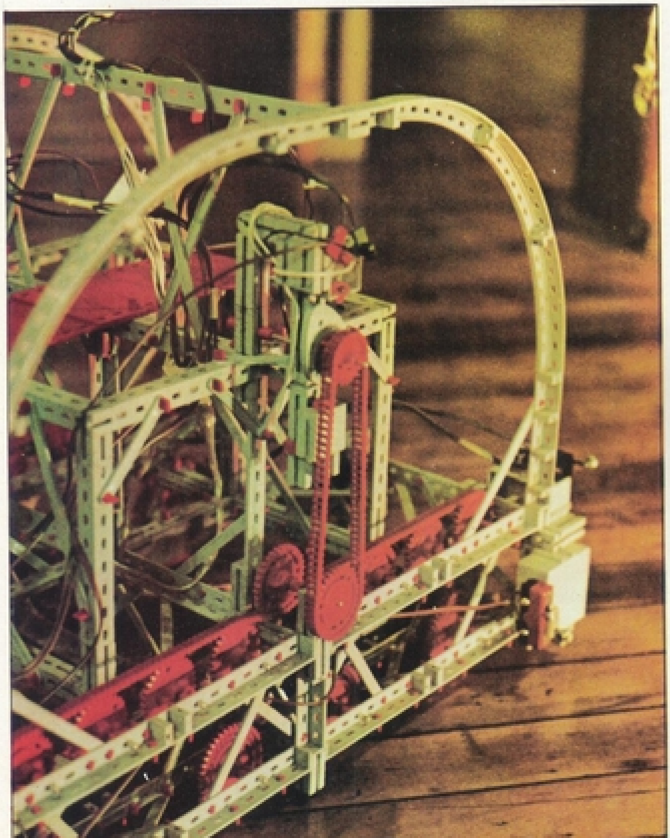
# Building Your BASIC Robot

BY SAM NEWHOUSE

Robots vary in sophistication and control structure, from Mars-walking space probes and industrial assembly robots to self-motivating household pets and computer-controlled "turtle" drawing robots.

My interest in these simple mobile computer input-output devices, robots, led me to construct my own "mechanical pet", Bert. Bert can move forward or backward, and pivot in either direction. Through a separate computer-controlled monitoring system, he responds to a pre-programmed direction after bumping into something and records his steps on film through a mounted video camera. A distance sensor controlled by the same computer keeps track of the distance he travels and guides him on a pre-programmed path.



Bert stops on command for any set length of time, and on command resumes his program where he left off. My reason for building this "robot" was, from the beginning, to construct a peripheral for my SWPC 6800 computer. My goal has been achieved – Bert is controlled entirely by SWPC 8K BASIC, version 2.0 And although there are speed limitations with BASIC, careful programming and reasonable compromise allow for the convenience of BASIC real-time control.

Bert is constructed almost totally of replaceable, interchangeable plastic parts manufactured by the fischertechnik Company of West Germany. Although expensive, the building material allows for quick translation of ideas into physical reality, and easy modification of an existing design.

Struts of various sizes, nylon/plastic equivalents of I-beams, rivets which attach struts to I-beams or other struts, plastic chain, gears, metal axles, cotter pins, motors with reduction gears and worm gears make up the building materials. Flexible skin-like material can also be attached to I-beams with rivets.

These materials result in structures that are light but strong. Bert can move 50 lbs. with only minor reduction in speed – Bert only weighs about 10 lbs.

Eight parallel fischertechnik low-voltage motors drive the robot (4 motors drive each of the two tracks). The motors are driven by regulated adjustable power supplies. The power now in use travels to Bert via cables at about 1.5 amps per side at 12 volts. A third power supply drives the CRC express serial relay interface and delivers .5 amps at 5 volts.

The CRC interface connects to the control computer via a serial output port. By sending ASCII characters to the interface, followed by an execute character, the relays change position, and power is supplied to specific motors in the robot.

For example, by sending the ASCII sequence "A!E!C!G!" to the interface, both tracks will move forward. The A!E! commands the two direction relays to supply forward voltage. The C!G! sets both sides to "on". A forward motion is accomplished simply by the program statement: print "A! E!C!G!".

At present, the robot may be programmed under BASIC to move forward or backward, or to rotate left or right. During rotation, one track moves forward, while the other is in reverse. During a continuous rotation the robot does not move from its initial position more than a few inches.

Inputs to the control computer consist of 5 bits of a parallel input port. These consist of 4 bits, 1 each, from the front, back, left, and right sensor arrays, and 1 bit for distance travelled detection.

A gear rides on the chain which forms the tread of the robot's track. This gear drives another chain which in turn drives a rotating elliptical gear. At the top and bottom of its rotation, the elliptical gear strikes two micro switches which are wired in parallel. Therefore, each rotation of the gear sends two pulses to the computer port.

The 6800 MPU uses what is called memory mapped input/output. This means that an input port, to the computer, appears as a memory location, just like any other location. The five inputs are, in effect, wired right into the memory so that whenever you want to "scan" this port, all you do is take a "PEEK" at its location, using BASIC.

The PEEK is a BASIC language function, such that PEEK (J) will return the decimal value of the contents of memory location J. To read the input port you just use the PEEK statement. No handshakes or interrupts are necessary.

A tight BASIC wait loop counts the distance pulses by constantly "PEEKING" the location where the input data appears. Since BASIC is slow, and these pulses are of short duration, a certain percentage of distance pulses are missed.

The same routine that counts distance pulses also constantly checks for a sensor closure — an interruption in the robot's control program that occurs when these "sensors", which are microswitches mounted on the front, back, and sides of the robot, are closed.

**This self taught tinkerer built his own robot and programmed it using BASIC.**

The robot then moves in a direction opposite to the original direction, upon hitting an obstruction.

The inputs are arranged so that the distance pulse (I0) represents the least significant bit. Front sensors are I1, back are I2, left I3 and right I4.

At any time, "PEEKING" the input port gives a number (decimal) between 0 and 255. If no sensors show a "hit", and if no distance pulse is present, the port will show $255_{10}$. However, if any sensor shows a hit, then the input port will show a number less than 254, regardless of the status of the distance pulse.

The input routine first looks for a sensor hit, then a distance pulse, and finally, for a second sensor hit. Any time a sensor hit registers, the computer goes into its interrupt subroutine, which stops the motors, and then reverses them for a fixed period of time. During this brief time the robot is not responsive to further sensor hits.

The BASIC control program recognizes the following commands: FX, BX, RX, LX, SX, and JX. The robot moves forward, backward, left, or right X number of pulses by FX, BX, LX, RX respectively. SX stops the robot for a time period proportional to X. If X = 0, control is immediately returned to the options menu portion of the program. JX loops control to Step Number X. This is usually used to repeat a pattern over and over by "jumping" back to the initial program step repeatedly.

The control program will also allow robot control patterns of up to 50 steps to be saved and loaded from paper tape. A list of the current program may also be obtained.

The control program is written in a top-down, structured programming style, and is easy to modify. If the robot were programmed in assembly or machine language, its control would be much more precise — no distance pulses would be lost because of the slow speed of BASIC.

Tests have shown that the control program and robot are precise enough (as currently developed) to bring the robot to the same spot after executing a short stored program, plus or minus about 1 foot. This means that, if the robot is placed in a particular spot, pointing in a known direction, execution of a stored program will leave it no farther than about 1 foot from where the same stored program drove the robot previously.

Another application of the robot is to carry a standard video camera. In this mode, the robot can be directly controlled by a SWTPC joystick. Again, a program coded in BASIC is used to interface the joystick to the CRC serial

## Technical Specs on Robot Package

| | |
|---|---|
| **H x W x L -** | 18" x 19" x 26" |
| **Weight -** | 25 lbs. including camera |
| **Speed -** | 1.5 mph at 12 volts |
| **Motors -** | 8 – fischertechnik motors with gearing |
| **Power -** | 2 – 1.5 ampere variable voltage regulated power supplies for motors |
| **Supplies -** | 1 – 1.5 ampere 5 volt power supply for relays |
| **Camera -** | Sony AVC3200 video camera |
| **Computer -** | SWTPC6800 – 16K memory |
| **Software -** | SWTPC 8K BASIC, Version 2-0 |
| **Peripherals -** | Slot 0 – serial port to relay interface |
| | Slot 1 – serial control terminal |
| | Slot 3 – parallel interface to joystick |
| | Slot 5 – parallel interface to sensors and distance pulses |
| | Slot 6 – serial interface for Teletype |
| **Sensors -** | 3 – forward – wired in parallel |
| | 2 – right " |
| | 2 – left " |
| | 5 – rear " |
| **Relays** | CRC Xpres serial control interface – contains 4 SPST relays |

interface which, in turn, controls Bert's motors.

The human operator sits in front of a TV monitor which shows what the robot-mounted TV camera is pointing at. By moving the joystick gently forward, the robot will respond by moving forward. Move the stick back to center, and the robot stops. Move the stick right or left, and the robot rotates right or left.

A signal transmitted by the camera goes through to a video recorder before it reaches the monitor. A scene could be set up and videotaped by the moving camera. By going forward or backward, slow zoom-ins are possible.

At this time, the camera cannot tilt up or down. As it approaches an article on the floor, the object disappears from the camera's view.

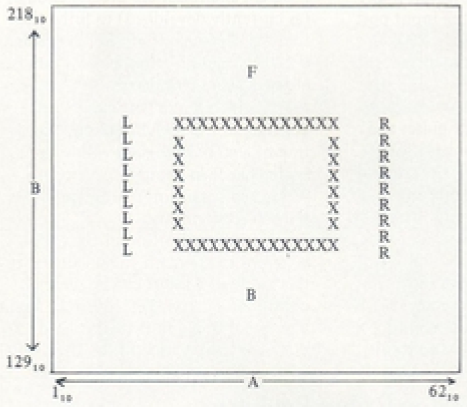
My plans for the future include constructing a "camera tilter" from fischertechnik parts, proximity sensors, an on-board computer and a radio link to the SWTPC computer.

The onboard computer will follow its own built-in program, responding to sensor hits and counting distance pulses. The radio link will enable the robot to carry out instructions sent out by the remote master computer. These instructions would come in a general rather than specific form; examples might include patrol, sleep, wait (until

## How the Robot Program Works

When you have constructed your robot, you can get him moving with these BASIC programs. First, though, you must understand the CRC Xpres Relay Interface before you can comprehend Program I. This board contains four relays and

### Screen Diagram



**Within the square outlined by the X's, the robot will stop. The cursor must be moved close to the "F" or "B" for motion to begin. With "L" and "R", it is sufficient for the cursor to be behind the line of L's or R's, and not above or below the ending L's or R's.**

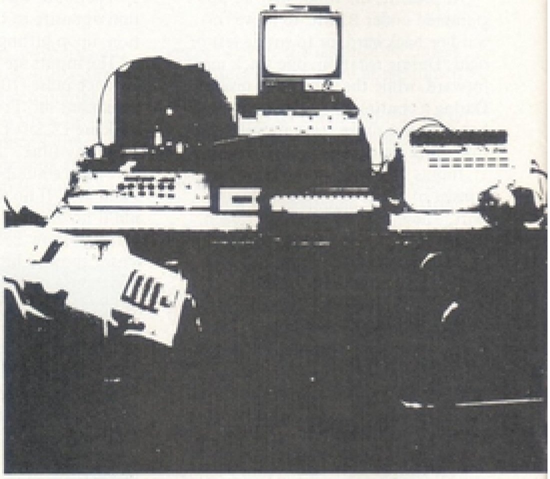
connects to a serial port at Slot Ø. Any relay may be turned off, one at a time, as follows:

| | On | Off |
|---|---|---|
| Relay 1 | A | B |
| Relay 2 | C | C |
| Relay 3 | E | F |
| Relay 4 | G | H |

The execute character to the relay board performs the most recent command received. The execute character is the exclamation point (!). Sending "E!" to Port 0 would turn on Relay 3.

Relay 1 controls a DPDT power-relay to the right side. This controls the motors forward or backward.

Relay 2 supplies power to Relay 1.



Relay 3 activates a DPDT reversing relay that controls the left side.

Relay 4 supplies power to Relay 3.

Thus, PRINT #0, "A!E!C!G!" would turn on all relays and the robot would march forward.

The commands implemented so far which control the robot motors are:

| Meaning | Forward | # | Comment |
|---|---|---|---|
| Forward | F | X | Where X is the # of the pulses |
| Backward | B | X | " |
| Left | L | X | " |
| Right | R | X | " |
| Jump | J | X | Where X is the control statement |
| Stop | S | X | Where X>0 stop for X time periods |
| or | S | 0 | Return to options menu |
| Recorder On | V | X | Where X can be anything |
| or Off | X | X | " |

Another relay activates the video recorder. This relay functions through the SWTPC AC-30 Cassette Interface Board's "Cassette Motor Control Output". Sending the appropriate control character to the CT-1024 terminal, opens or closes the relay. Control T turns on the relay, Control R causes it to turn off.

motion detector is activated) or program.

The program instruction conditions the on-board computer to accept a series of specific motion control commands from the SWTPC computer, store these commands, and execute them until an override command is received via the radio link. Examples of specific motion control commands are: forward 10 pulses, right 5 pulses, stop 3 time periods.

The radio link will be made from a Citizens Band Class "C" tone transmitter and receiver. This transmitter will be modulated, or turned on and off, by the SWTPC 6800. All output will appear in the form of serial ASCII characters. At the receiver, a waveform corresponding to the original ASCII character will be produced, which will be fed through a level converter to the serial input port of the on-board computer.

Miniature lead-acid batteries will provide power to the motors. Two batteries will power each track (see photos). The on-board computer will be powered by 7 amp/hr Gould nicads.

What the purpose of all this is, I'm not sure. I had a lot of fun building and designing my robot and I plan to go a lot further. My fascination with robots will continue unabated. This simple project opened my eyes to the incredible complications that can pop up when you try to design a robot. ▶◀

## Program I

```
0001  F4=32790
0002  F5=254
0003  F1=1
0004  F0=0
0005 DIM A$(50),P(50)
0006  F2=2
0007  F3=3
0008  F9=4
0009  F8=2
0010 PRINT #1,CHR$(16);CHR$(22);
0020 PRINT #1,"       OPTIONS"
0030 PRINT #1,"1 - PROGRAM"
0040 PRINT #1,"2 - RUN"
0050 PRINT #1,"3 - SAVE"
0060 PRINT #1,"4 - LOAD"
0070 PRINT #1,"5 - END"
0075 PRINT #1,"6 - LIST"
0078  F2=6
0080 PRINT #1
0090 PORT= 1
0100 INPUT "#",C
0105 IF C>6 THEN 10
0110 IF C<1 THEN 10
0115  C=INT(C)
0120 ON C GOSUP 1000,2000,3000,4000,5000,6000
0130 GOTO 10
1000 REM *** PROGRAM SECTION
1010 PRINT #1,CHR$(16);CHR$(22);"MAXIMUM
     NUMPER OF STEPS IS 50"
1020 PRINT #1,"MAXIMUM DIRECTION IS 50"
1022 PRINT #1,"ENTER '#' WHEN PROGRAM DONE"
1030  I=1
1035 PRINT #1,"STEP #";I;
1040 INPUT A$(I)
1050 IF A$(I)="#" THEN RETURN
1060 INPUT "DISTANCE",P(I)
1070 IF P(I)>50 THEN 1060
1090  I=I+1
1100 GOTO 1035
2000 REM *** RUN SECTION
2010  I=F1
2020  A$(I)=LEFT$(A$(I),1)
2025 IF A$(I)="F" THEN 2200
2030 IF A$(I)="P" THEN 2300
2040 IF A$(I)="L" THEN 2400
2050 IF A$(I)="R" THEN 2500
2060 IF A$(I)="S" THEN 2600
2070 IF A$(I)="J" THEN 2700
2075 IF A$(I)="V" THEN PRINT"";:I=I+1:GOTO2020
2077 IF A$(I)="X" THEN PRINT"";:I=I+1:GOTO2020
2085 PRINT #1," *** ERROR AT STEP #";I
2090  I=I+1
2100 GOTO 2020
2200 REM *** F ROUTINE
2210  T=F0
2215  D=F1
2217 PRINT #0
2220 PRINT #0,"A!E!C!G!A!E!C!G!A!E!C!G!";
2230  Q=PEEK(F4)
2235 IF Q<F5 THEN GOSUP 7000
2240 IF Q<>F5 THEN 2230
2245 IF Q<F5 THEN GOSUP 7000
2250  T=T+F1
2260 IF T>P(I) THEN 2270
2265 GOTO 2230
2270  I=I+F1
2280 GOTO 2020
2300 REM *** PACKWARD ROUTINE
2310  T=F0
2315  D=F2
2317 PRINT #0
2320 PRINT #0,"P!F!C!G!P!F!C!G!P!F!C!G!";
2330  Q=PEEK(F4)
2335 IF Q<F5 THEN GOSUP 7000
2340 IF Q<>F5 THEN 2330
2345 IF Q<F5 THEN GOSUP 7000
2350  T=T+F1
2360 IF T>P(I) THEN 2370
2365 GOTO 2330
2370  I=I+F1
2380 GOTO 2020
2400 REM *** LEFT ROUTINE
2410  T=F0
2415  D=F3
2417 PRINT #0
2420 PRINT #0,"P!E!C!G!P!E!C!G!P!E!C!G!";
2430  Q=PEEK(F4)
2435 IF Q<F5 THEN GOSUP 7000
2440 IF Q<>F5 THEN 2430
2445 IF Q<F5 THEN GOSUP 7000
2450  T=T+F1
2460 IF T>P(I) THEN 2470
2465 GOTO 2530
2470  I=I+F1
2480 GOTO 2020
2500 REM *** RIGHT ROUTINE
2510  T=F0
2515  D=F9
2517 PRINT #0
2520 PRINT #0,"A!F!C!G!A!F!C!G!A!F!C!G!";
2530  Q=PEEK(F4)
2535 IF Q<F5 THEN GOSUP 7000
2540 IF Q<>F5 THEN 2530
2545 IF Q<F5 THEN GOSUP 7000
2550  T=T+F1
2560 IF T>P(I) THEN 2570
2565 GOTO 2530
2570  I=I+F1
2580 GOTO 2020
```

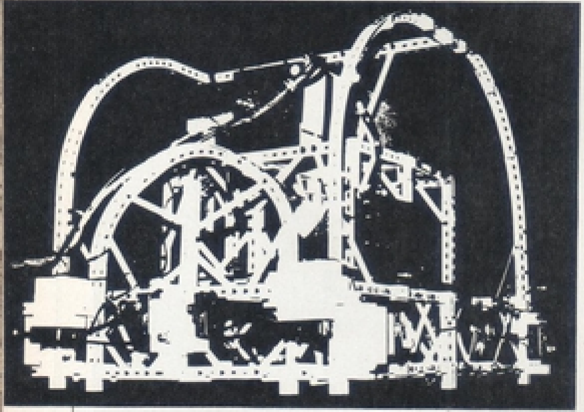
## Program I - continued

```
2600 REM *** STOP ROUTINE
2610 PRINT #0,"D!H!";
2611 IF B(I)=0 THEN RETURN
2612 FOR Q6=1 TO B(I)*100
2613 NEXT Q6
2614  I=I+1
2620 GOTO 2020
2700 REM *** JUMP ROUTINE
2710  I=B(I)
2720 GOTO 2020
3000 PRINT #1," TURN TTY ON"
3010 INPUT " IS IT ON",G$
3020 IF LEFT$(G$,1)="Y" THEN 3040
3030 GOTO 3000
3040 PORT= P2
3050 INPUT " RANGE (1,R)",R
3060 PRINT "   TURN PUNCH ON"
3075 LINE= 240
3080 FOR I=1 TO20
3085 PRINT #P2,CHR$(255);
3090 NEXT I
3105 PRINT R
3107 FOR Q8=1 TO 10:PRINT CHR$(255);:NEXT Q8
3110 FOR I=1 TO R
3112 FOR Q8=1 TO 10:PRINT CHR$(255);:NEXT Q8
3115 PRINT A$(I);",";B(I)
3120 NEXT I
3130 FOR I=1 TO 20
3140 PRINT #P2,CHR$(255);
3150 NEXT I
3160 PORT= 1
3170 PRINT "SAVE DONE"
3180 RETURN
4000 PRINT "TURN TTY ON"
4010 INPUT "IS TTY ON",G$
4020 IF LEFT$(G$,1)<>"Y" THEN 4010
4030 PORT= P2
4040 PRINT "  PUT TAPE IN READER"
4050 PRINT "  TURN READER ON"
4060 INPUT R
4090 FOR I=1 TO R
4100 INPUT A$(I),B(I)
4110 NEXT I
4120 PORT= 1
4125 PRINT "  LOAD FINISHED"
4130 RETURN
5000 END
6000 PRINT #1,CHR$(16);CHR$(22);"LIST"
6010 PRINT #1
6020 INPUT "RANGE (R1,R2)",R1,R2
6030 FOR I=R1TOR2
6040 PRINT #P2,"  STEP #";I;"
     ";A$(I);"    ";B(I)
6050 NEXT I
6060 PRINT #1," LIST DONE"
6070 RETURN
7000 PRINT #0,"D!H!";
7002 FOR D9=1 TO 100
7003 NEXT D9
7010 ON D GOSUB 7300,7200,7500,7400
7020 FOR G4=1 TO 200
7030 NEXT G4
7095  T=B(I)
7100 RETURN
7200 PRINT #0,"A!E!C!G!A!E!C!G!A!E!C!G!";
7210 RETURN
7300 PRINT #0,"B!F!C!G!B!F!C!G!B!F!C!G!";
7310 RETURN
7400 PRINT #0,"B!E!C!G!";
7410 RETURN
7500 PRINT #0,"A!F!C!G!";
7510 RETURN
9900 PRINT PEEK(32790)
9910 GOTO 9900
```

## Comments on Program I

**Program Line 1–9** - Initialize constants, DIM statement. 32790 in Statement 1 is the sensor input location in decimal notation.

**10–100** - This clears the CT-1024 terminal's screen and prints an options menu. Then it requests your choice. Line 78 sets the Teletype$^{TM}$ to Port 6. Line 90 sets the control terminal to the SWTPC CT-1024.



**105–130** - This section checks the input for error and jumps to the appropriate routine through line 120.

**1000–1100** - This section allows the user to input a program, a step at a time. See the command chart. When "#" is entered, the options menu is again displayed.

**2000–2100** - This section runs the stored program. The robot performs each programmed step in order. Line 1085 prints out an error message if the command is invalid. Execution continues.

**2200–2280** - This routine drives the robot forward B(I) spaces. It counts the pulses coming from the distance sensors, continually checking for any microswitch sensor that has reached an obstruction.

"T" is first set to 0 in line 2210. "T" is the counter of pulses from the distance sensor. "D" is then set to 1. "D" denotes the current direction. Line 2217 prints a carriage return/line feed to the relay interface port which resets the current character location points in BASIC. This avoids insertion of unwanted CR&LF's. Line 2220 activates the motors so that the robot goes forward. Line 2230 inputs the sensors into variable "Q". Line 2235 checks to see if a microswitch is closed indicating an obstruction. Line 2240 checks to see if a distance pulse is present. If not the program goes to Line 2230 and again inputs the sensors. Line 2245 check for sensor closure. Line 2250 increments the distance pulse count. Line 2260 compares the new total to the prescribed amount. This checks the forward movement. Line 2265 re-reads sensors at line 2230 if the count has not reached the programmed distance. Line 2270 increments the step counter. Line 2280 returns to command decoder (Line 2020).

**2300–2380** - The backward routine – duplicates lines 2200–2280 although line 2320 is different.

**2400–2480** - The left routine – same as above.

**2500–2580** - The right routine – same as above.

**2600–2620** - The stop routine – if B(I)=0, return to options menu. If not, wait B(I) time periods.

**2700–2720** - Jump routine – go to control statement #B(I).

**3000–3180** - This routine saves a control program on paper tape. Line 3107 prints nulls which are necessary for timing. Line 3112 does the same thing. Line 3115 outputs a set of values.

**4000–4130** - The LOAD routine loads a paper tape generated by the SAVE routine. Line 4060 gets the number of steps to come from the tape.

**6000–6070** - The LIST routine is used to list the stored program.

**7000–7510** - This routine is called whenever a sensor closure is detected. The routine stops the robot and waits for a while, (lines 7000-7003). Then Line 7010 directs the program to a line which makes the robot motors avoid the obstruction that has been detected. "D" is a variable whose value can be 1, 2, 3, or 4. 1 means forward; 2, back; 3, left; 4, right. Line 7010 thus selects proper directions for the robot. Lines 7020–7030 cause a fixed delay. Line 7095 should be "T=B(I)". This, in effect, causes the robot to skip the step involving the obstruction.

**9900–9910** - A utility routine for printing sensor inputs in decimal on the terminal.

## Program II

```
0001 GOSUP 5000
0002  P2=3
0005 GOSUB 100
0010  A=PEEK(32782)
0015 IF A>62 THEN 10
0016  P=PEEK(32782)
0017 IF P<129THEN16
0030 GOSUP 1000
0035 GOSUB 500
0038 GOSUP 8000
0040 GOTO 10
0100 FOR H=0 TO 63
0110 FOR V=128 TO 223
0120 PRINT #P2
0130 PRINT #P2,CHR$(H);CHR$(V);
0140 NEXT V
0150 NEXT H
0155 RETURN
0500 REM
0525 PRINT #P2,CHR$(A);CHR$(P);
0540 RETURN
1000 PRINT #P2,CHR$(A+64);CHR$(P);
1035 GOSUP 500
1040 RETURN
1050 GOTO 10
5000 PRINT #1,CHR$(16);CHR$(22);
5010 PRINT #1,"                    F"
5020 PRINT #1:PRINT #1:PRINT #1
5030 PRINT #1,"          ****************"
5040 FOR Z9=1 TO 6
5050 PRINT #1, "L            *
     *          R"
5060 NEXT Z9
5070 PRINT #1,"          ****************"
5080 PRINT #1:PRINT #1:PRINT #1:PRINT #1,"
     P";
5095 PRINT #1,CHR$(16);
5100 RETURN
8000 IF A>45 THEN 8050
8005 IF A<18 THEN 8050
8010 IF B>145 THEN 8050
8015 IF B<120 THEN 8050
8020 REM *** FORWARD MOVEMENT
8025 GOSUP 8500:PRINT #0,"A!E!C!G!";:RETURN
8050 IF A>45 THEN 0
8055 IF A <18 THEN 8100
8060 IF B<190 THEN 8100
8065 REM *** PACKWARDS MOVEMENT
8070 GOSUP 8500 :PRINT #0,"P!F!C!G!";:RETURN
8100 IF A>10 THEN 8150
8105 IF P<160 THEN 8150
8110 IF P>180 THEN 8150
8115 REM *** LEFT HARD TURN
8120 GOSUP 8500:PRINT #0,"A!F!C!G!";:RETURN
8150 IF A<55 THEN 8200
8155 IF P<160 THEN 8200
8159 IF B>180 THEN 8200
8165 REM *** RIGHT HARD TURN
8170 GOSUB 8500:PRINT #0,"P!E!C!G!";:RETURN
8200 REM *** STOP
8205 PRINT #0,"D!H!";:PRINT #1,"";:RETURN
8500 PRINT "";
8505 RETURN
```

## Comments on Program II

This program uses the SWTPC Joystick and SWTPC GT-6144 Graphics Display. A map (see diagram) is put on the video screen showing front, back, right and left. A blinking cursor moves around the screen as the Joystick is moved. When the cursor moves near one of the letters F, B, L, or R on the screen, the robot moves in that direction. The screen now changes from computer-generated images to video signals from the robot's camera. When the stick is again centered, the robot stops, and the map reappears.

To understand the comments on the program which follows, you must understand the two peripheral devices.

The Joystick is interfaced to the input of a SWTPC parallel I/O port. The stick moves in two dimensions. The onboard analog to digital converter transforms the stick motion into output values. "A" is the horizontal stick position value, and "B" is the vertical position value. Joystick continually outputs "ABABABABAB – " at a rate of fifty times a second. "A" varies from 1 to 62 (decimal) while "B" varies from 218 to 129. Thus, A=1, B=129, would be the lower left corner; A=62, B=129, would be the lower right corner; A=62, B=218, would be the upper right corner; and A=1 B=218, would be the upper left corner.

The GT-6144 Graphics Device displays rectangular dots on the video screen. These dots overlay at the same time, as characters from the CT-1024. The graphics display is a grid