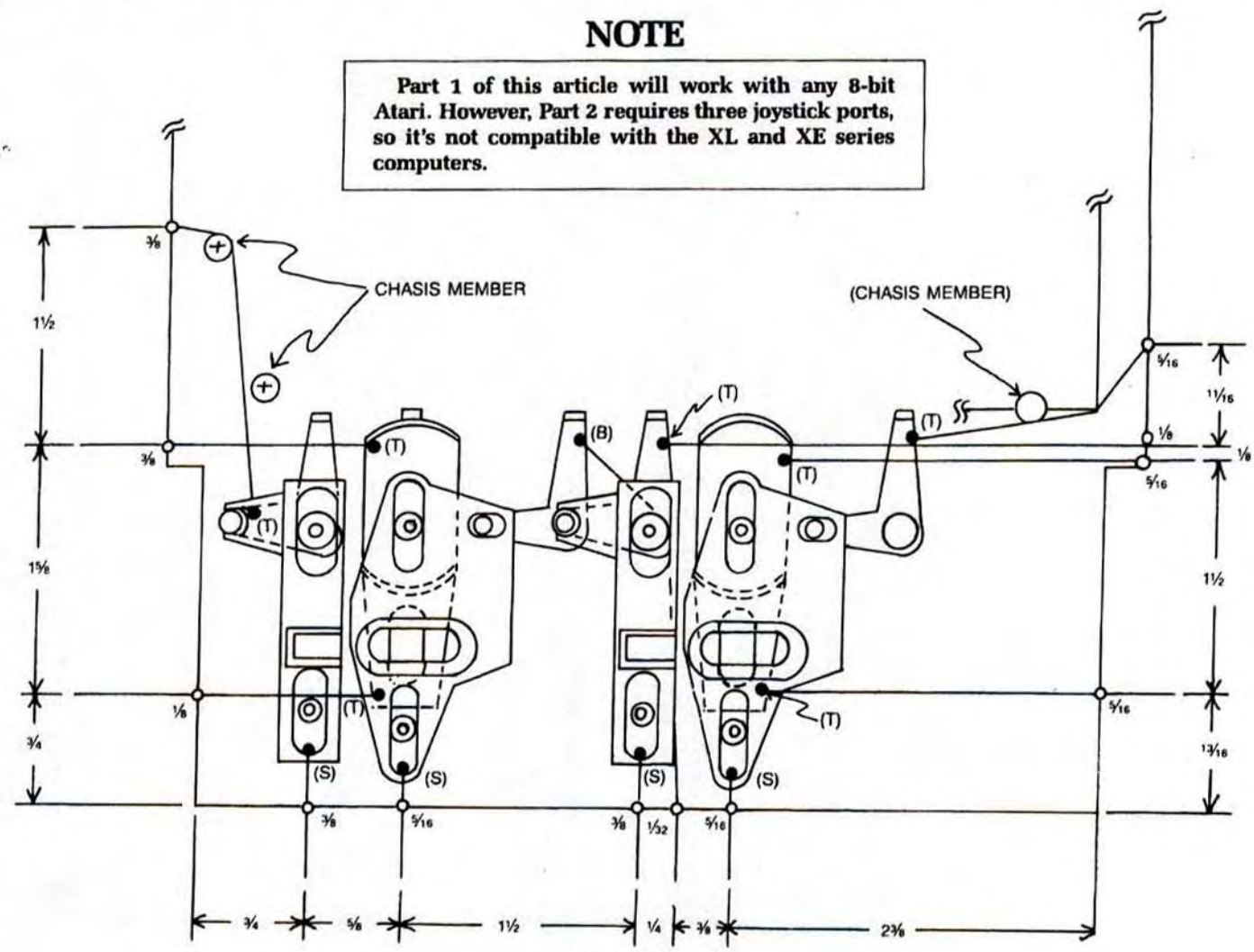


Figure 2.
Plate Assembly View/Drill Guide.

Figure 3 shows the location of the remaining eight holes. With all the chassis holes completed, the plates are now ready to be drilled. For best results, use a 1/8" drill.

Each joystick has its own plate assembly, composed of five parts: three plates and two levers. Also, each plate assembly is a mirror image of the other. Figure 2 shows a view of the plate assembly and where the holes should be drilled in the various parts. The letters *B*, *T* and *S* indicate where the nylon fishing line should exit the part.

For example, the rightmost lever in the plate assembly has a *T*, indicating that the line should exit through the top of the hole. The second to the right plate has an *S*, indicating that the line should exit from the side.

Again, a #60 PCB drill should be used to make all holes. Once all the plate holes are completely cut (twelve 20-inch pieces of 12-pound test nylon fishing line), the line should be tied in a knot about four times, before being drawn through the part (a good-sized knot, so the line won't slip through the hole). Refer to Figure 2 to see which hole each line passes through.

Now, let's make the legs and solenoid supports. There's a lot of flexibility in the leg design. The only real requirement is that they raise the arm at least 1 inch. Why? Well, later on, we'll be mounting a potentiometer under the arm's chassis for directional feedback.

I found scrap 2x4s cut into 1-inch square blocks worked well, but anything will do—as long as it raises the arm 1 inch. Oh, by the way, you'll need four legs for this critter.

The solenoid supports will be a bit more complicated. You'll need three pieces of 4 1/2 x 5 1/2 x 1/8-inch Plexiglas (or similar material).

Figure 3 shows the shapes necessary and their location on the chassis. Now, mount the legs and solenoid supports as per Figure 3.

Check the fishing lines to be sure they won't catch on anything (like the planetary gears or plates) during normal operation, then reinstall the gear and joystick housings.

Next, install the solenoids in their mounting holes (slots), as close to the arm as possible. Hand tighten them.

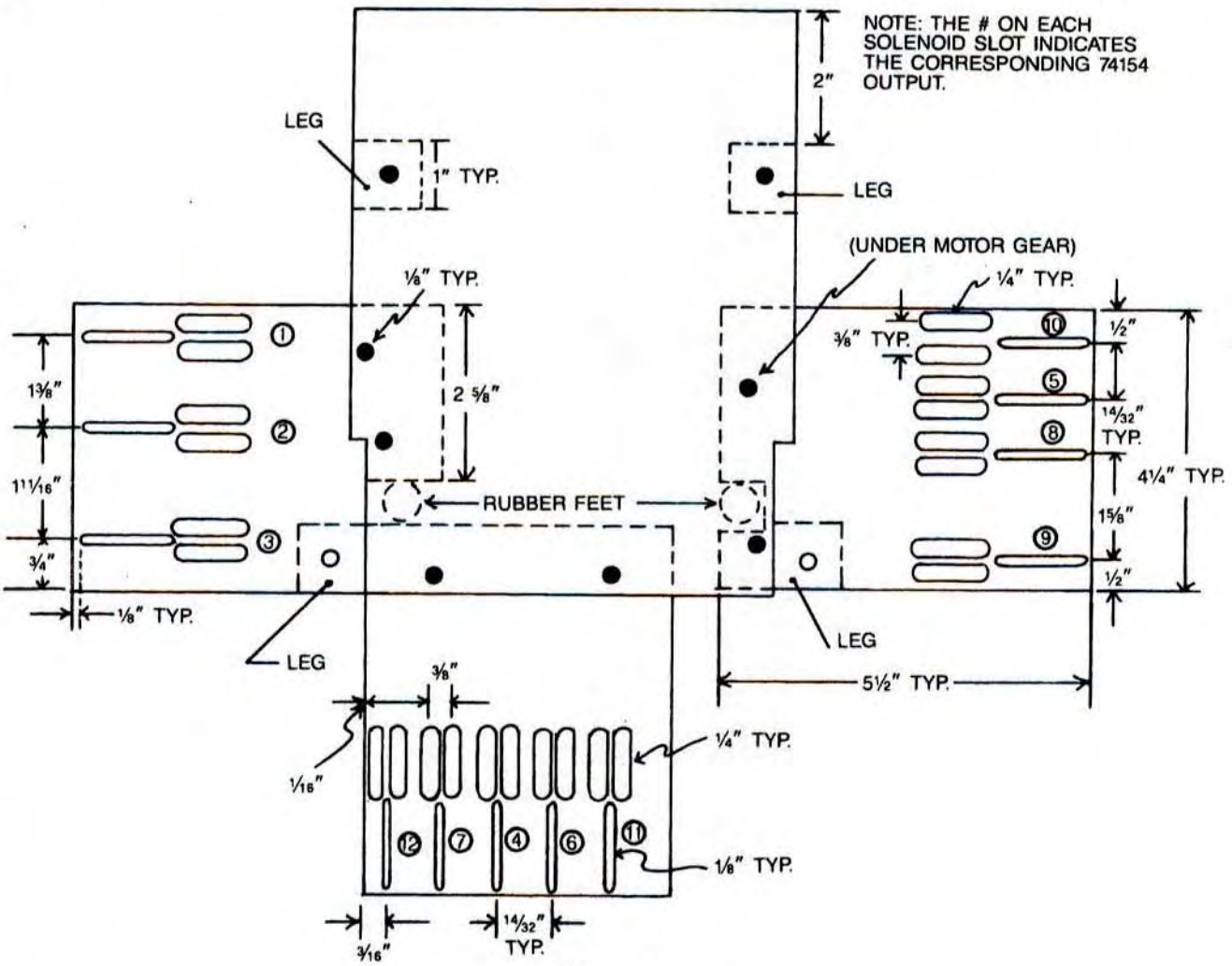


Figure 3.
Leg/Solenoid Mount View.

Note: the solenoids I used (listed in the parts list) had their ends bent perpendicular to the plunger axis. If you use these too, bend them so each end is parallel with the plunger axis.

Now, install the joysticks in their sockets (on top of the joystick housing). Starting at one corner, pull a piece of fishing line snug, until it just starts to move the joystick. With the solenoid plunger fully retracted, tie the line firmly to the solenoid in three to four knots. Repeat this procedure until all solenoids have been connected to their corresponding lines.

At this point, the three ring gears should be replaced in the top of the gear housing and the top or chassis of the arm reassembled.

Making it work.

Congratulations! Most of the mechanical work is finished. Now for some electronics.

Our first task will be to construct a power supply for the solenoids. Why a separate power supply? Well, solenoids are generally high power devices requiring substantial current to operate, far more than our computer is capable of supplying.

For that reason, we'll use a separate source to drive the solenoids, having the computer control them. Figure 4 shows the schematic diagram of the raw DC solenoid power supply. This gives the solenoids 12 volts DC.

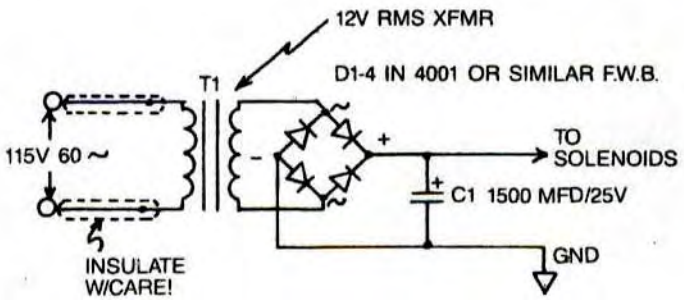


Figure 4.
Power Supply Schematic.

If you're using solenoids other than those mentioned in the parts list, check their voltage requirements. Use an appropriate transformer, to avoid overloading the power source or the solenoids.

Your power supply may be constructed using any common circuit-building technique (wire wrap, PC board, and the like). I used wire wrap construction; it's fast and inexpensive. *Note:* the transformer's primary leads must be well insulated, to avoid the inevitable shock hazard!

With the power source ready, turn on the arm and solenoid power source. Start at one corner and energize a solenoid. Then loosen the solenoid screw and slowly pull the solenoid away from the chassis until the corresponding joint is activated. Tighten the solenoid screw and repeat the procedure until all solenoids have been aligned.

Now we're ready to interface the formerly mechanical arm to our electronic computer. Joystick port 0 will be used to output data to the arm.

The port has a 4-bit output, capable of sixteen unique data patterns. To make life easy, we'll use a 74154 four- to sixteen-line decoder to change the computer's binary output into a form usable by the arm.

Basically, the decoder has four input and sixteen output lines. The output lines are normally at logic 1 (5 volts). When binary data is sent to the input lines, the corresponding output will go to a logic 0, or ground.

For example, suppose we sent a 5 (or *0101*) to the decoder. Output line 5 would go low, while all the others remained high.

At this point, we've only decoded the computer's binary output. We still have to drive the solenoids. Like the computer, the 74154 has a limited current carrying capability. Therefore, we must buffer the decoder's output, so it can supply enough current to drive the solenoids.

I used reed relays to accomplish the buffering, simply because there were quite a few in my "junk box." However, if you have an affinity for transistors, they may also be used. Figure 5 shows the decoder/buffer board schematic, and the joystick plug wiring.

Note: the relay and transistor schemes are both shown, to allow for greater flexibility. Again, the decoder board may be constructed using any of the common circuit-building techniques. I elected to go with the wire wrap method on this board, too.

Ta-da! You've now modified the arm for computer control. Now for some software to control it. . .

Listing 1 is a simple BASIC demo program in BASIC to get you started. Essentially, it sets up joystick port 0 for output, then outputs data using a for. . . next loop and data statements. If you wired and modified the arm according to instructions, the values listed in Table 1 will move the corresponding joints.

I'm sure you'll become very skilled at maneuvering the arm by using timing loops or repeated data values. However, no matter how good you get at guessing the arm's position, you can never really be sure of its location. That's why directional feedback is required.

In the second part of this article, I'll explain how to add directional feedback to the arm, using potentiometers (Atari paddle controllers) and some new software. Until then, have fun building!

The two-letter checksum code preceding the line numbers here is *not* a part of the BASIC program. For further information, see the *BASIC Editor*, page 17.

Listing 1.
BASIC listing.

```
ZT 10 P=PEEK(54018):POKE 54018,P-4:POKE 5
   4016,127:POKE 54018,P
EW 20 FOR A=1 TO 12
XY 30 READ D
FA 40 POKE 54016,D+64:POKE 54016,D
FS 50 FOR W=1 TO 450:NEXT W
GA 60 NEXT A
GL 70 POKE 54016,64:POKE 54016,0
AG 80 DATA 1,2,3,4,5,6,7,8,9,10,11,12
YX 90 END
```

Table 1.

OUTPUT DATA	RESPONSE
1	Close Jaw
2	Elbow Left
3	Elbow Right
4	Wrist CCW Rotation
5	Wrist CW Rotation
6	Wrist Up
7	Wrist Down
8	Shoulder CW Rotation
9	Shoulder CCW Rotation
10	Shoulder Up Rotation
11	Shoulder Down Rotation
12	Open Jaw

Parts List.

QTY.	DESCRIPTION	Vendor/Part #
(12)	12V Solenoids	All electronics SOL-12D, $1.50 ea.
(1)	74 LS 154 4- to 16-bit decoder	R/S, Jameco, etc.
(12)	Transistors/buffers	ECG 159
or		
(12)	Reed Relays, any type with ½-amp contacts	
(1)	12V Transformer, ½-amp and line cord	
(1)	Bridge rectifier or 4-inch 4001 diodes	
(1)	2500 MFD filter capacitor	
(3)	9-pin female connectors	
(4)	1 MΩ linear potentiometers	
(2)	Micro switches	

Miscellaneous wire, metal for mounting brackets, fishing line, etc.

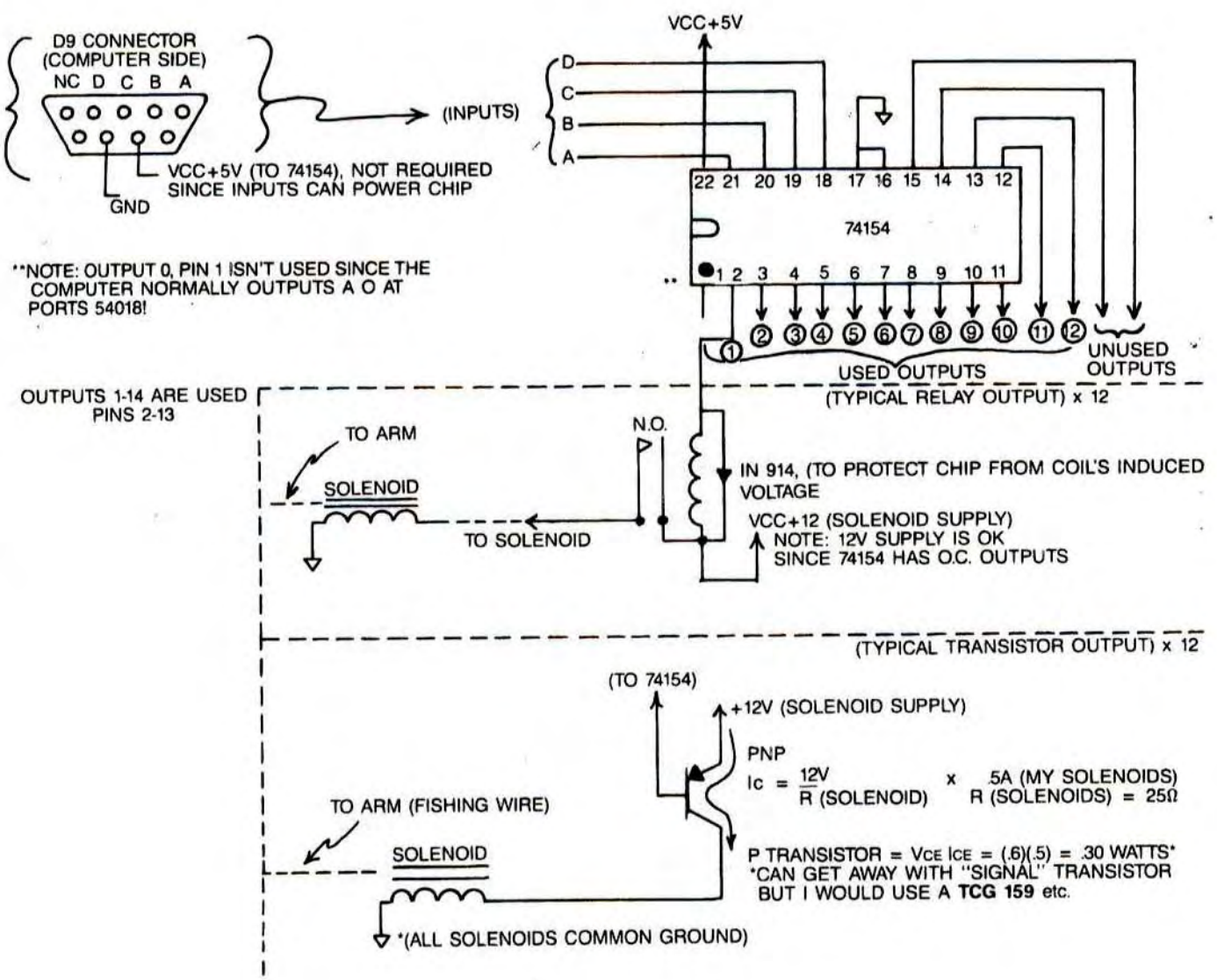
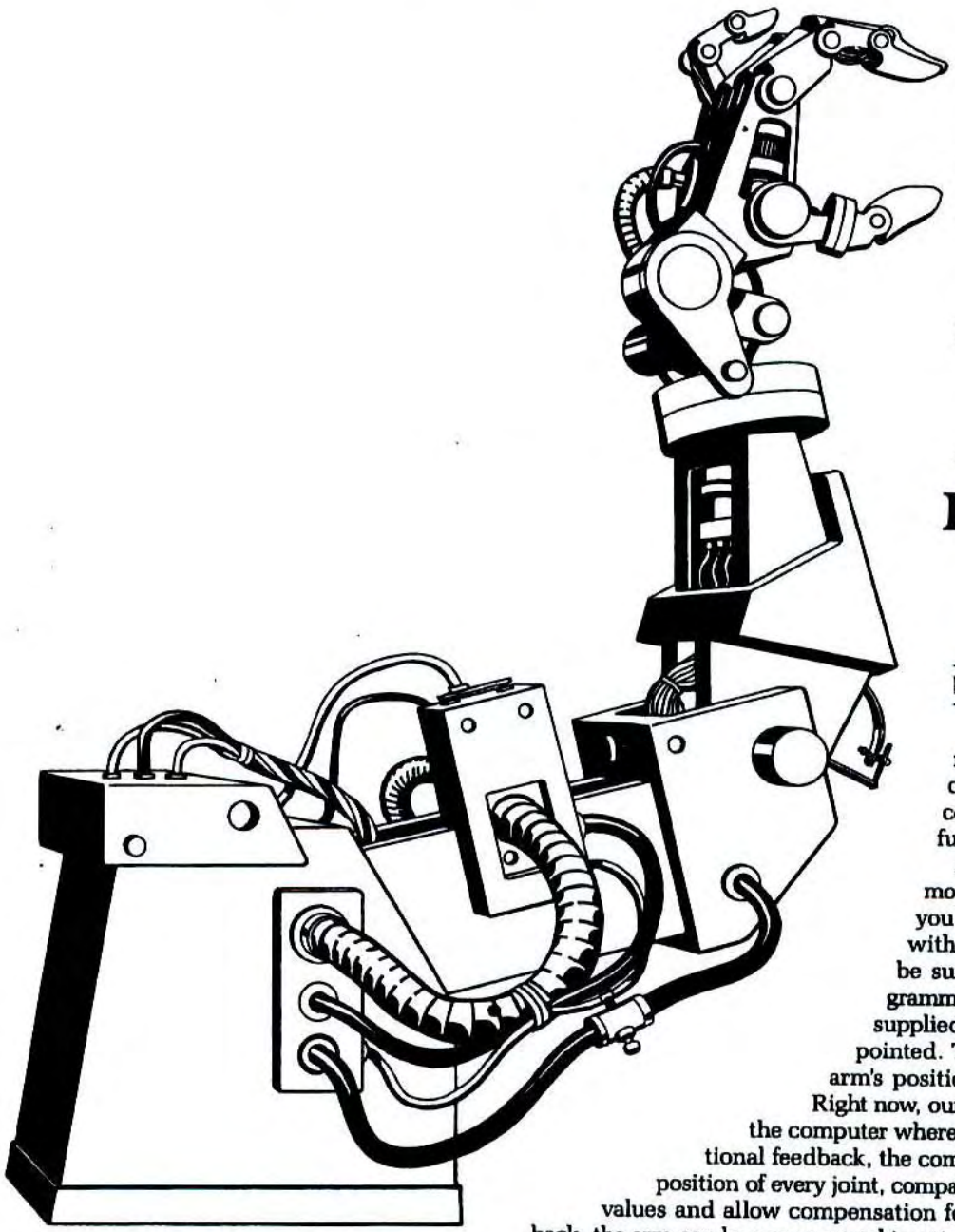


Figure 5.
Decoder Schematic/Joystick Wiring Diagram.



Arm your Atari

Part 2

by Ted Wilmot

Welcome back for another installment of **Arm your Atari**. I hope your construction efforts have been successful up to now, because the real fun's about to begin.

As I mentioned at the close of last month's section, no matter how skilled you become, when you control the arm with timing loops, you can never really be sure of its position. If you tried programming the arm to move the game pieces supplied with it, you were greatly disappointed. There was no way of predicting the arm's position.

Right now, our arm is essentially lost —it can't tell the computer where it is in space. However, with directional feedback, the computer can continuously monitor the position of every joint, compare this data to some preprogrammed values and allow compensation for any errors. Moreover, with feedback, the arm can be programmed to automatically perform any task the operator would have had to execute manually. Also, with feedback, the arm will be able to repeat tasks with great precision—just like an industrial robot.

To add feedback to the arm, potentiometers (Atari paddle controllers) must be attached to every joint. This way, the computer can simply read (in BASIC) the values of the paddles. It can then output data corresponding to the direction the arm should be moved, with respect to some preset values. In all, four potentiometers will be required: two for the shoulder joints, one for the elbow and one for the wrist.

If you've been keeping track, two joystick ports will be required for the feedback data (one port for every two potentiometers), in addition to the port required for the output data. Let's see. . .that's three!

I'll bet you XL owners are really happy at this point, but don't despair; I'm revising the hardware/software to allow you to get away with two ports, with a slight decrease in speed. If there's enough interest, I'll test the design and send it in.

Onward.

Enough preliminary stuff; let's plunge in! To begin with, you'll need four (4) 1-meg ohm linear taper potentiometers with ¼" shafts (preferably plastic). Atari paddles are really 1-meg potentiometers in disguise, but I wouldn't recommend using them. The shafts are too large, and the potentiometer housings are difficult to work with.

Arm your Atari *continued*

Cut the shafts of two potentiometers, so that ½" projects from the housing. Then cut four pieces of sheet metal into ½" by 3" strips. Drill them according to Figure 1, depending on the type of potentiometer you're using.

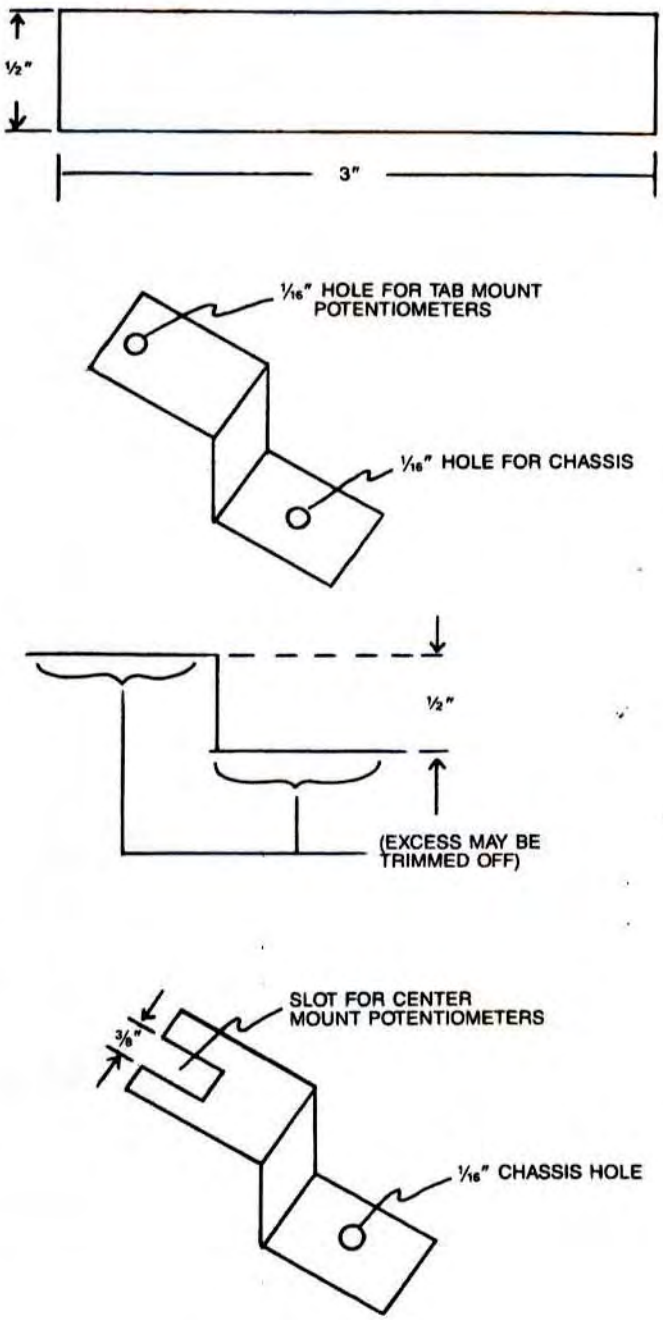


Figure 1.

Next, attach two potentiometers and strips, bending the strips as per Figure 2. Then set the elbow/wrist joints in their mid-positions. Set the potentiometers in their mid-positions, as well. Referring to Figure 3, install the wrist and elbow potentiometers. Wasn't that easy?

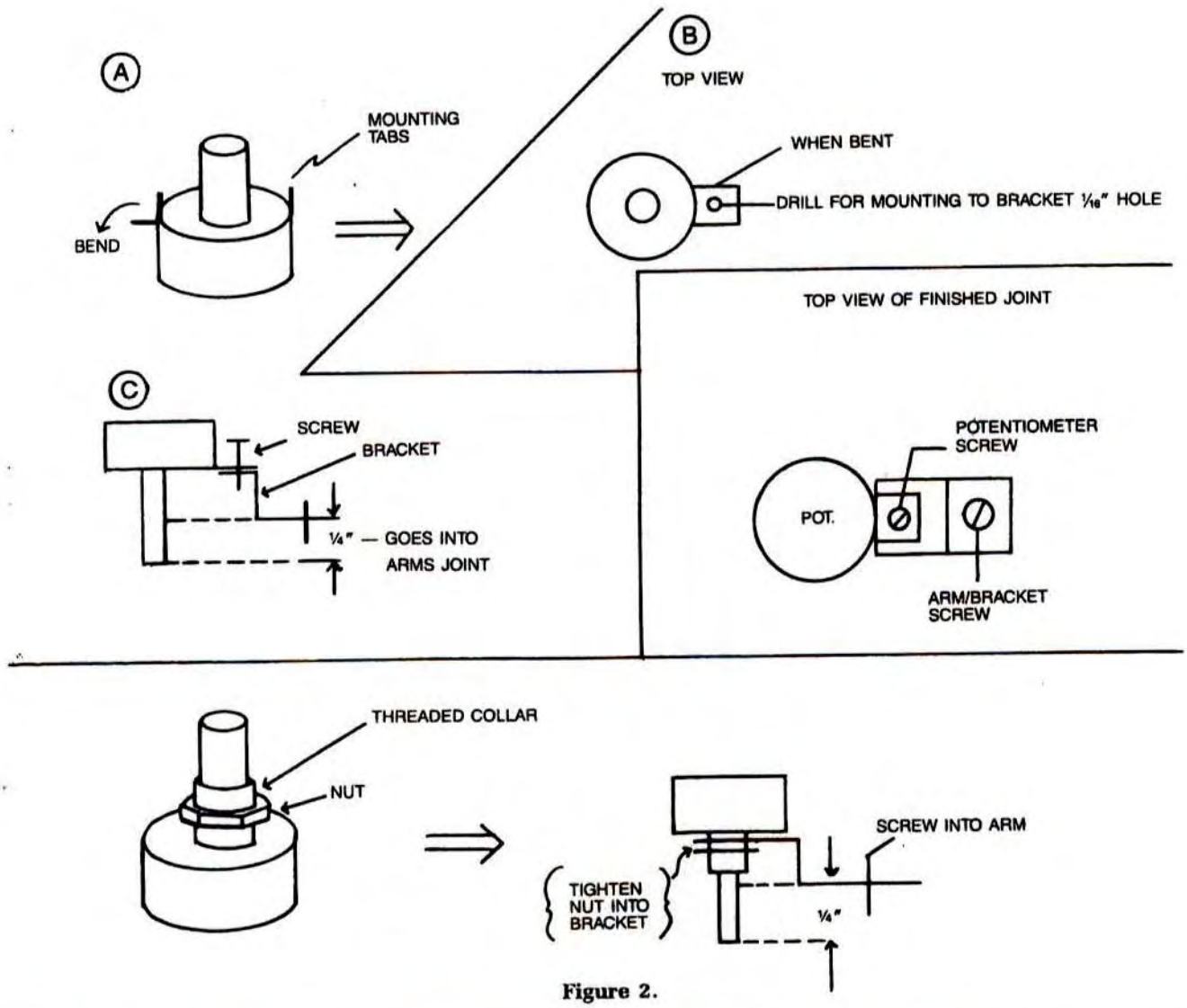
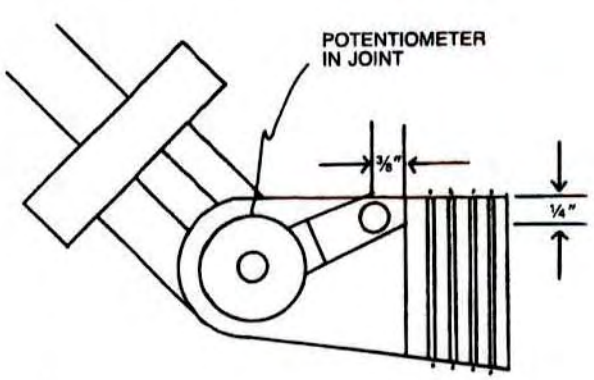
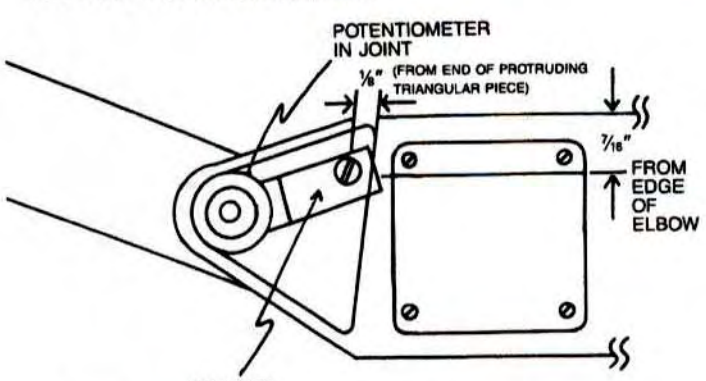


Figure 2.

WRIST POTENTIOMETER:



ELBOW POTENTIOMETER:



Note: These dimensions are important! Internal parts may be damaged if not followed.

Figure 3.

Beware—life gets harder. Drill two 3/32" holes in the shafts of the remaining two potentiometers, to a 1/4" depth. Now obtain a 3" piece of 7/8"-round stock threaded with 6NC32 threads. A cut off screw works well, or you can thread a piece of the energy level indicator support left over from last time.

Cut the threaded stock into 1" and 2" pieces. Screw the 1" piece into one of the potentiometers. Now, remove the housing on the end of the arm, the one that says "Radio Shack ARMATRON," and drill a 1/4" hole in the housing, as in Figure 4.

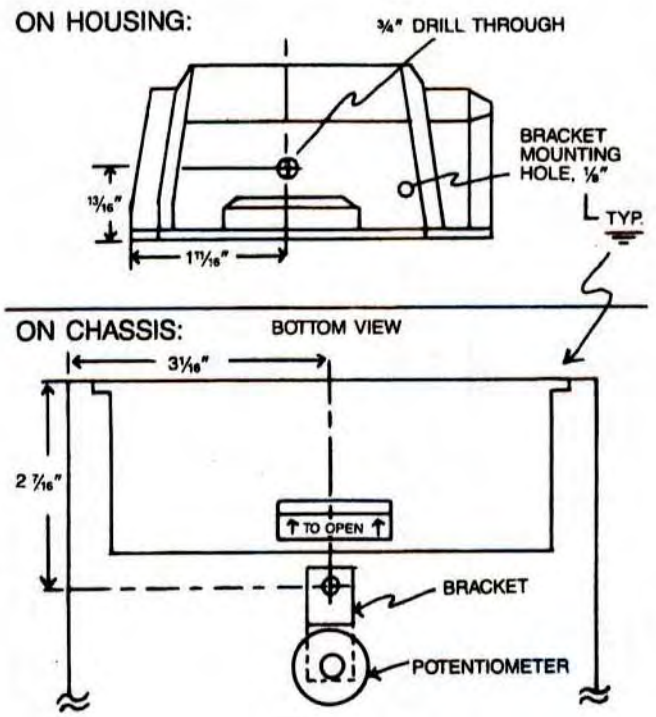


Figure 4.
Shoulder Potentiometer Instructions — Side View.

Now, remove the screw that would lie directly behind the hole and reinstall the housing. Firmly screw the end of the shaft into the hole. Next, attach one of the sheet metal strips to the potentiometer, and drill a hole through the strip and the housing as in Figure 4. Use a small machine screw to hold the bracket to the housing.

Now for the bottom potentiometer. As I mentioned last time, a potentiometer has to be mounted under the arm's chassis to detect the shoulder's position. As you probably know by now, the shoulder, like the wrist, is capable of rotary motion.

Unfortunately, the potentiometer has a limited turning angle. Therefore, we must construct a free-spinning potentiometer capable of rotary motion. To do this, we must disassemble the remaining potentiometer. With it taken apart, you'll notice a dimple in the housing directly below the contacts. This is the enemy; destroy it, being careful not to damage the rest of the potentiometer's housing.

With the dimple removed, the wipers (electrical contacts on the shaft's end) must be altered. Bend the wipers at the ends, so they won't get caught on the contacts—regardless of the direction of rotation.

Now, remove the screw directly below the arm's main shaft; screw in the remaining 2" threaded piece. Attach the remaining metal bracket to the potentiometer and screw it onto the shaft. Drill a hole through the bracket and the chassis as seen in Figure 4 and attach the bracket to the chassis using a sheet metal screw.

Now for the wrist. . .Unlike the other arm joints, the wrist joint is nearly impossible to mount a potentiometer on. The joint is fed by three gears: one to raise and lower, one to rotate, and one to open or close the "jaw." All are contained in the wrist housing, making it too cramped to mount a potentiometer inside.

How can we achieve feedback from the wrist to the computer? Well, I'm sure you electrically oriented types have just thought up at least a dozen ways—optical, magnetic and similar approaches, all of which can be implemented outside the wrist housing.

While optical and magnetic routes are perfectly acceptable, they would generally be too expensive and technically involved for our application. I've decided to use micro switches for the job. In this way, the switches can be mounted outside the wrist housing, and may be actuated by the turning jaw—thus eliminating any wires to the jaw itself!

Moreover, the switches, like the potentiometers, can be read directly by the computer in BASIC with a PTRIG command. If you have a 400/800 machine with four joystick ports, you could (theoretically) read eight switches. While that amount of resolution would be nice, the wiring would be obtrusive.

I elected to go with two switches on my prototype arm, one each for vertical and horizontal jaw orientation. This doesn't sound like many switches, and there isn't any angular resolution to speak of, but it's more than adequate for our purposes.

Before the switches can be mounted on the wrist housing, the latter must be altered. To do this, the wrist must be dissassembled. Remove the wrist-mounted potentiometer and the two round black "bearings."

Then remove the two screws holding the wrist housing together. Separate them, releasing the jaw and associated gears. Now, file down the round collar at the end of the wrist housing as shown in Figure 5. Next, drill the housing pieces as in Figure 6 and install the switches/mounting brackets.

Note: make sure all switch levers face the same direction of rotation with respect to the jaw's axis of rotation.

With the switches in place, reassemble the wrist housing and reinstall the wrist potentiometer.

Now for some wiring. . .In my arm, the potentiometers and switches were wired as you see in Table 1. It would be worth your while to check the response of each joint individually, with a simple PTRIG/PADDLE(x) routine, before soldering any wires together.

If you're unfamiliar with the wiring convention used in Atari paddles, Figure 7 shows how to connect two potentiometers to a single joystick port.

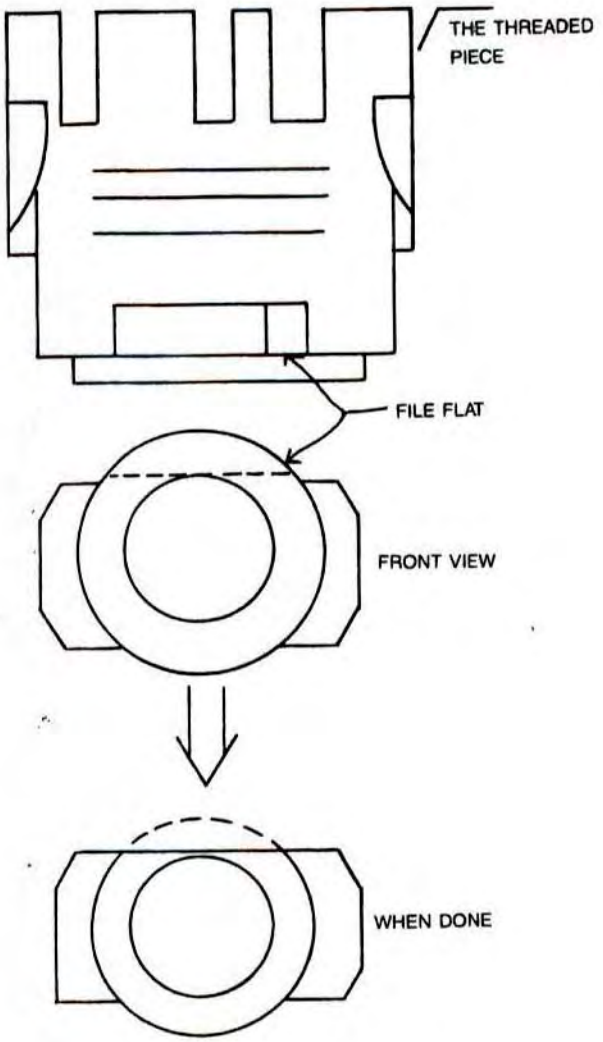


Figure 5.
Collar File Guide.

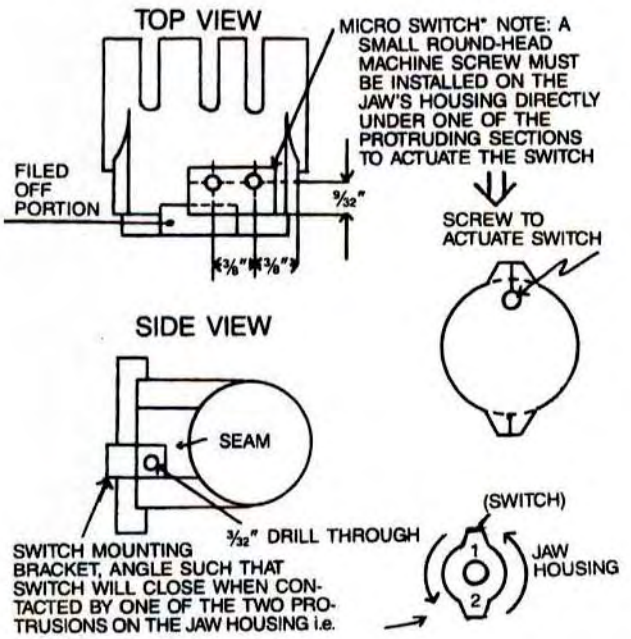


Figure 6.
Collar Drill Guide.

Note: to avoid software conflicts, use joystick port 3 to read the PTRIG (wrist) values—i.e., PTRIG (4-5).

If you've been wondering why no feedback has been added to the jaw itself, the answer is simple. The jaw is internally protected from damage due to too much opening and closing. Therefore, it's best to use simple timing loops to control it.

Trying it out.

Now to test our creation. . . Listing 1 shows a BASIC program that allows you to read in the values of the various joints in a data statement, and have the arm execute the movements you wish—based on your data.

To use the program manually (with the joysticks), move the arm to the object you want to pick up. Align the wrist so one of the two switches is closed (the orientation may be important at times). Type *GOSUB 5000* to read the joint values—and write them down. Manually move the arm to the target destination and, again, type *GOSUB 5000.*

Repeat the above procedure for all the objects you want to move, then put the data into Line 170's data statement. *Note:* change the variable *L* (Line 10) to reflect the number of iterations required (i.e., the number of data values divided by 2).

Good luck with your new toy. I hope all of you enjoy your creations as much as I do mine!

Ted Wilmot is a senior at SUNY Binghamton, majoring in Electrical Engineering Technology. He has an A.A.S. in E.E.T. and has been interested in computer/electronics since 1972. He's designed and built numerous electronic/ software projects, most frequently with BASIC, APL and assembler.

The two-letter checksum code preceding the line numbers here is *not* a part of the BASIC program. For further information, see the *BASIC Editor II*, page 43.

Listing 1.
BASIC listing.

```
ZI 5 P=PEEK(54018):POKE 54018,P-4:POKE 54
   016,127:POKE 54018,P
AV 6 REM SET UP PORT 1 FOR OUTPUT
AJ 10 FOR L=1 TO 4:REM # OF ITERATIONS OF
    # OF DATA VALUES/2
HO 11 FOR Z=1 TO 450:POKE 54016,76:POKE 5
   4016,12:NEXT Z
SN 12 FOR Z=1 TO 550:POKE 54016,74:POKE 5
   4016,10:NEXT Z:POKE 54016,64:POKE 5401
   6,0
```

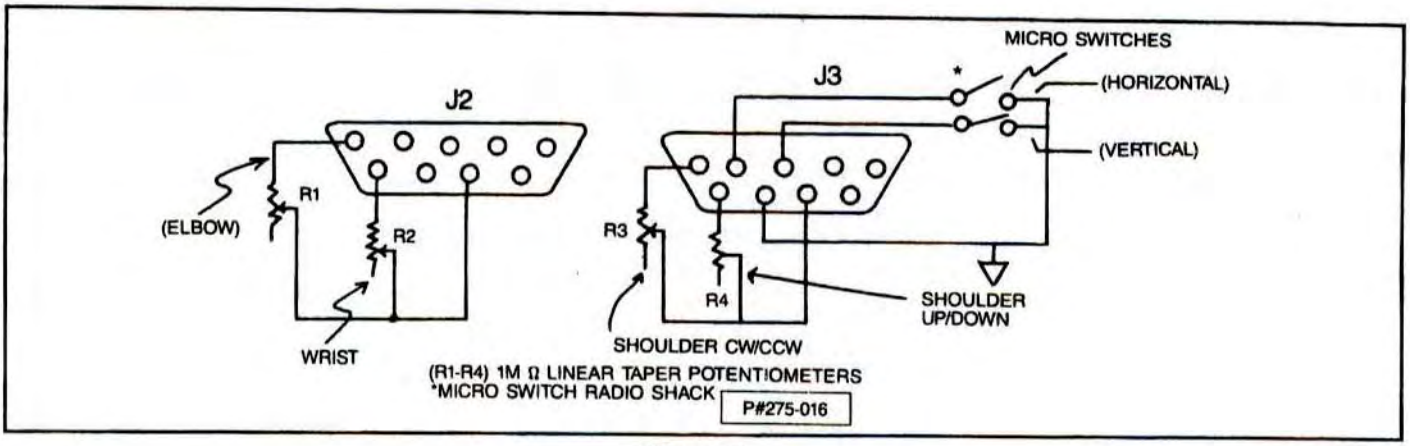


Figure 7.
Paddle/Switch Wiring Diagram.

```
HJ 13 FOR I=1 TO 9
MQ 14 READ PA,PB,PC,PE,WL,PPA,PPB,PPC,PPE
TN 15 GOSUB 40
SW 16 FOR Z=1 TO 450:POKE 54016,65:POKE 5
   4016,1:NEXT Z
OR 17 FOR Z=1 TO 300:POKE 54016,74:POKE 5
   4016,10:NEXT Z:POKE 54016,64:POKE 5401
   6,0
QV 35 PA=PPA:PB=PPB:PC=PPC:PE=PPE
TR 36 GOSUB 40
JF 37 NEXT I
KI 38 NEXT L
ZJ 39 END
DK 40 C=PADDLE(2):REM MAIN MOVEMENT ROUTI
   NE
DE 50 E=PADDLE(4)
DC 60 A=PADDLE(5)
CH 70 B=PADDLE(3)
SY 80 IF A<PA THEN D=8:N=5:X=PA:GOSUB 100
   0
VS 90 IF A>PA THEN D=9:N=5:X=PA:GOSUB 200
   0
ET 100 IF B<PB THEN D=3:N=3:X=PB:GOSUB 10
   00
GE 110 IF B>PB THEN D=2:N=3:X=PB:GOSUB 20
   00
ID 120 IF C<PC THEN D=6:N=2:X=PC:GOSUB 10
   00
LC 130 IF C>PC THEN D=7:N=2:X=PC:GOSUB 20
   00
NJ 131 IF WL=1 THEN GOSUB 3000
NZ 132 IF WL=0 THEN GOSUB 4000
UM 140 IF E<PE THEN D=11:N=4:X=PE:GOSUB 1
   000
VX 150 IF E>PE THEN D=10:N=4:X=PE:GOSUB 2
   000
ZJ 160 RETURN
JX 170 DATA 112,145,1,80,1,228,106,49,131
QU 900 REM
ES 910 REM MAIN OUTPUT ROUTINES
QY 920 REM
HM 1000 A=PADDLE(N)
AN 1020 IF A<X THEN GOTO 1050
FY 1030 POKE 54016,64:POKE 54016,0
AL 1040 RETURN
QY 1050 POKE 54016,D+64:POKE 54016,D
NG 1060 GOTO 1000
HN 2000 A=PADDLE(N)
CF 2020 IF A>X THEN GOTO 2050
FZ 2030 POKE 54016,64:POKE 54016,0
AM 2040 RETURN
QZ 2050 POKE 54016,D+64:POKE 54016,D
N5 2060 GOTO 2000
UE 3000 P=PTRIG(4)
QJ 3010 IF P=1 THEN GOTO 3040
FX 3020 POKE 54016,64:POKE 54016,0
AK 3030 RETURN
PD 3040 POKE 54016,69:POKE 54016,5
OB 3050 GOTO 3000
UT 4000 P=PTRIG(5)
RH 4010 IF P=1 THEN GOTO 4040
FY 4020 POKE 54016,64:POKE 54016,0
AL 4030 RETURN
PP 4040 POKE 54016,69:POKE 54016,5
ON 4050 GOTO 4000
KU 4997 REM
AR 4998 REM ROUTINE TO OBSERVE JOINT VALU
   ES
LC 4999 REM
RY 5000 C=PADDLE(2)
SK 5010 B=PADDLE(3)
TL 5020 A=PADDLE(5)
TX 5030 E=PADDLE(4)
BV 5040 PRINT A,B,C,E
DZ 5050 GOTO 5000
```

ARM OUTPUT DATA	REACTION	PADDLE VALUE REACTION
2	ELBOW L	DECREASE
3	ELBOW R	INCREASE
4	HAND CCW	*
5	HAND CW	*
6	HAND UP	INC
7	HAND DOWN	DEC
8	SHOULDER CW	INC
9	SHOULDER CCW	DEC
10	SHOULDER UP	DEC
11	SHOULDER DOWN	INC
12	JAW OPEN	—
1	JAW CLOSE	—

**Note:* These are PTRIG functions. If your arm is built like mine, or more importantly, for the software, PTRIG (4) goes low when the jaw is vertical. PTRIG (5) goes low when jaw is horizontal.

Table 1.