16K Extended Color Basic

Armabot

What do you get when you hook a Color Computer up to an Armatron™?

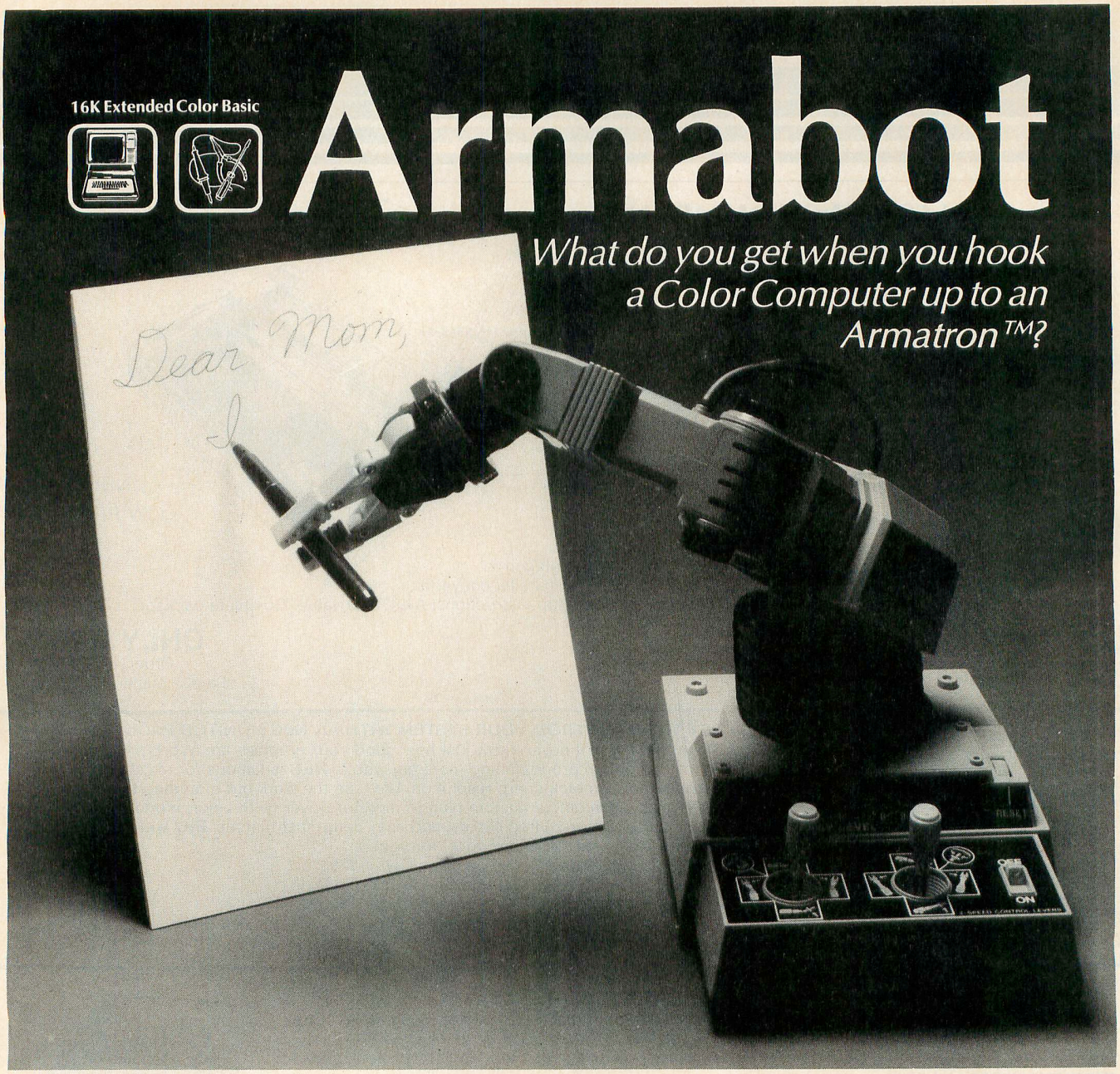


photo by Charley Freiberg

by Steve Cox, with Steve McMaster and Carlos Escobar

IT IS 11:00 P.M. THE NIGHT before the Engineering Week contest, and the motor control circuitry has just been finished and tested. The Armatron is now functional. However, we still need software to control the Armatron via the Color Computer. Judging begins at 9:00, so we have most the night to develop some simple algorithms for control. We made a pot of coffee, ordered a pizza, and forged ahead at a mad pace.

How It all Began

Carlos and other students at Oklahoma State University had organized a group to modify an Armatron robot arm (manufactured by Tomy, sold by Radio Shack, Child World, Toys R Us and other distributors) for a senior design class. He needed me to design an interface to control the Armatron using a microcomputer. I could not pass up this challenge. I promptly called one of my electronics cronies, Steve McMaster, and asked him to work on the project with me. We also decided to use this project for the annual Engineering Week project competition. (Each year during Engineering Week at Oklahoma State, different engineering interest groups enter low-cost — less than $150 — projects designed and built during the semester.) Our engineering group had decided to try to design and build an Armatron-based system with computer control. An article published in *Robotics Age* used solenoids to control the Armatron's joysticks. We believed fitting the Armatron with individual motors for each joint was a better way to go.

Carlos' group worked on the mechanical modifications and Steve and I concentrated on the interface, motor control circuitry, and the necessary software drivers.

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES

PRICE BREAKTHROUGH

Super Sale on New Disk Drives

Starting at **$169.00**

Tandon — Siemens — Remex — MPI — Teac — Shugart — Tabor

40 or 80 Tracks — Single or Dual Head — New 3½" Drivette™
Our Disk Drives Are Capable Of Single **And** Dual Density Operation

The Newest Technology Major Brands Capable Of Operating *On:*
RADIO SHACK[1] — HEATH/ZENITH[2] — APPLE[3]/FRANKLIN[4]
IBM/PC[5]-TEXAS INSTRUMENTS[6] & MOST OTHER COMPUTERS

Drive a Hard Bargain!!™ — 5 M.B.-12 M.B. Hard Drive, Everything You Need
Complete Systems . . . . . from $999.95

Diskette Breakthrough — 10 Pack in Library Case — ~~$18.95~~

NEW LOW PRICE

Since We Are Always Finding Ways To Save You Money,
Please Call For Our Most Current Pricing.

TOLL FREE ORDERING	GENERAL AND TECHNICAL
1-800-343-8841	**1-617-872-9090**

Model I/III/IV Drives (0 1 2 3)	starting at **$169.00**
Color Computer Drive (0 1 2 3)	**$ Call Toll Free**
Apple/Franklin Compatible Add-On Drives with Case & Cables	$219.95
Apple/Franklin Compatible Drives Complete with Controller — Now 40 Track	**$259.95**
Model I/III/IV Memory Upgrade	**$ Call Toll Free**
Printers — Daisywheel/Dot Matrix	**$ Call Toll Free**
Diskettes in Library Cases	~~$18.95~~ New Low Price
Cases and Power Supplies — (Single-Dual-1/2 Height)	starting at $ 44.95
Printer **Buffers** 8K to 512K	starting at $143.95
Percom Double Density Controller (Model I)	**$ Call Toll Free**
Holmes Model I/III Speed-up Mod-VID/80	starting at $90.00
Gold Fingered **Edge Card Extenders**	starting at $13.00
Cables — Printer/Disk Drive	starting at $23.00
DOSPLUS	**$ Special Prices**
Repair Services Now Offered — Fast Turn-a-Round	**$ Call Toll Free**

Warranty on Disk Drives — 6 Months — Extended Warranty $ Call Toll Free

SOFTWARE SUPPORT, INC.

One Edgell Road, Framingham, MA 01701 (617) 872-9090
Hours: Mon. thru Fri. 9:30 am to 5:30 (E.S.T.) Sat. 10 am to 4:30 pm

TERMS:
M.C./Visa/Amex and personal checks accepted at no extra charge.
C.O.D., please add $3.00.
Shipping: Please call for amount.
Not responsible for typographical errors.

Canada
MICRO R.G.S. INC.
751, CARRE VICTORIA, SUITE 403
MONTREAL, QUEBEC, CANADA, H2Y 2J3
Regular **Tel. (514) 845-1534**
Canadian Toll Free 800-361-5155

Dealer inquiries invited.

1 ™ TANDY CORPORATION
2 ™ ZENITH DATA SYSTEMS
3 ™ APPLE COMPUTER CORP.
4 ™ FRANKLIN COMPUTER, INC.
5 ™ IBM CORPORATION
6 ™ TEXAS INSTRUMENTS

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES

☎ Enter 59 on TeleResponse page 159

We had some very specific goals for the working system: the total cost (not including computer and power supplies) must be less than $100; computer control would be manual and automatic; some type of positional feedback must be available for each joint; and we'd like some type of limited vision (photocell).

The first point was to prove that robotics did not have to be as expensive as many people believe. We also wished to demonstrate that another use of the home computer is as a powerful real-time controller. The Color Computer was chosen because of ease of interfacing (all bus lines come out to the game cartridge port) and also because I happened to own one.

We used a 32K Color Computer with Extended Color Basic, although Extended Basic is not necessary. We did not use a disk-based system because we were using the game port for our 6522 interface and the 6522's chip-select is derived from the disk's motor control line — pin 36 of the Color Computer.

Hardware

We had the following hardware criteria:

- Each motor (at each joint) must be able to be switched by software to move in forward or reverse directions.
- Each motor must be able to be turned on or off under software control.
- The speed of each motor should be individually controlled by hardware (as simple as using a speed control potentiometer).
- The transistor switches used for the motors must have a low resistance when on and be capable of sinking at least one ampere.
- The motor control circuitry (steering logic and transistors) must be TTL compatible and as simple as possible (to reduce device count and point-to-point connections).
- Transistors should cost approximately $1 each.
- Flexibility must be designed into the motor controller in case circuit modifications become necessary.

We needed a transistor/logic arrangement which emulated a double-pole, double-throw (dpdt) switch. With the switch in the up position, current flows in one direction (the motor moves in the corresponding direction). The current and motor direction are reversed when the switch is changed to the down position. (As will be seen, Q1 through Q4 and the CMOS NAND gates U11 and U12, Figure 1, provide the dpdt forward/reverse function outlined in goal number one.)

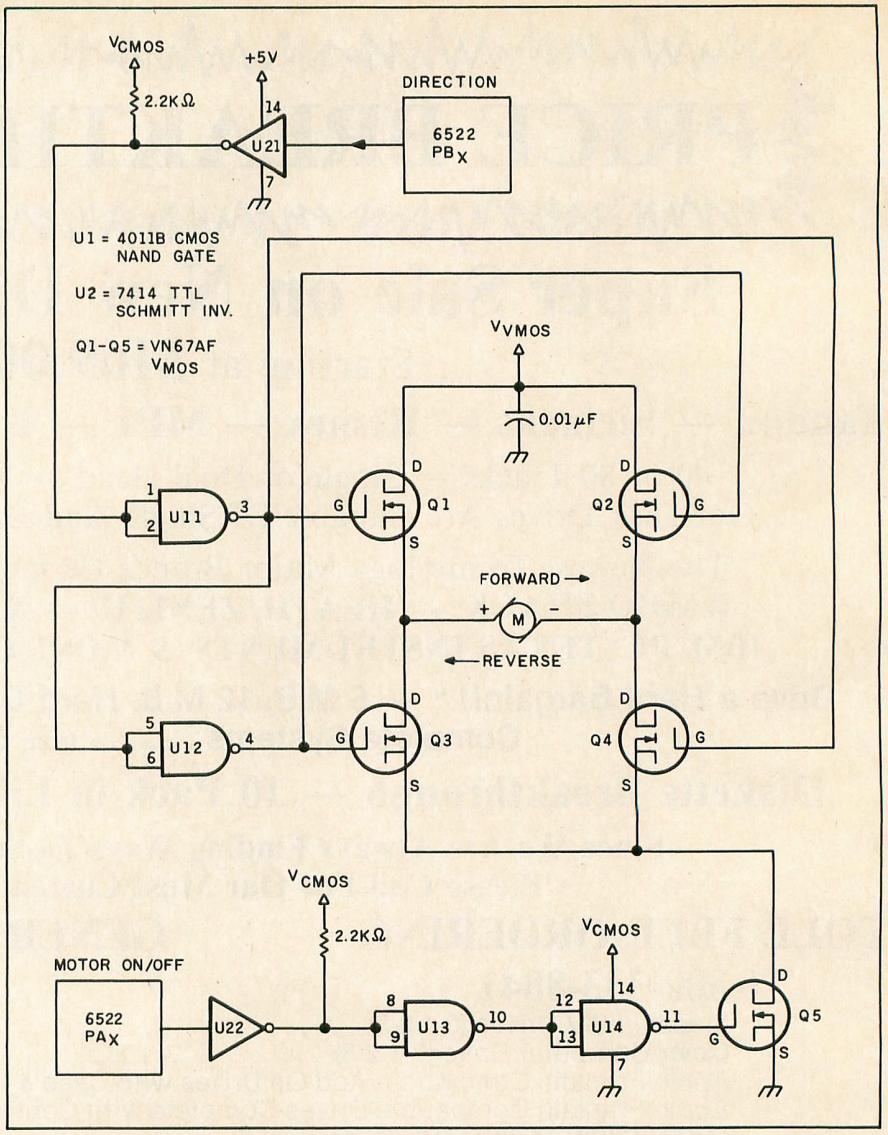


Figure 1. Motor Control Circuit

Transistor Q5 provided the on/off function outlined in goal two. When Q5 was turned on, current began to flow through the motor (M) in Figure 1.

To achieve individual speed control in each motor we had planned to use U13 and U14 as an adjustable astable multivibrator (oscillator) which would Pulse Width Modulate (PWM) the gate of transistor Q5. After bread-boarding a prototype motor controller, we discovered the PWM feature was unnecessary and was not implemented. Also, only one of the CMOS gates (U13 or U14) was necessary, but we continued with both.

Hardware goals four and five were met by using VMOS power transistors. Unlike Bipolar Junction Transistors (BJT) which are current driven, VMOS are voltage controlled. Since they draw negligible gate current (voltage driven), they are readily interfaced with TTL and CMOS logic. Using VMOS will greatly simplify the controller circuitry, since CMOS and TTL cannot directly drive high-current BJT power transistors. The 4000B series CMOS gates were used for their wide operating voltage. The VN67AF VMOS transistor made by Siliconix was used in the motor controller because it is economical ($1.05), it has a relatively low resistance when on, and it can handle over one ampere. There are now much better devices than the VN67AF available at a lower cost.

To semi-isolate the output of the 6522 from the CMOS gates, 7414 Schmitt-Trigger inverters were used; however, a high voltage open collector device such as the 7416 or 7417 should have been used.

To pull the TTL logic levels up to CMOS logic levels a 2.2 kilohm resistor was used. The 2.2 kilohm gave us good switch-

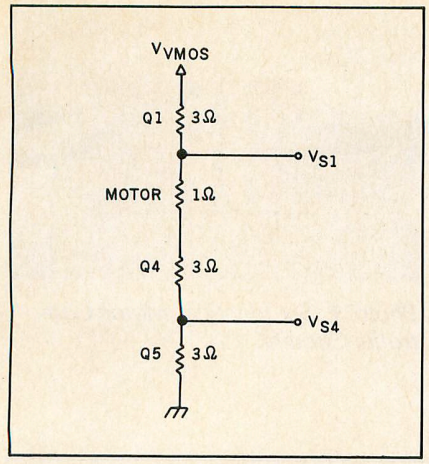


Figure 2. VMOS 3-ohm Series Resistor

ing performance and ensured that the 7414 would never sink more than seven milliamps (its maximum is 16 milliamps).

Please note the three separate power supplies used in Figure 1. Vvmos, Vcmos, and +5V. Vcmos > Vvmos > +5V, with Vcmos=15V, and Vvmos=10V. The Vvmos supply should be capable of at least four amperes due to the heavy current draw if several motors are on at once. Four amperes should allow at least three motors to be on at the same time. The power supplies were turned on in the following order: CMOS; VMOS; and TTL (+5 volt supply). CMOS was turned on first to ensure its input never received a voltage greater than Vcmos. Failure to follow this turn-on procedure produced ruined CMOS gates and strange capacitive storage characteristics from the VMOS transistors.

A Functional Description

We'll look first at the motor control circuitry in Figure 1; the 6522 will be described later.

A logic 1 input to U21 causes the output of U11 to go to +Vcmos turning both Q1 and Q4 on. The output of U12 will be 0V, turning off Q2 and Q3. A logic 0 (0 volts) at the input of U22 makes its output go high (+Vcmos), so U13's output goes low and U14's output will be at +Vcmos. Q5 will turn on as its gate-to-source voltage (Vgs) will be +Vcmos, more than sufficient to be fully saturated. Essentially, the VMOS transistors act as voltage controlled switches with a low series resistance (when on). When the motor is running, each VMOS device appears as a 3-ohm series resistor. This is illustrated in Figure 2.

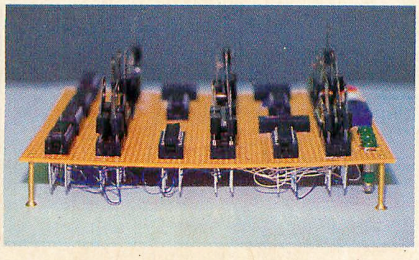


Photo 1. Motor Controller Board

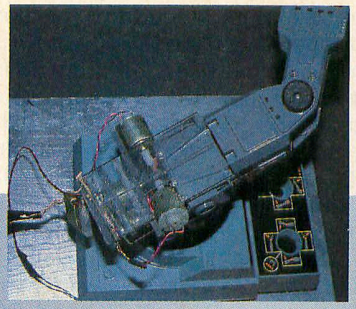
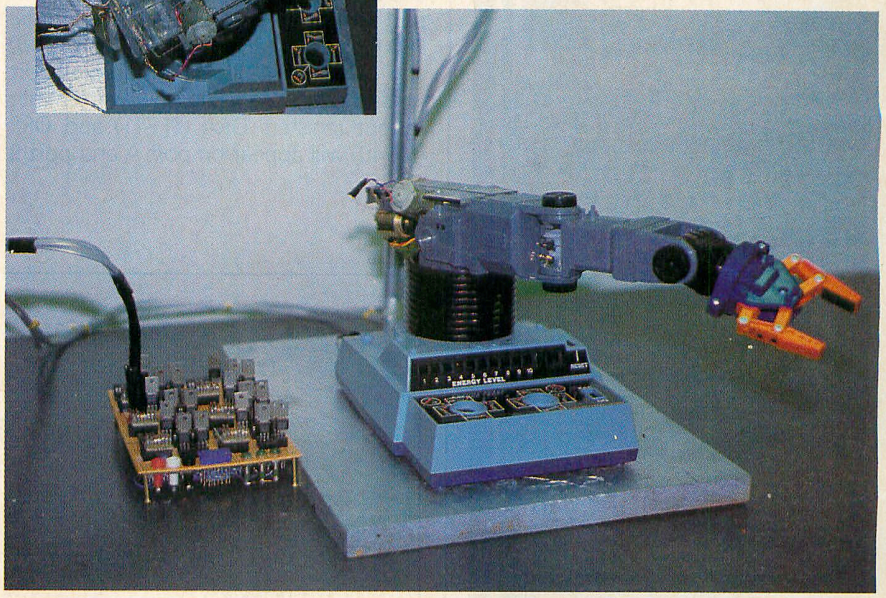


Photo 2. Modified Armatron, with Two Motors Showing

Photo 4. Functioning Arm System, Less Power Supplies, Color Computer and Connecting Cables



Motor Controller

The motor controller (Photo 1) was constructed to accommodate anticipated circuit changes. The VMOS transistors were plugged into 14-pin DIP wire-wrap sockets. This technique made the circuit changes a snap. When a VMOS device is introduced that is pin compatible with the VN67AF, we can upgrade by plugging in the new device. A drawback is the lack of adequate heat-sinking, but this can be overcome with a little ingenuity. (Paper clips work well in a pinch.)

Motors

To implement our plan to use six motors on the arm, one for each axis of rotation, these motors would have to be very inexpensive (dirt-cheap) hobbyist-type dc motors, such as those available at Radio Shack or through hobbyist magazines, if we were to remain in our final price range. The motors on the Armatron were connected to the motor controller via ordinary telephone cord, a special male-female connector (Photo 2, left) and a 16-pin DIP male-female combination (Photo 3, gray cord in the foreground).

6522 VIA

A 6522 VIA (versatile interface adaptor) was used to enable the Color Computer to communicate with the motor controller board. The 6522 has two 8-bit programmable bidirectional (input/output) parallel ports. This gave us 16 independent motor control lines. As the motor controller requires only two control lines for each motor, we needed only twelve control lines, leaving four lines (PA0, PA7, PB0, and PB7) unused. The 6522 was connected to the Color Computer bus as shown in Figure 3. Since I had a 6522 interface built prior to this project, the connections from the 6522 to the motor controller board were made using a 16-pin DIP ribbon cable (gray cable in the background of Photo 3).

Six identical motor controllers are located on the motor controller board (Photo 3). Each motor controller is the equivalent of Figure 1. If we connect the elbow motor controller to the 6522, then PAx in Figure 1 would be connected to PA1 of Figure 3 and PBx of Figure 1 would be connected to PA1 of Figure 3. These connections were made for each

of the six motors, producing a functioning computer controlled arm (Photo 4).

The 6522 must first be properly initialized so it may "talk" with the motor controller. The 6522 is a bidirectional device; the data direction registers (DDRA and DDRB, see Figure 4) for port A and for port B must be programmed as outputs. POKEing 255 ($FF) into DDRA ($FF43) and DDRB ($FF42) will configure all port lines as output. A value now written into output registers ORA ($FF41) and ORB ($FF40) will appear on port A and port B

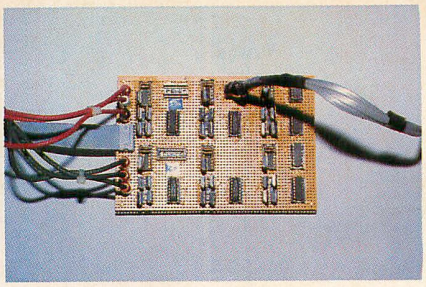


Photo 3. Six Identical Motor Controller Circuits

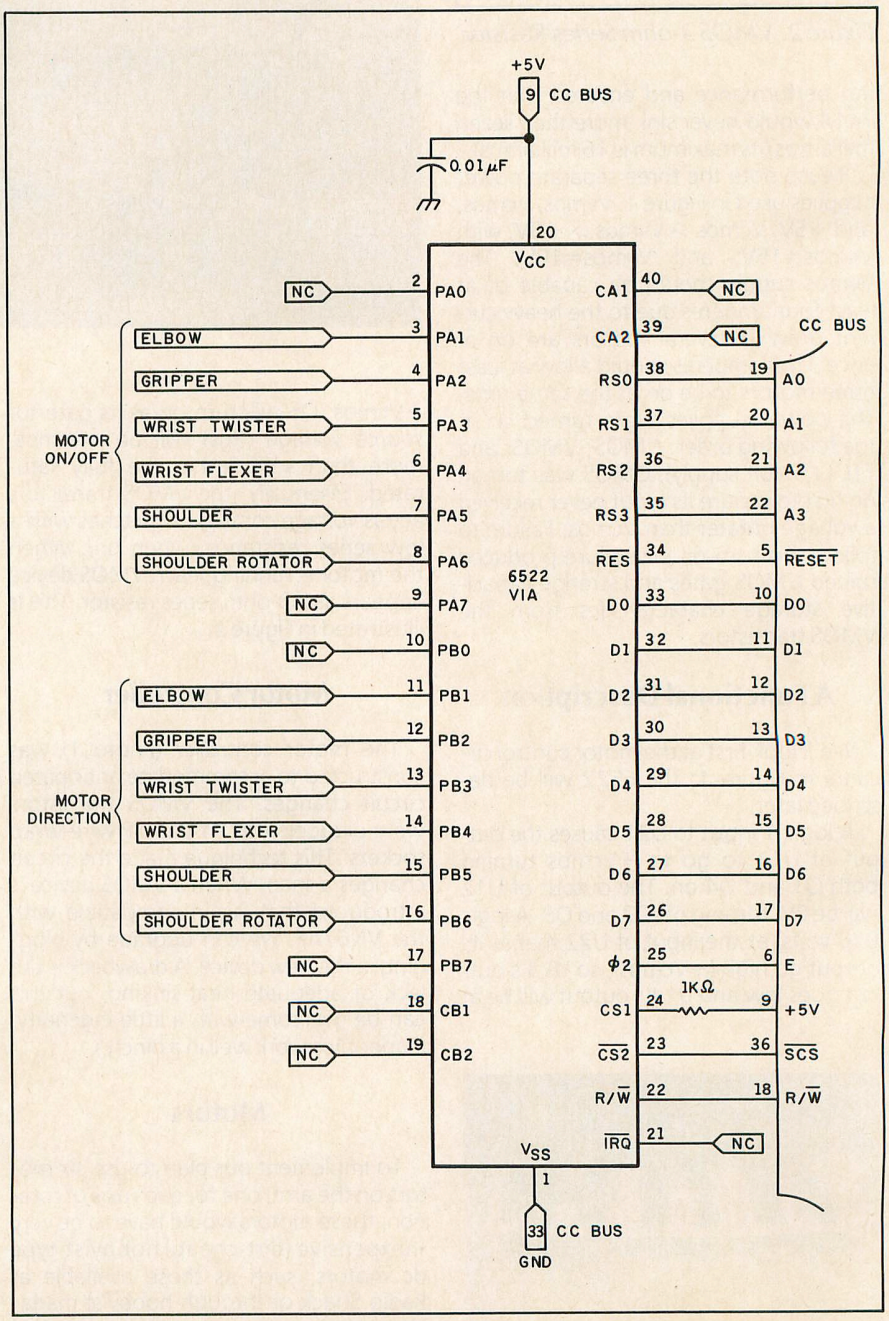


Figure 3. Color Computer to 6522 Connection

Register Number	RS Coding				Register Desig.	Description	
	RS3	RS2	RS1	RS0		Write	Read
0	0	0	0	0	ORB/IRB	Output Register "B"	Input Register "B"
1	0	0	0	1	ORA/IRA	Output Register "A"	Input Register "A"
2	0	0	1	0	DDRB	Data Direction Register "B"	
3	0	0	1	1	DDRA	Data Direction Register "A"	
4	0	1	0	0	T1C-L	T1 Low-Order Latches	T1 Low-Order Counter
5	0	1	0	1	T1C-H	T1 High-Order Counter	
6	0	1	1	0	T1L-L	T1 Low-Order Latches	
7	0	1	1	1	T1L-H	T1 High-Order Latches	
8	1	0	0	0	T2C-L	T2 Low-Order Latches	T2 Low-Order Counter
9	1	0	0	1	T2C-H	T2 High-Order Counter	
10	1	0	1	0	SR	Shift Register	
11	1	0	1	1	ACR	Auxiliary Control Register	
12	1	1	0	0	PCR	Peripheral Control Register	
13	1	1	0	1	IFR	Interrupt Flag Register	
14	1	1	1	0	IER	Interrupt Enable Register	
15	1	1	1	1	ORA/IRA	Same as Reg 1 Except No "Handshake"	

Figure 4. 6522 Information (Reprinted with permission of Synertek Inc, 3001 Stender Way, Santa Clara, CA 95052, copyright 1978.)

Program Listing. Armatron Drive

```
10 'This is the Armatron Drive p
rogram, for communication by 652
2 VIA with the Armatron Motor Co
ntroller
20 '6522 base address is $FF40.
Port A of the 6522 turns the mot
ors on and off. Port B controls
the motors' direction.
25 'References to Left, Right,
Up or Down relative to operator
50 'by Steve Cox, Steve McMaster
 03/29/83     Rev 1: 11/26/83
100 'Initialize DDRB ($FF42) and
 DDRA ($FF43) as outputs
110 POKE &HFF42,255 :POKE &HFF43
,255
130 'All references to (+ to -)
or (- to +) refer to direction o
f current through motor, Fig. 2
140 'Set port B ($FF40) to all m
otors Off
150 'Set port A ($FF41) to all m
otors Forward (+ to -, Fig 2)
160 POKE &HFF40,255 :POKE &HFF41
,255
170 CLS
180 SOUND 5,5
190 PRINT"DO YOU WANT TO OPERATE
 IN THE   MANUAL OR AUTOMATIC MO
DE (M OR A)?"
```

output lines. For example, POKE &HFF40,2 (ORB) will result in PB1=1 (+5 volts) and PB0,PB2–PB7=0 (0 volts). If bit 3 of ORB is set by POKE &HFF40,8, PB3 will equal logic one (+5 volts); however, if bit 3 is cleared (0), PB3 will be logic low (0 volts). The port output lines logic levels (0 or +5 volts) follow their corresponding bits logic values (0 or 1) in ORA and ORB.

Software Development

We decided we would at least need full keyboard control of the Armatron. If we were still awake and had time before the contest, some automatic routines would also be nice.

A rough flowchart for the manual mode is presented in Figure 5. Between the flowchart and the Program Listing, we can fully describe the manual mode.

After initializing the 6522 data direction registers (DDRA and DDRB) as outputs in Line 110, the output port registers (ORA and ORB) are initialized in Line 160 for all motors off and in the forward direction. This direction is in reference to the current flow through the motor in Figure 1 and not a physical direction of the Armatron. In this description, forward indicates motor current flow from plus to minus, with reverse indicating current flow from minus to plus. Physical direction (such as left, right, up or down) is referenced to my position at the Color

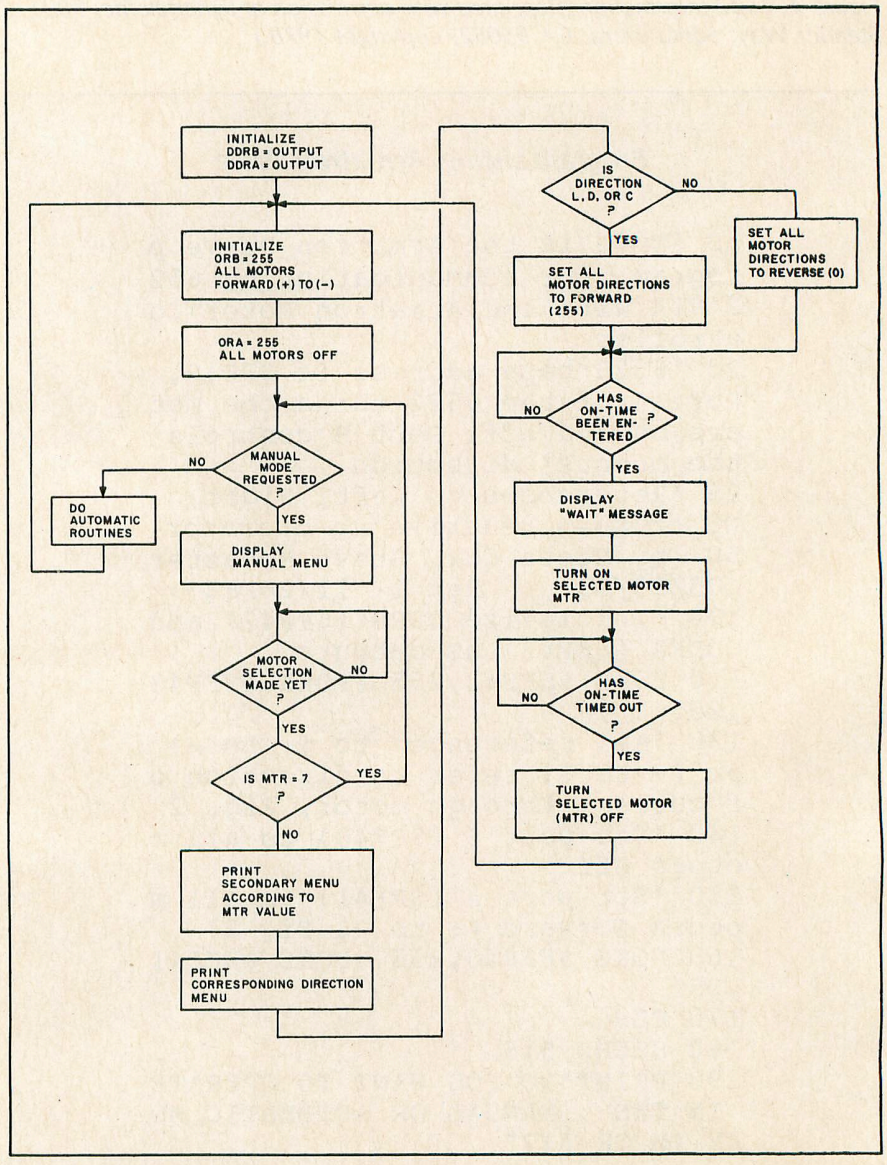


Figure 5. Flowchart For Manual Mode

Computer keyboard. Picture yourself sitting at the keyboard which is situated at a right angle to the rear of the Armatron. As you look around to the Armatron, which faces away from you, your right would also be the Armatron's right. Following the 6522's initialization, the user is queried for the automatic or manual mode (Line 190). Choosing the manual mode displays the manual menu in Lines 300 – 370. This menu describes the various motorized joints of the Armatron. Making a selection will display a secondary menu (Lines 450 – 590) according to the joint chosen. For example, if you choose number one, the elbow, the screen will display:

ELBOW
ENTER DIRECTION
(L OR R)

Entering a direction brings a prompt:

ENTER ON TIME (SECONDS)?

Once the motor direction has been entered, Line 630 checks to see if the directions L (left), D (down), or C (close) have been entered. If so, it sets all motors to forward (relative to Figure 1) even though only one motor is actually going to be turned on. If L, D, or C has not been entered, all motors are set for reverse. The actual direction setting is accomplished in Lines 650 – 690. After entering the motor on time, the Motor On subroutine (1390 – 1440) is called, turning on the selected motor (MTR). A timing loop in Line 750 will wait the number of seconds specified until the Motor Off subroutine (1470 – 1510) is called, disabling the motor's current flow. After the motor is turned off the program returns to the manual menu.

Subroutines

The Motor On and Motor Off subroutines are very important to the manual and automatic modes.

Motor On: Line 1410 reads output register A (ORA) to get the current status of the motors — which motors are on or off. Lines 1420 – 1430 then clear (logic 0) the bit in ORA associated with the motor (MTR variable) to be turned on, but leaves the other bits in ORA unaltered. Only the selected motor (MTR) is turned on while the other motors are unaffected. If motor 1 (MTR=1 is the elbow) is chosen, bit 1 of ORA will be cleared to logic 0, turning on the elbow motor.

Motor Off: This routine is similar to Motor On, except that Lines 1490 – 1500 will set (logic 1) the bit in ORA associated with the variable MTR. Only the selected motor (MTR) will be turned off, leaving the other motors unaffected.

Set Bit For Direction: This subroutine will set the direction bit in ORB as-

```
200 A$=""
210 A$=INKEY$ :IF A$="" THEN210
220 IF A$="A" THEN 1810
230 CLS
250 'Initialize 6522 parameters
ever Manual mode loop.
260 POKE &HFF41,255 :POKE &HFF40
,255
270 'Port A Output ($FF41) Motor
 On/Off, 0=On, 1=Off
280 'Port B Output ($FF40) Motor
 Direction, 1=Forward (+ to -),
0=Reverse (- to +), see Fig 2
290 '***** Manual Menu *****
300 PRINT"ENTER MOTOR KEY NEEDED
"
310 PRINT"1)  ELBOW"
320 PRINT"2)  GRIPPER"
330 PRINT"3)  WRIST-SPIN"
340 PRINT"4)  WRIST"
350 PRINT"5)  SHOULDER"
360 PRINT"6)  SHOULDER-ROTATION"

370 PRINT"7)  RETURN TO MENU"
380 PRINT
390 A$=""
400 A$=INKEY$ :IF A$="" THEN400
410 IF A$="7" THEN 170
420 MTR=VAL(A$)
430 ON MTR GOTO 450, 460, 470, 4
80, 490, 500
440 'Label Secondary Menues
450 PRINT"ELBOW" :GOTO 510
460 PRINT"GRIPPER" :GOTO 510
470 PRINT"WRIST-SPIN" :GOTO 510
480 PRINT"WRIST" :GOTO 510
490 PRINT"SHOULDER" :GOTO 510
500 PRINT"SHOULDER"
510 PRINT"ENTER DIRECTION"
530 ON MTR GOTO 540,550,560,570,
580,590
540 PRINT"(L OR R)" :GOTO 600
550 PRINT"(C OR O)" :GOTO 600
560 PRINT"(S)":GOTO600
570 PRINT"(U OR D)":GOTO600
580 PRINT"(U OR D)":GOTO600
590 PRINT"(L OR R)"
600 C$=""
620 C$=INKEY$ :IF C$="" THEN620
630 A$=C$:IF C$="L" OR C$="D" OR
 C$="C" THEN C$="F" ELSE C$="R"
640 PRINT A$ :PRINT
650 IF C$="F" THEN 690
660 'All motors Reverse
670 POKE &HFF40,0 :GOTO 700
680 'All motors Forward
690 POKE &HFF40,255
```

sociated with MTR. That is, MTR=1 affects ORB bit 1. Setting the direction bit will cause the selected motor's (MTR) current to travel positive and minus (forward) relative to Figure 1.

Clear Bit For Direction: This routine is the same as the Set Bit routine, but the selected motor's direction bit is cleared (logic 0), forcing the motor to turn in reverse (current minus to plus).

Change Direction: This subroutine was written for use in the automatic routines. Lines 1570 – 1580 read ORB and determine which direction the motor (MTR) is turning. If the direction is forward, the direction is changed to reverse, and vice versa. Only the direction bit associated with MTR is affected. This subroutine made moving joints left and then right or up and then down very easy.

Automatic Mode

Since we were running out of time, we decided to try to create a simple routine to demonstrate the movement of each Armatron joint in both directions, with more than one motor on during some points of the routine.

The first automatic routine resides from 870 – 1240. It was developed in panic mode, so it is not very refined. The

```
700 INPUT"ENTER ON-TIME (SECONDS
)";T
710 PRINT:PRINT:PRINT:PRINT"PLEA
SE WAIT WHILE I EXECUTE THE COMM
AND"
720 'Turn MTR motor On
730 GOSUB 1390
740 'Wait w/ motor On T seconds
750 FOR X=1 TO 460*T :NEXT X
760 'Turn MTR motor Off
770 GOSUB 1460
780 'Back to Manual Menu
790 GOTO 230'*End Manual Menu**
820 '1st auto routine developed
3/30/83 @ 5:00 AM by Steve Cox,
Steve McMaster, Carlos Escobar
840 POKE &HFF43,255 :POKE &HFF42
,255
850 I=0
860 ' Delay times for motors
870 T1=.2 :T4=.7 :T5=.15 :T6=.25
 :MTR=2
930 GOSUB 1390
940 MTR=3 :GOSUB 1330
960 GOSUB 1390
970 MTR=1 :GOSUB 1390
980 FOR X=1 TO 460*T1 :NEXT X
```

```
990 GOSUB 1560 :FOR X=1 TO 460*T
1 :NEXT X
1000 MTR=3 :GOSUB 1460
1010 MTR=2 :GOSUB 1460
1020 MTR=1 :GOSUB 1460
1030 MTR=5 :GOSUB 1390
1040 FOR X=1 TO 460*T5 :NEXT X
1050 GOSUB 1560 :FOR X=1 TO 460*
T5*2 :NEXT X
1060 GOSUB 1560
1070 FOR X=1 TO 460*T5 :NEXT X
1080 GOSUB 1460
1090 MTR=4 :GOSUB 1390
1100 MTR=6 :GOSUB 1390
1110 MTR=4 :GOSUB 1560
1120 FOR X=1 TO 460*T6 :NEXT X
1130 GOSUB 1460
1140 MTR=6 :GOSUB 1460
1150 MTR=6 :GOSUB 1560
1160 MTR=6 :GOSUB 1390
1170 FOR X=1 TO 460*T6*2.4 :NEXT X
1180 GOSUB 1560
1190 FOR X=1 TO 460*T6 :NEXT X
1200 GOSUB 1460
1210 MTR=6 :GOSUB1560
1220 GOSUB 1390 :FOR I=1TO460*T6
 :NEXT I :GOSUB 1460
```

more

routines Demo 2 (Line 2540) and Demo-Auto 2 (Line 1810) were developed later. These two routines let the Armatron strut its stuff in the Automatic mode. In the Program Listing, Demo-Auto has been well commented, so the reader may follow each movement.

Since this project did not incorporate any type of positional feedback due to time constraints during development, software timing loops were used throughout the automatic routines. To make the Armatron's joints move to specific positions from an initialized state we used the manual routine and experimented with the motor on times. With these experimentally derived on times for each joint, we knew approximately how long it took each joint to go through a movement from its initial position.

Here is the logic flow for an automatic routine: decide the movements for each joint; initialize the Armatron in a standard position; derive the motor on time for each movement of each joint, using the manual mode; note the on times for the various joints and movements; write an automatic routine using these derived motor on times in conjunction with the three Basic subroutines.

Since the automatic routines rely so heavily on timing loops, they also rely on the Vvmos supply voltage. These auto-

matic routines were written at a Vvmos =10 volts. Changing the Vvmos supply voltage would alter the response of the Armatron during the automatic routines because the Vvmos supply governs the motor current. Altering Vcmos could also affect the timing loops.

Some type of positional feedback for each joint would have been ideal. With feedback we could tell a joint to move a specific number of degrees independent of timing loops, supply voltage, and joint stiffness.

Hardware Improvements

Although the circuit in Figure 1 worked very well, hindsight always reaps circuit improvements. Six of the circuits in Figure 1 were designed and built in two weeks. The motor controller in Photo 1 was the finished result. As expected, problems did arise.

Occasionally, our motor controller would blow out a CMOS NAND gate. I believe this was due to the fact that CMOS logic gates, operating with high supply voltages (around 15 volts), can latch up on switching transients. A latched up gate will draw excessive current from the Vcmos supply through its internal MOS transistors, effectively ruining the logic gate. To counteract this problem, generous use of decoupling capacitors (.01 microFarad) on the Vcmos and Vvmos supply lines should help. More importantly, the Vcmos supply current should be limited by inserting a series resistor of 3 kilohms to 4 kilohms between pin 14 of the CMOS and Vcmos. To slow down switching transients, series damping resistors (5 – 10 kilohm) could be placed between the CMOS gates output and the VMOS transistors' gate. A better idea is to replace the CMOS NAND gates with TTL high-voltage open-collector 7426 NAND gates, with their output pulled up to the voltage necessary to drive the VMOS into saturation.

To protect the VMOS from back-emf generated by turning off the motors, 15-volt zener diodes should be placed on the drains of Q3 and Q4. Transistor Q5 can be eliminated by using a configuration such as Figure 6; however, all four transistors should not be allowed on at the same time (see truth table in Figure 6.)

New Power MOS

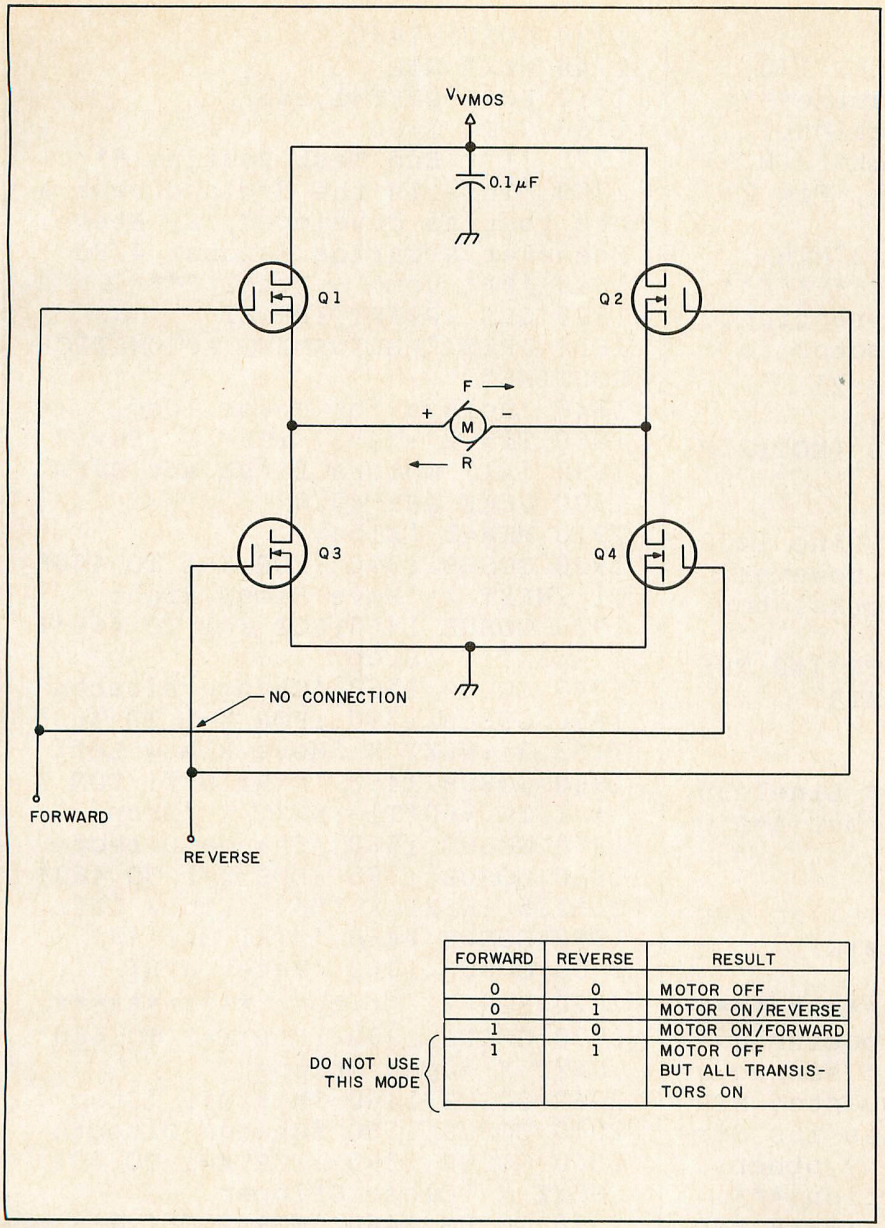
Advances in MOS power devices have brought lower prices and lower conducting resistances. A new device which I would highly recommend (although I have not tried it) is the BUZ71 EconoFET by Siemens. It boasts an on resistance of 100 milliohms, drain-to-source breakdown (Vds) of 50 volts, and drain-to-source current (Ids) of 12 amperes, at a cost of 65¢ to $1. Using these devices instead of the VN67AFs, the Vvmos supply can be lowered by a factor of ten because of their extremely low on resistance. By lowering the Vvmos supply voltage, the voltage applied to the gate of the VMOS devices could also be substantially lowered. The following should explain why.

When Q1-Q4-Q5 were on (see model in Figure 2) the voltage at Vs1 (Q1's source) is approximately 7 volts if Vvmos=10 volts. Recall that VMOS transistors are turned on according to their Vgs. When Q1 is turned on, its gate voltage referenced to ground is at Vcmos (or 15 volts). This gives Q1 a net Vgs of 15V−7V=8 volts. Vgs=8 volts is probably just enough to saturate (fully turn on) Q1. The Vgs of Q4 will be 15V−Vs4 or 15−3V=12 volts: more than enough to saturate Q4.

The Vcmos supply voltage needed to be greater than the Vvmos supply , so the net Vgs was enough to saturate all transistors. However, using devices such as the BUZ71, the voltage drops across each transistor dramatically decreases, lowering the needed gate voltage. This condition holds as long as the current through the devices is low enough to avoid a substantial voltage drop across them.

Some type of positional feedback for each joint of the Armatron would have been ideal. Time permitting, we would have tried to place linearly tapered potentiometers in a voltage-divider configuration on each joint. Then an eight-channel analog-to-digital converter (A/D) could be used to determine positions by reading the wiper voltage of the potentiometers on each joint.

We also planned to implement simple photocells and a parabolic mirror from a cheap flashlight to give the Armatron simple vision. The photocell could be configured in a voltage divider and the voltage could be read by the Color Computer's joystick ports. The joystick ports could also be used to read the potentiometers. Another novel approach to controlling the Armatron using the Color Computer would be by remote control via a modem. With positional feedback this could easily be realized. Using a software package such as REMOTERM — a remote terminal program — and a modem, the Armatron could be controlled from any remote location with another computer/modem or terminal/modem combination. I have used REMOTERM this way with excellent results. Another



FORWARD	REVERSE	RESULT
0	0	MOTOR OFF
0	1	MOTOR ON/REVERSE
1	0	MOTOR ON/FORWARD
1	1	MOTOR OFF BUT ALL TRANSISTORS ON

DO NOT USE THIS MODE

Figure 6. Alternate Motor Control Circuit

possibility is using the joysticks to control the Armatron. By including a Learn mode, the Armatron's movements could be programmed via the joysticks. These movements could then be saved as data on a tape, disk, or EPROM for later execution. The possibilities seem endless.

Our total hardware cost was approximately $96, including the 6522 VIA chip. Our project won second place, missing first place by one point.

This project was not intended to be fully understood by a beginner in electronics and the Color Computer. Nor is this article intended to be a cookbook "how to" article. Robotics using home computers is virtually virgin territory. This field is begging for your skill and insight.

So what are you waiting for? Go for it!

■ ■ ■

Contributors

Dr. Joe Cleveland
Professor Gerald McClain
Dr. Larry Jones
Dr. Neal Willison
Dr. Perry McNeill
Bill Newberry
Preston Barber
Bob Bosselman
SEET membership

On Motors

by Carlos Escobar

THE IDEA OF INTERFACING an inexpensive toy to a computer to produce a working robot came after studying an Armatron. We found many problems after studying the complexity of the mechanisms used to operate the Armatron's six functions; many hours were dedicated to find a way to interface the Color Computer to the model. The solution we hit on was to use independent actuated motors to perform each of the functions.

We then had to find the best location for each motor, the speed at which the motor should run, and finally, the torque required to perform each individual function. Lack of physical space in the arm, and difficulty in getting to the drive shafts and gears, made this one of the most delicate tasks.

The location of the motors was found by determining the place on the mechanism where high speed of rotation (RPM) and low torque were available to perform each operation. Our findings presented some problems: space and accessibility to the specified shafts and gears were limited to these locations. Also, the inaccessibility of small high-torque motors was a set-back.

Deciding to use inexpensive, low torque, high RPM motors was inevitable. The motors used were similar to the one originally located in the Armatron — 300 mAmps, 3 Vdc, 8000 RPM: and cost $2.50 each.

The first motor was mounted inside the base. This motor activates the gearing system which rotates the arm. Motor number two was located inside the shoulder. The function of this motor is to move the arm up or down. The four remaining motors were placed at the back of the arm. These last four motors actuate the hand rotation, the gripper, the wrist, and the elbow.

Many modifications were made to accommodate these last four motors: existing shafts were replaced with longer shafts; and new gears were added to reduce the RPM and increase the torque of these motors.

■ ■ ■

```
1230 I=I+1
1240 IF I<6 THEN 870 ELSE 170
1250 '*End 1st auto routine****
1270 'Set Bit for Direction...
This causes selected motor (MTR)
 to run Forward (+ to -, Fig 2)
1290 P= PEEK( &HFF40 )
1300 POKE &HFF40, P OR 2^MTR
1310 RETURN'******************
1330 ' Clear Bit for Direction..
 This causes selected motor to r
un Reverse (- to +, Fig 2)
1350 PP= PEEK( &HFF40 )
1360 POKE &HFF40,PP AND (NOT(2^M
TR))
1370 RETURN
1390 '*** Motor On *** Line 1420
 was used, works fine; however,
A=A AND (NOT(2^MTR)) works, too
1410 A= PEEK( &HFF41 )
1420 IF (A AND INT(2^MTR))=0 THE
N 1430 ELSE A=A-INT(2^MTR)
1430 POKE &HFF41,A
1440 RETURN
1460 '*** Motor Off *** Line1490
 was used, works fine; but A=A O
R (2^MTR) work, too
1480 A= PEEK( &HFF41 )
1490 IF (A AND INT(2^MTR))>0 THE
N 1500 ELSE A=A+INT(2^MTR)
1500 POKE &HFF41,A
1510 RETURN'******************
1530 'This direction routine use
d by automatic mode. It senses w
hich direction the requested MTR
 is going, then reverses its dir
ection w/o affecting any other.
1560 '*** Change Direction ***
1570 B= PEEK( &HFF40 ) :BB=B AND
 (INT(2^MTR)) :IF BB=0 THEN B=B+
INT(2^MTR) ELSE B=B-INT(2^MTR)
1580 POKE &HFF40,B :RETURN'****
1600 '@@Test Routine. Used durin
g hardware debug. Sequences each
 motor On, waits .4 secs., turns
 Off, changes direction, turns O
n, waits, turns off, next motor
1630 T=.4
1640 FOR MTR=1 TO 6
1650 PRINT"MOTOR NUM=";MTR
1660 INPUT DUM
1670 GOSUB 1390
1680 FOR I=1 TO 460*T :NEXT I
1690 GOSUB 1460
1700 GOSUB 1560
1710 GOSUB 1390
1720 FOR I=1 TO 460*T :NEXT I
1730 GOSUB 1460
1740 NEXT MTR
1750 POKE &HFF41,255
1760 GOTO 1600
1770 '*** End Test Routine ***
1790 'This is the 3rd and best a
uto routine developed. By Steve
McMaster & Carlos Escobar 4/83
1810 '*** Demo - Auto2 ***
1820 CLS :PRINT :PRINT
1830 PRINT"PERFORMING AUTOMATIC
ROUTINE"
1840 'Delays for Motor loop
1850 T1=.2 :T4=.7 :T5=.1 :T6=.2
1890 'All motors first set Rvrs
1900 POKE &HFF40,0
1910 MTR=1 'Elbow
1920 GOSUB 1390 :FOR X=1 TO 460*
T1 :NEXT X 'Move Elbow Right
1930 GOSUB 1460:FOR X=1 TO 460*T
1 :NEXT X 'Stop
1940 GOSUB 1560 'Change Directn
1950 GOSUB 1390 :FOR X=1 TO 460*
T1*3.3 :NEXT X 'Move Elbow Left
1960 GOSUB 1460 'MTR1 Off: FOR X
 = 1 TO 460*T1 :NEXT X 'Stop
1970 GOSUB 1560 'Change Directn
1980 GOSUB 1390 :FOR X=1 TO 460*
T1*1.5 :NEXT X 'Move Elbow Left
1990 GOSUB 1460 'MTR1 Off ***
2000 GOSUB 1560 'Reset MTR1 Dir
2020 MTR=2 'Gripper ***********
2030 GOSUB 1390 :FOR X=1 TO 460
:NEXT X 'Open Gripper
2040 GOSUB 1460 'MTR Off 'Stop
2050 GOSUB 1560 'Change Directn
2060 GOSUB 1390 :FOR X=1 TO 800
:NEXT X 'Close Gripper
2070 GOSUB 1460 'MTR 2 Off ***
2090 MTR=3 'Wrist (Twist) *****
2100 GOSUB 1390 :FOR X=1 TO 460
:NEXT X 'Twist Wrist
2110 GOSUB 1460 'MTR 3 Off'Stop
2120 FOR X= 1 TO 460 :NEXT X
2140 MTR=4 'Wrist (Up/Down) ***
2150 GOSUB 1390 :FOR X=1 TO 460*
T4 :NEXT X 'Move Wrist Up
2160 GOSUB 1460 'MTR 4 Off'Stop
2170 FOR X= 1 TO 460 :NEXT X
2180 GOSUB 1560 'Change Directn
2190 GOSUB 1390 :FOR X=1 TO 460*
T4*2.0 :NEXT X 'Move Wrist Down
2200 GOSUB 1460 'MTR 4 Off'Stop
2210 FOR X= 1 TO 460 :NEXT X
2220 GOSUB 1560 'Change Directn
2230 GOSUB 1390 :FOR X=1 TO 460*
T4 :NEXT X 'Move Wrist Up
```