

COMPUTER-CONTROLLED ROBOT

An easy-to-build interface that lets you use your Commodore 64 computer to control a popular robot arm.

JIM BARBARELLO

For those of us interested in experimenting with robotics, the *Armatron* from Radio Shack was both agony and ecstasy. Still available, *Armatron* is a low-cost, fully functioning robot arm with six degrees of freedom (meaning there are six different portions of the arm whose movement can be controlled). For robotics experimentation, its serious shortcomings include the inability to lift anything of significant weight and its totally mechanical controls. While many different articles have described modifications that allow computer control of the *Armatron*, all required a substantial amount of mechanical skill.

Many of those shortcomings have been eliminated in a new version of that device called the *Mobil Armatron*; that new version easily lends itself to computer control. In this article, we'll describe a simple computer interface for the robot arm, as well as a comprehensive controller program for the Commodore 64.

The interface

The *Mobile Armatron's* movement is controlled using a series of switches. Those switches are used to direct the motion of the unit's base (forward, back, left, and right), and to control the positioning and motion of the arm (arm up, arm down, wrist up, wrist down, wrist turn, and fingers open/clamp). Four D cells are used to provide ± 3 volts for the unit's motors. By changing the polarity of the applied voltage, the motors can be reversed, therefore obtaining complimentary functions such as arm up and arm down. One motor each is used to control the left wheel, the right wheel, the movement of the arm, the movement of the wrist, and the wrist-turn/finger-position. Seven control wires run from the switches, which are located in a control module, to the motors; one wire each for positive or negative voltage, plus a return (ground) wire from each motor.

Because of its design, it is a relatively simple matter to control the *Mobile Armatron* using a personal computer. All that is required is to replace the control module switches with a simple interface. An appropriate circuit, designed for use with a Commodore 64, is shown in Fig. 1.

In that circuit, the switches are replaced by low-current relays. Those relays are activated by seven transistors, which are in turn controlled by seven of the eight available Commodore 64 user-port lines. Use of low-current relays ensures that the 100-mA maximum allowable current draw from the user port is not exceeded.

Construction

While the circuit is simple, we still recommend using a PC board. A suitable pattern for a double-sided board is shown in PC Service. The corresponding parts-placement diagram is shown in Fig. 2. Once the board is etched, inspect it carefully for shorted or open traces, etc.

The main reason for using a double-sided board is mechanical rigidity. Once assembled, the PC-board is mounted on a card-edge socket and soldered in place. With a double-sided board, solder connections can be made on both sides; with a single-sided board, connections can be made only on one side.

If you find the thought of etching a double-sided board intimidating, the component-side pattern can be eliminated. Then, you will need to add a jumper between pad A and edge-connector pad 2. Remember that you will lose the rigidity offered by the double-sided design, so extra care must be taken when handling the unit.



Mount the components on the PC board as shown in Fig. 2. If using a double-sided board, install a feedthrough at pad A; if using a single-sided board, install a jumper as previously discussed. Follow proper construction techniques.

Once the board is complete, insert it into SO1, a 12-position card-edge socket with 0.156-inch spacing. If you are only able to obtain a longer connector, for instance 22-position, it can be cut

TABLE 1—6502 OP CODES

MSD	LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		BRK Implied 1 7	ORA (IND, X) 2 6			TSB ZP 2 5	ORA ZP 2 3	ASL ZP 2 5	RMB0 ZP 2 5	PHP Implied 1 3	ORA IMM 2 2	ASL Accum 1 2		TSB ABS 3 6	ORA ABS 3 4	ASL ABS 3 6	BBR0 ZP 3 5**
1		BPL Relative 2 2**	ORA (IND), Y 2 5*	ORA (IND) 2 5		TRB ZP 2 5	ORA ZP, X 2 4	ASL ZP, X 2 6	RMB1 ZP 2 5	CLC Implied 1 2	ORA ABS, Y 3 4*	INC Accum 1 2		TRB ABS 3 6	ORA ABS, X 3 4*	ASL ABS, X 3 7	BBR1 ZP 3 5**
2		JSR ABS 3 6	AND (IND, X) 2 6			BIT ZP 2 3	AND ZP 2 3	ROL ZP 2 5	RMB2 ZP 2 5	PLP Implied 1 4	AND IMM 2 2	ROL Accum 1 2		BIT ABS 3 4	AND ABS 3 4	ROL ABS 3 6	BBR2 ZP 3 5**
3		BMI Relative 2 2**	AND (IND), Y 2 5*	AND (IND) 2 5		BIT ZP, X 2 4	AND ZP, X 2 4	ROL ZP, X 2 6	RMB3 ZP 2 5	SEC Implied 1 2	AND ABS, Y 3 4*	DEC Accum 1 2		BIT ABS, X 3 4*	AND ABS, X 3 4*	ROL ABS, X 3 7	BBR3 ZP 3 5**
4		RTI Implied 1 6	EOR (IND, X) 2 6				EOR ZP 2 3	LSR ZP 2 5	RMB4 ZP 2 5	PHA Implied 1 3	EOR IMM 2 2	LSR Accum 1 2		JMP ABS 3 3	EOR ABS 3 4	LSR ABS 3 6	BBR4 ZP 3 5**
5		BVC Relative 2 2**	EOR (IND), Y 2 5*	EOR (IND) 2 5			EOR ZP, X 2 4	LSR ZP, X 2 6	RMB5 ZP 2 5	CLI Implied 1 2	EOR ABS, Y 3 4*	PHY Implied 1 3			EOR ABS, X 3 4*	LSR ABS, X 3 7	BBR5 ZP 3 5**
6		RTS Implied 1 6	ADC (IND, X) 2 6†			STZ ZP 2 3	ADC ZP 2 3†	ROR ZP 2 5	RMB6 ZP 2 5	PLA Implied 1 4	ADC IMM 2 2†	ROR Accum 1 2		JMP (ABS) 3 6	ADC ABS 3 4†	ROR ABS 3 6	BBR6 ZP 3 5**
7		BVS Relative 2 2**	ADC (IND), Y 2 5†	ADC (IND) 2 5†		STZ ZP, X 2 4	ADC ZP, X 2 4†	ROR ZP, X 2 6	RMB7 ZP 2 5	SEI Implied 1 2	ADC ABS, Y 3 4†	PLY Implied 1 4		JMP (ABS, X) 3 6	ADC ABS, X 3 4†	ROR ABS, X 3 7	BBR7 ZP 3 5**
8		BRA Relative 2 3*	STA (IND, X) 2 6			STY ZP 2 3	STA ZP 2 3	STX ZP 2 3	SMB0 ZP 2 5	DEY Implied 1 2	BIT IMM 2 2	TXA Implied 1 2		STY ABS 3 4	STA ABS 3 4	STX ABS 3 4	BBS0 ZP 3 5**
9		BCC Relative 2 2**	STA (IND), Y 2 6	STA (IND) 2 5		STY ZP, X 2 4	STA ZP, X 2 4	STX ZP, Y 2 4	SMB1 ZP 2 5	TYA Implied 1 2	STA ABS, Y 3 5	TXS Implied 1 2		STZ ABS 3 4	STA ABS, X 3 5	STZ ABS, X 3 5	BBS1 ZP 3 5**
A		LDY IMM 2 2	LDA (IND, X) 2 6	LDX IMM 2 2		LDY ZP 2 3	LDA ZP 2 3	LDX ZP 2 3	SMB2 ZP 2 5	TAY Implied 1 2	LDA IMM 2 2	TAX Implied 1 2		LDY ABS 3 4	LDA ABS 3 4	LDX ABS 3 4	BBS2 ZP 3 5**
B		BCS Relative 2 2**	LDA (IND), Y 2 5*	LDA (IND) 2 5		LDY ZP, X 2 4	LDA ZP, X 2 4	LDX ZP, Y 2 4	SMB3 ZP 2 5	CLV Implied 1 2	LDA ABS, Y 3 4*	TSX Implied 1 2		LDY ABS, X 3 4*	LDA ABS, X 3 4*	LDX ABS, Y 3 4*	BBS3 ZP 3 5**
C		CPY IMM 2 2	CMP (IND, X) 2 6			CPY ZP 2 3	CMP ZP 2 3	DEC ZP 2 5	SMB4 ZP 2 5	INY Implied 1 2	CMP IMM 2 2	DEX Implied 1 2		CPY ABS 3 4	CMP ABS 3 4	DEC ABS 3 6	BBS4 ZP 3 5**
D		BNE Relative 2 2**	CMP (IND), Y 2 5*	CMP (IND) 2 5			CMP ZP, X 2 4	DEC ZP, X 2 6	SMB5 ZP 2 5	CLD Implied 1 2	CMP ABS, Y 3 4*	PHX Implied 1 3			CMP ABS, X 3 4*	DEC ABS, X 3 7	BBS5 ZP 3 5**
E		CPX IMM 2 2	SBC (IND, X) 2 6†			CPX ZP 2 3	SBC ZP 2 3†	INC ZP 2 5	SMB6 ZP 2 5	INX Implied 1 2	SBC IMM 2 2†	NOP Implied 1 2		CPX ABS 3 4	SBC ABS 3 4†	INC ABS 3 6	BBS6 ZP 3 5**
F		BEQ Relative 2 2**	SBC (IND), Y 2 5†	SBC (IND) 2 5†			SBC ZP, X 2 4†	INC ZP, X 2 6	SMB7 ZP 2 5	SED Implied 1 2	SBC ABS, Y 3 4†	PLX Implied 1 4			SBC ABS, X 3 4†	INC ABS, X 3 7	BBS7 ZP 3 5**



† Add 1 to N if in decimal mode.
 * Add 1 to N if page boundary is crossed.
 ** Add 1 to N if branch occurs to same page;
 Add 2 to N if branch occurs to different page.

Incompatible compatibles

The proliferation of 6502 "clones" (cloning really began in 1975 when IMSAI "cloned" their version of the Altair 8800) can lead to confusion and incompatibility. For example, a program written using Rockwell's enhanced branching instructions is guaranteed not to work on an Apple IIc, which uses the NCR IC.

So programmers and systems developers are faced with a choice: Stick to the original 6502 instruction set and maintain compatibility at the expense of performance, or use the enhanced instructions, knowing that programs may not run on all machines.

The bottom line is that the only way to guarantee compatibility is to stick to the manufacturer's specified hardware. Anything else is a gamble.

In this article we have examined the 6502 family of micro-processors. The original 6502 was used in numerous machines from Apple, Commodore, Atari, and many others. In spite of many predictions of its early demise, it and its descendants are still being used in many personal computers and dedicated controllers. We hope this article has helped you understand the major differences between the various members of the 6502 family. ▶◀

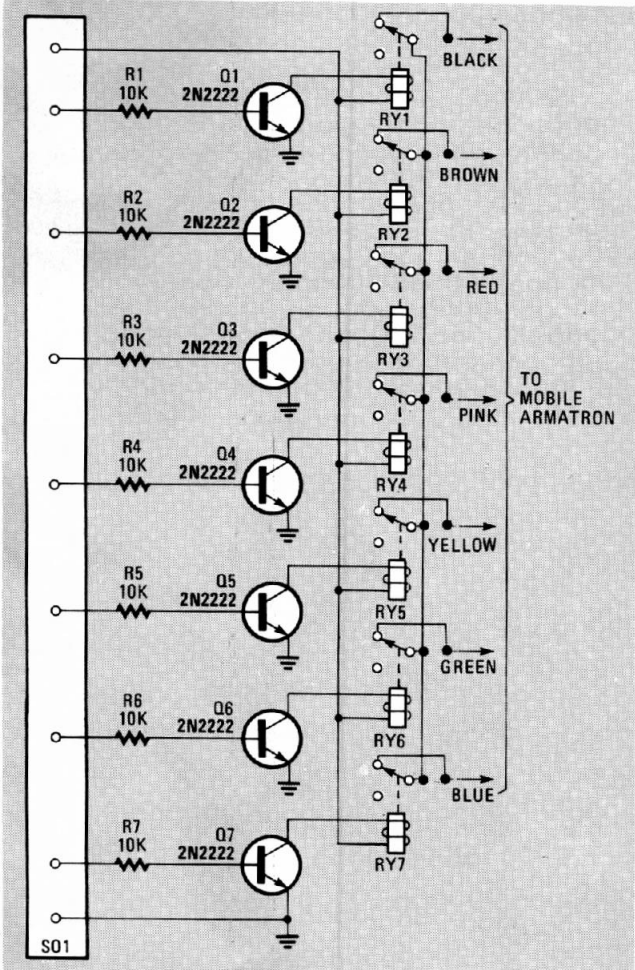


FIG. 1—YOU CAN ADD COMPUTER CONTROL to a Radio-Shack *Mobile Armatron* with this simple interface. It replaces the robot arm's control module.

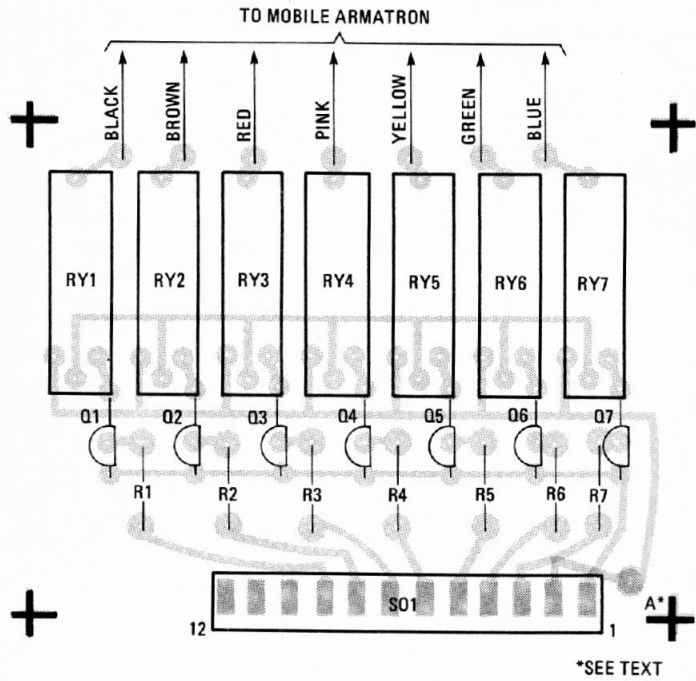


FIG. 2—ONCE ALL PARTS ARE MOUNTED, this double-sided PC board is inserted into a 12-position card-edge connector and the circuit is connected to the *Armatron*.

```

1 rem *****
2 rem ** mobile armatron software **
3 rem ** name: robot **
4 rem ** (c) 1986, jjb **
5 rem ** manalapan, nj 07726 **
6 rem ** v 860928 **
7 rem *****
10 dim a(12),b(250,1):gosub 3000:gosub 5000:poke
    56579,255:poke 56577,0
20 a(1)=38:a(2)=70:a(3)=36:a(4)=34:a(5)=33:a(6)=65
30 a(7)=40:a(8)=72:a(9)=48:a(10)=80:a(11)=66:a(12)=68
40 ss$="1234567890+~":rcs=chr$(5)+chr$(18)
50 gosub 3000:ro=6:co=10:gosub 5050:
    printrc$;" m a i n m e n u "
60 print:printtab(12);"< f1 >:
    learn":print:printtab(12);"< f3 >: do"
70 print:printtab(12);"< f5 >: save":print:printtab(12);
    "< f7 >: retrieve"
80 print:printtab(10);"<ctrl> q: quit (end)
90 ro=19:co=10:gosub 5050:print"which... "
100 get sr$:if sr$="" then 100
110 sr=asc(sr$)-132:if sr=-115 then sr=5
120 if sr<1 or sr>5 then 90
130 on sr gosub 200,300,400,600,800
140 goto 50
200 gosub 1000:j=b(0,0):x=0:rem*** learn mode ***
210 get a$:if a$="" then 210
220 if asc(a$)=133 then return
230 gosub 2000:if i=0 then poke 56577,0:goto 210
240 poke 56577,a(i)
250 get a$:if a$="" then x=x+1:goto 250
260 poke 56577,0
270 j=j+1:b(j,0)=i:b(j,1)=x:x=0:b(0,0)=j
280 ro=i+10:co=5:gosub 5050:print mid$(ss$,i,1);goto 210
300 rem** do procedure
310 printchr$(147):printbl$:
    print" do procedure":printbl$:print
    if b(0,0)=0 then print"no procedure in memory.":
        goto 375
330 print"press any key to begin procedure."
340 get a$:if a$="" then 340
350 print"procedure execution in progress"
355 for i=1 to b(0,0):poke 56577,a(b(i,0))
360 for j=1 to b(i,1)
365 get a$:if a$="" then x=x+1:next j
370 poke 56577,0:for j=1 to 500:next j:
    next i:print"procedure done."
375 ro=18:co=0:gosub5050:
    print"press any key to return to the menu."
380 get a$:if a$="" then 380
390 return
400 rem** save procedure
410 printchr$(147):printbl$:
    print" save procedure":printbl$:print
    if b(0,0)=0 then print"no procedure in memory.":
        goto 480
430 input"enter file name to save";f$
440 open 1,8,15:open 2,8,2,"@@"+f$+" ,s,r":
    input#1,e,ed$,tn,bl:close 1:close 2
450 if e=62 then 500
460 print"file exists. continue (y/n)...":gosub 1500
470 if a$="y" then 500
480 print"abort. press any key..."
490 get a$:if a$="" then 490
495 return
500 open 2,8,2,"@"+f$+" ,s,w":print:
    print"saving procedure. wait."
510 for i=0 to b(0,0):print#2,b(i,0):print#2,b(i,1):
    next i:close 2:close 15
520 print:print"procedure saved. press any key.":goto 490

```

down to the proper length. That's what was done in the author's prototype. (See Fig. 3.) Solder each edge-connector pad to the corresponding terminal on the socket. For double-sided boards, remember to solder on both sides.

Connecting the interface

Place the *Mobile Armatron's* controller module face down and remove the six screws that hold the unit together. When you remove the rear half of the case you will see a PC board that is held in place with just a single screw. Remove that screw and the board.

```

600 rem** retrieve procedure
610 printchr$(147):printbl$:
  print"      retrieve procedure":printbl$:print
620 if b(0,0)=0 then 650
630 print"procedure in memory. continue (y/n)?":gosub 1500
640 if a$="n" then print"abort. ":goto 700
650 input"enter file name to retrieve":f$
660 open 1,8,15:open 2,8,2,"@0:"+f$+",s,r":
  input#1,e,ed$,tn,bl:close 1:close 2
670 if e=0 then 720
680 if e=62 then print"file doesn't exist. ";
690 print"press any key."
700 get a$:if a$="" then 700
710 return
720 print"retrieving procedure. wait.":
  open 2,8,2,"@0:"+f$+",s,r"
730 input#2,b(0,0):input#2,b(0,1)
740 for i=1 to b(0,0):input#2,b(i,0):
  input#2,b(i,1):next:close 2
750 print"retrieval complete. ":goto 690
800 rem** end
810 ro=5:co=10:gosub 5050:for q=1 to 16:
  print tab(10);b$:next
820 ro=10:co=0:gosub 5050:close1:close2
830 print"program ended. to re-enter, type goto 50";
840 print:end
1000 rem** learn mode screen
1005 printchr$(147):printbl$
1006 print" mobile armatron robot learn mode ":
  printbl$:print
1007 print"  press key to do function.  press any other
  key to stop.";
1008 print"  press <f1> to return to menu.":print
1009 print"  key  function":
  print"  ---  -----"
1010 printtab(5);"1 = forward":printtab(5);"2 = backward"
1020 printtab(5);"3 = right forward turn":
  printtab(5);"4 = left forward turn"
1030 printtab(5);"5 = arm up":printtab(5);"6 = arm down"
1040 printtab(5);"7 = wrist up":
  printtab(5);"8 = wrist down"
1050 printtab(5);"9 = hand turn":
  printtab(5);"0 = fingers move in/out"
1060 printtab(5);"+" = right reverse turn"
1070 printtab(5);"-" = left reverse turn"
1080 return
1500 get a$:if a$="" then 1500
1510 a$=chr$(asc(a$) and 223)
1520 if a$<>"y" and a$<>"n" then 1500
1530 print a$:return
2000 rem** position in string
2010 for i=1 to 12:if a$=mid$(ss$,i,1) then 2030
2020 next:i=0:return
2030 ro=i+10:co=5:gosub 5050:printrc$a$:return
3000 rem** format screen=
3010 poke 53280,6:poke 53281,6:printchr$(147):
  b$=chr$(5)+chr$(18)
3020 bl$=b$+"          ":
  print bl$
3030 print b$;"  mobile armatron robot controller  "
3040 printbl$
3050 b$="          "
3060 return
5000 rem** cursor control using plot          kernel ($fff0)
5010 data 162,0,160,0,24,32,240,255,96,999
5020 a=49300:sc=a
5030 read b:if b<>999 then poke a,b:a=a+1:goto 5030
5040 return
5050 poke sc+3,col:poke sc+1,row:sys sc
5060 return

```

Examining the circuit you will see that seven wires are terminated at one edge of the board. Orienting the board so that that edge is at the top, from left to right those wires are colored black, brown, red, pink, yellow, green, and blue. Unsolder the wires from the PC board and connect them to the interface as shown in Fig. 2.

Using the interface

Install four D cells in the *Mobile Armatron*. Plug SO1 and the interface board into the 64's user port (rear left of the computer) so that the interface board circuit faces upward. Power up the computer and enter the program shown in the listing; save the program under the name ROBOT.

PARTS LIST

R1-R7—10,000 ohms, 1/4 watt, 5%
 Q1-Q7—2N2222 NPN transistor
 RY1-RY7—reed relay, SPST, 5-volt, 250-ohm coil, Radio-Shack 272-232 or equivalent
 SO1—12-position card-edge socket, 0.156-inch spacing
Miscellaneous: *Mobile Armatron* robot arm (Radio-Shack), PC board, wire, solder, four D cells, etc.
The program shown in Listing 1 and a series of demonstration procedures is available on a Commodore 64 disk for \$6.00 (U.S. funds only) postpaid from B&BTC, RD 1, Box 241H, Tennent Road, Manalapan, NJ 07726. New Jersey residents, please add \$0.36 for sales tax.

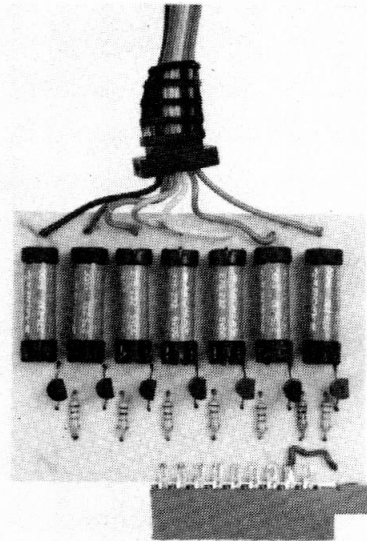


FIG. 3—THE AUTHOR'S PROTOTYPE. Note that it uses a single-sided board (as indicated by the jumper) and that SO1 has been cut down from a larger socket.

When you RUN the program, a menu with five options will appear on the screen. Those options are LEARN, DO, SAVE, RETRIEVE and QUIT (END). To better understand what each of those options do and how the program operates, let's discuss each option separately.

LEARN allows you to "teach" the robot to perform a series of functions in a specific manner. Together those functions form a procedure. Once created, the procedure can be saved, retrieved, added to, and executed at any time. Select LEARN by pressing key F1. The LEARN screen defines the keys to select the twelve possible movement functions. To perform one of the functions described, press the associated key. To end movement, press any key. The program remembers each function selected and the length of time the function is performed. To end the teaching session, press the F1 key to return to the main menu.

DO executes a procedure resident in memory. Select DO by pressing key F3. If a procedure is resident in memory, you will be advised to press any key to begin execution. Otherwise you will be informed that there is no procedure in memory and asked to press any key to return to the main menu.

SAVE allows you to transfer a procedure in memory to a disk file using a name that you specify. Select SAVE by pressing key F5. If there is no procedure in memory, you are so informed and asked to press any key to return to the main menu. If you specify the name of a file that already exists, you are asked if you want to continue. Continuing erases the old disk file and replaces it with the current contents of memory.

RETRIEVE copies the contents of a disk file into memory, erasing any

procedure that is stored in memory at the time. Select `RETRIEVE` by pressing key `F7`. If you specify the name of a file that doesn't exist, you will be so informed and asked to press any key to return to the main menu. If memory already contains a procedure you will be so advised and asked if you want to continue. Continuing erases the current contents of memory and replaces it with the procedure saved on disk.

`QUIT` is selected by pressing both the `CTRL` and `Q` keys. That ends the program, ensures all files are closed, and prints a message "To re-enter, type `GOTO 50`". If you end prematurely without saving a procedure in memory, executing a `GOTO 50` allows you to re-enter the program with memory intact. You can then save memory contents to a disk file.


Tips for experimenters

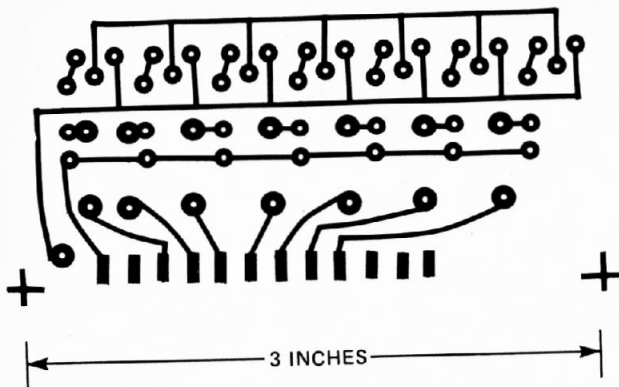
Each movement you specify is called a step. After loading a procedure from disk, you can add steps to it in the `LEARN` mode. That allows you to build a procedure one part at a time, add to it, test it, and save the revised procedure if it proves acceptable. You

can also have standard procedures on file rather than having to build them manually each time.

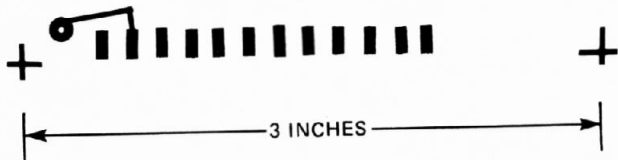
You should always start a procedure with the robot in the same initial position. To do that, save the procedure in memory and enter the `LEARN` mode. Perform the functions necessary to get the robot to the initial position. Then retrieve the procedure you wish to perform.

Each procedure step is saved as a movement-function number and a time count. The movement function is specified by a number between one and twelve (corresponding to the twelve function-definition listings on the `LEARN` screen). The time count is a number that indicates how long the function should be performed. As the robot's batteries drain, the time count may produce slightly different results since the robot will move at a slower speed. Therefore, for precise procedures it is recommended that you always use a fresh set of batteries.

If your procedures will be longer than the currently allowable 250 steps, modify the `DIMENSION` statement in program line 10 to increase the size of the `B` array. 



COMPONENT-SIDE DIRECT-ETCH FOIL PATTERN for the C64 Armatron controller. Don't forget to install the jumper!



SOLDER-SIDE DIRECT-ETCH FOIL PATTERN for the C64 Armatron controller. The story begins on page 144.