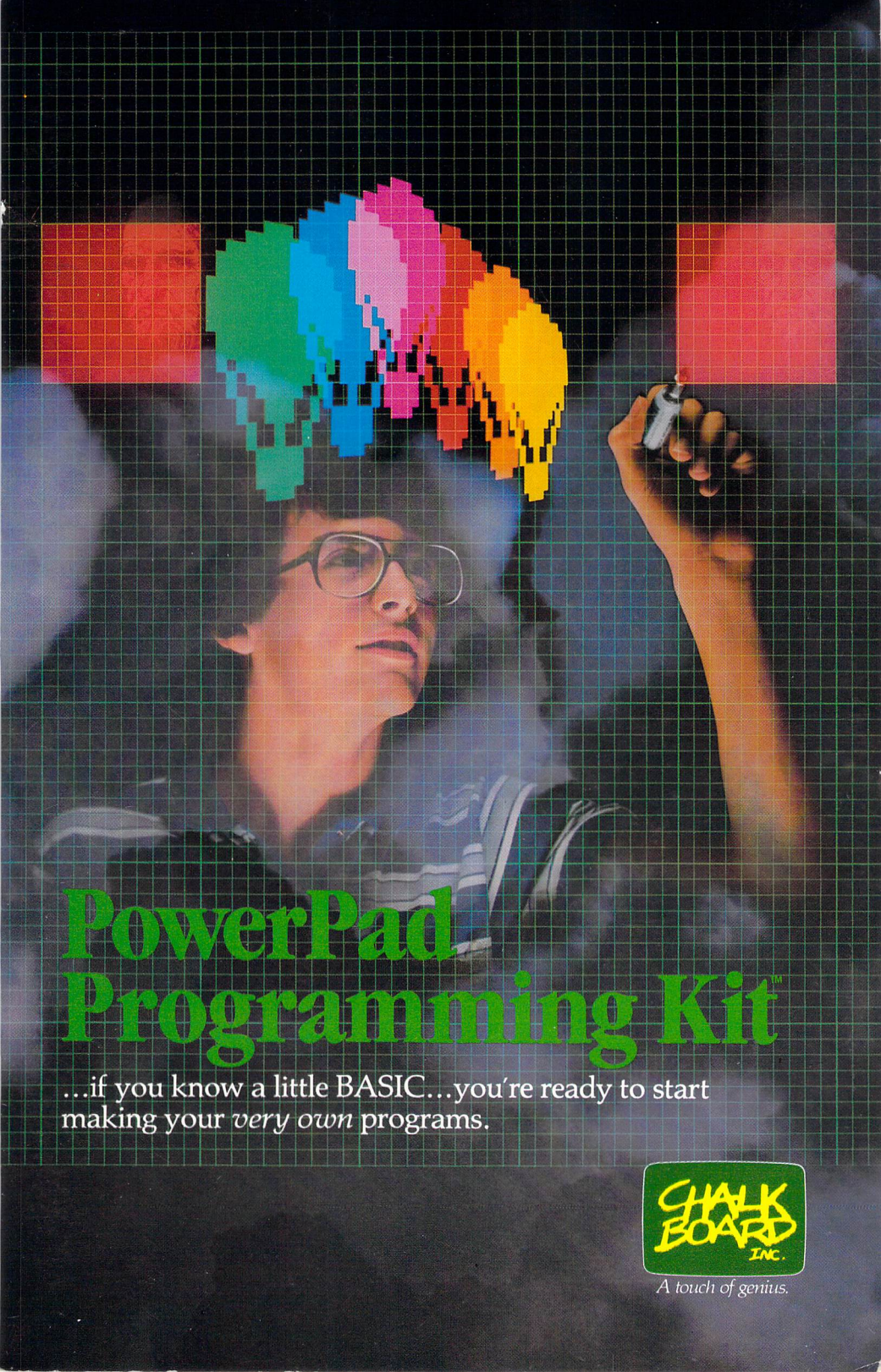

PowerPad™
Programming Kit
...if you know a little BASIC...you're ready to start making your *very own* programs.
CHALK BOARD INC.
A touch of genius.

Version 2.2/830824

PowerPad Programming Kit™

User's Guide For Commodore 64

Chalk Board, Inc.
3772 Pleasantdale Road
Atlanta, Georgia 30340
(404) 496-0101
(800) 241-3989 (outside Georgia)

Table of Contents

Table of Figures

Table of Programs

INTRODUCTION

The Chalk Board PowerPad offers an alternative to the typewriter-style keyboard in working with your home computer. For many people this development makes contact with the computer less frustrating and more fun.

But for those who have mastered the computer keyboard and are familiar with the BASIC programming language, PowerPad offers some exciting and powerful new dimensions to writing and executing programs. While you learn to write programs for PowerPad, you gain a deeper working knowledge of your own home computer.

The information in this manual explains and demonstrates how to give commands to PowerPad in Commodore 64 BASIC. It provides a series of activities which will familiarize you with PowerPad's programming. Lastly, there are some challenging questions for you to consider on your way to more sophisticated programming.

If you are unfamiliar with the operation of the Chalk Board PowerPad, be sure to refer to the PowerPad User's Guide, which accompanied your PowerPad. This guide contains instructions concerning how to connect the pad to your Commodore 64 home computer system. Find the section of the directions which illustrates use with the Commodore 64 computer.

Getting Started

- Do not try to begin without reading the PowerPad User's Guide.
- Refer to the section which illustrates the use of PowerPad with Commodore 64 home computers.
- Insert the overlay included with this product by gently guiding the overlay's frame into the groove around PowerPad's work surface.

Important Note:

Do not plug in PowerPad until you have entered, saved and run a program.

When you plug PowerPad into the Commodore 64's control port 1, the pad communicates with the computer along some of the same circuit lines as the keyboard. Consequently, whenever PowerPad is plugged into the port and you type a key on the keyboard, the computer receives invalid data from both the pad and the keyboard. However, no electrical damage can occur.

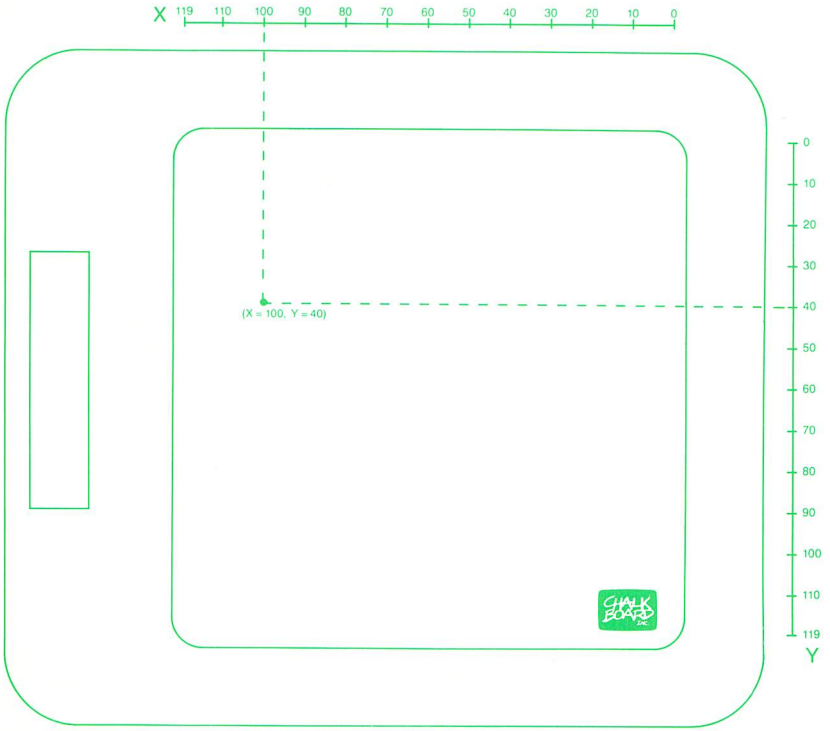
All of the sample programs in this manual do watch for the space bar to halt execution. Once any of these programs is running, touch no key other than the space bar. Also, be sure that the SHIFT LOCK key is in the OFF (up) position.

To use any PowerPad program in BASIC with the Commodore 64, you **first** type RUN, then press return and **then** plug in the pad.

PowerPad Test in BASIC

The Chalk Board PowerPad is a pressure-sensitive input device. It senses the touch of a finger (or any blunt stylus) and provides the location of the point of contact. The location point is given in the form of X and Y coordinates (similar to those found on the Cartesian Coordinate System). Figure 1 shows how the coordinates are arranged on the PowerPad's surface. The values on each axis range from 0 to 119. There are 120 rows by 120 columns, or 14,400 different possible points which PowerPad can sense. Figure 1 also shows a sample point and the X and Y values that would be returned by touching the pad at that point.

Figure 1. PowerPad Coordinate Arrangement and Sample Point



PowerPad sends the X and Y coordinates to your computer in a serial format. This means that the values for X and Y are converted to binary numbers and are sent one bit at a time. To read the X and Y values, the computer must read the first bit, request that the next bit be sent, read that bit, and so on, until all of the bits have been read. The computer must then reassemble those bits.

There are a total of 16 bits. The first two have no meaning and should be ignored. The following seven are for the Y value, and the last seven are for the X value.

Program 1, "PowerPad Test," uses subroutines to scan the pad and print to the screen the X and Y coordinates of the point of contact. The subroutines can be called by any BASIC program to communicate with PowerPad.

The first subroutine (lines 10000–10070) initializes the variables used by the other subroutines. It also tells you to plug in PowerPad.

The second subroutine (lines 11000-11250) sets up control port 1 to talk to PowerPad and tells PowerPad to begin scanning for points being touched. The fifth time that this subroutine is called, the message "UNABLE TO TALK TO POWERPAD. DID YOU PLUG PAD INTO CONTROL PORT 1?" is printed. The subroutine then falls through to a section which restores proper communication between the computer and its keyboard. This section **must** be executed before the program halts. Failure to do so may leave the keyboard unreadable, even if you unplug PowerPad.

The third subroutine (lines 12000-12090) determines whether PowerPad or the space bar has been touched. If PowerPad has been touched, it is ready to send an X and Y coordinate, and the subroutine returns with the variable SENSE equal to zero. If the space bar has been pressed, the message "PROGRAM STOPPED" is printed and the exit routine executed. Otherwise the subroutine continues to wait for you to touch

the pad or space bar. If the counter C reaches the value 30, then PowerPad is not talking to your computer and the subroutine jumps to the initialization subroutine (INIT PAD).

The fourth subroutine (lines 13000-13070) reads the values of the X and Y coordinate from PowerPad. It is called after the third subroutine has determined that you have touched the pad. The fourth subroutine calls the fifth subroutine twice: first to read in Y, and again to read in X. The fourth subroutine then tells the pad to resume scanning for another touch. It returns with the variables X and Y set to the position of the point of touch.

The fifth subroutine (lines 14000-14090) reads the value of one number (either X or Y) from PowerPad. It is called only by the fourth subroutine. It is this subroutine which assembles the binary bits coming from PowerPad back into a whole number. The value of the number read is returned in the variable BYTE.

A flow chart (Figure 2) which depicts schematically the BASIC PowerPad Test program follows the listing of Program 1.

Suggestions:

Chalk Board has taken great care to insure that this and subsequent programs have been printed correctly and that they run properly. Should you have any difficulty in getting any of the programs in this manual to run properly, check your entries to be sure that you have typed each line correctly. Once you have typed your program and are sure that it runs properly, save it on a diskette or cassette to simplify using the program in the future. Refer to the User's Guide which came with your Commodore 64 computer for instructions concerning saving programs.

To help you keep track of the programs you create, three PowerLog™ cards are included in this package. Be sure to label your cassette tapes or diskettes carefully and clearly. Use the PowerLog cards to record the identity and location of your own programs.

As you type program 1, as well as all of the other example programs in this manual, you can save time by leaving out the REM statements.

Program 1: PowerPad Test (all BASIC).

```
1 REM ******** POWERPAD TEST ********
2 REM POWERPAD DEMONSTRATION PROGRAM
3 REM IN BASIC FOR THE COMMODORE 64.
4 REM NOTE: USE THE SUBROUTINES
5 REM       IN LINES 10000-14090
6 REM       FOR YOUR OWN PROGRAMS
7 REM    (YOU MAY LEAVE OUT ALL REMS)
8 REM CHALK BOARD, INC. 1983
9 REM ******************************
100 PRINT CHR$(147): REM CLEAR SCREEN
110 PRINT "POWERPAD TEST"
120 GOSUB 10010: REM INIT VARS
130 GOSUB 11010: REM INIT PAD
140 GOSUB 12010: REM CHECK IF POINT/SPACE PRESSED
150 GOSUB 13010: REM GET X AND Y
200 REM USE X AND Y
210 IF X=0 AND Y=0 THEN 140
220 PRINT "X=";X;"  Y=";Y
230 GOTO 140
240 REM ******************************
10000 REM INIT VARIABLES
10010 PRINT "PLUG POWERPAD INTO CONTROL PORT 1."
10020 BDDR=56323: REM $DC03 = CIA DDRB
10030 BPR =56321: REM $DC01 = CIA PRB
10040 ADDR=56322: REM $DC02 = CIA DDRA
10050 APR =56320: REM $DC00 = CIA PRA
10060 IC=0: REM COUNTS # OF CALLS OF INIT PAD
10070 RETURN
10080 REM ****************************
11000 REM INIT CONTROL PORT AND POWERPAD
11010 IC=IC+1: IF IC=5 THEN 11110
11020 POKE ADDR, 0: REM SET UP PRA TO WATCH SPACE
11030 POKE BDDR,22: REM " CTRL PORT 1 TO TALK TO PAD
11040 POKE BPR,  0: REM ZERO CLOCK AND CLEAR LINES
11050 POKE BPR,  2: REM PULSE CLEAR LINE
11060 POKE BPR,  0: REM (BEGIN SCANNING)
11070 RETURN
11080 REM ==========
11100 REM INIT HAS BEEN CALLED 5 TIMES
```

```
11110 PRINT "UNABLE TO TALK TO POWERPAD.  DID"
11120 PRINT "YOU PLUG PAD INTO CONTROL PORT 1?"
11130 REM ==========
11200 REM RE-ENABLE KEYBOARD
11210 PRINT "UNPLUG POWERPAD BEFORE TYPING."
11220 POKE ADDR,255: REM RESTORE DDRA
11230 POKE APR, 255: REM RESTORE PRA
11240 POKE BDDR,  0: REM RESTORE DDRB
11250 END
11260 REM ****************************
12000 REM CHECK IF POINT OR SPACE BAR PRESSED
12010 C=0: REM COUNTER
12020 SPACE=(PEEK(APR)AND128)/128: REM CHECK SPACE
12030 IF SPACE=1 THEN 12050: REM SPACE NOT PRESSED
12040 PRINT "PROGRAM STOPPED.": GOTO 11210
12050 SENSE=(PEEK(BPR)AND8)/8: REM CHECK SENSE LINE
12060 IF SENSE=1 THEN 12080: REM POINT NOT FOUND
12070 IC=0: RETURN: REM POINT FOUND
12080 C=C+1: IF C<30 THEN 12020: REM 30 TRIES
12090 GOSUB 11010: REM CALL INIT AGAIN
12100 GOTO 12010
12110 REM ****************************
13000 REM GET X AND Y
13010 POKE BPR,4: REM PULSE CLOCK LINE
13020 POKE BPR,0: REM (SHIFT OFF FIRST BIT)
13030 GOSUB 14010: Y=BYTE: REM GETBYTE
13040 GOSUB 14010: X=BYTE: REM GETBYTE
13050 POKE BPR,2: REM PULSE CLEAR LINE
13060 POKE BPR,0: REM (RESUME SCANNING)
13070 RETURN
13080 REM ****************************
14000 REM GET BYTE
14010 BYTE=0
14020 FOR L=1 TO 7: REM READ AND COMBINE 7 BITS
14030 POKE BPR,4: REM PULSE CLOCK LINE
14040 POKE BPR,0: REM (SHIFT OFF FIRST BIT)
14050 DTA=PEEK(BPR) AND 1: REM READ DATA LINE
14060 IF DTA=0 THEN BYTE=BYTE+128: REM BUILD VALUE
14070 BYTE=BYTE/2: REM                 IN BYTE
14080 NEXT L
14090 RETURN
```

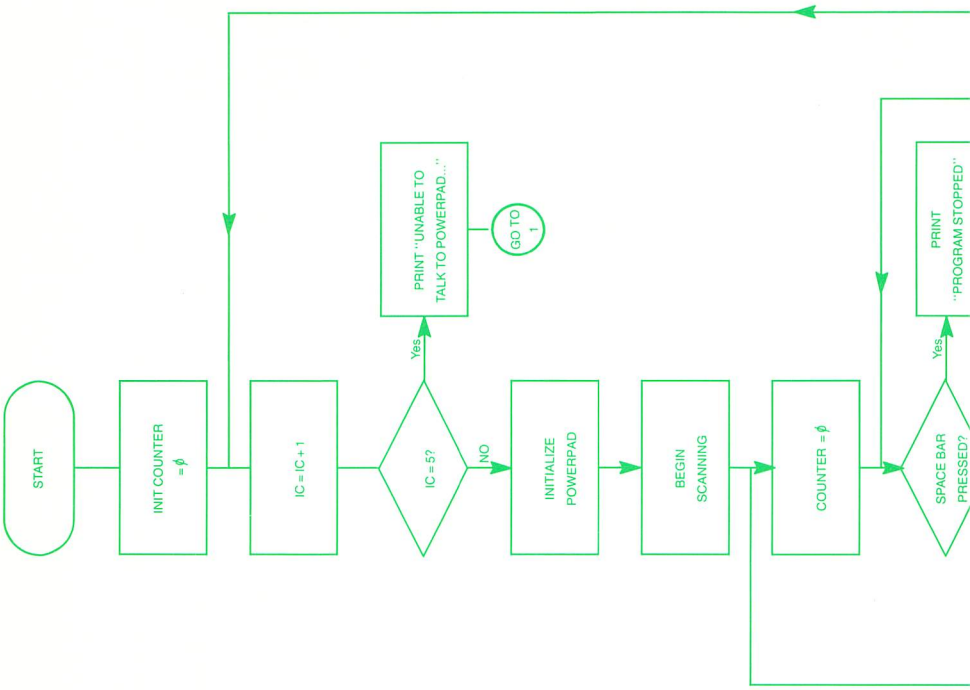


Figure 2. Flowchart of PowerPad Test Program

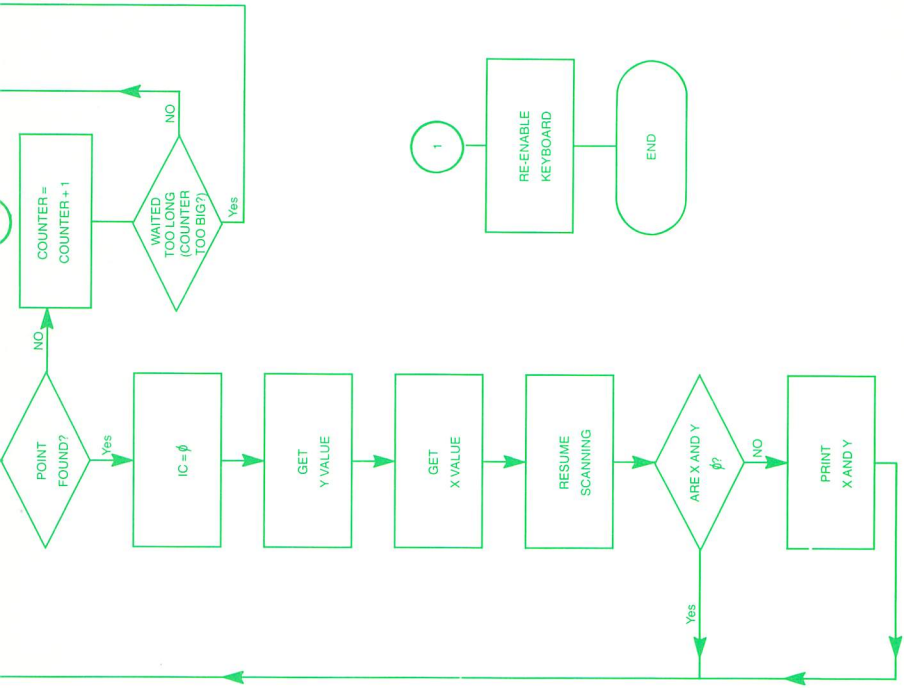
How PowerPad Works

This section is designed for the experienced programmer who wants to know more about how PowerPad communicates with the Commodore 64 computer. The machine language version of the PowerPad Test program appears at the end of this section. You need not understand machine language to take advantage of this faster-running program.

PowerPad's Interface Lines and Shift Register:

PowerPad uses four lines to interface to the Commodore 64. The CLOCK and CLEAR lines are outputs from the computer to PowerPad. The DATA and SENSE lines are outputs from PowerPad to the computer.

The CLEAR line is used to tell PowerPad to scan for a touch point. It must be pulsed high



(brought from low to high and back to low again) to begin the scan.

The SENSE line is PowerPad's signal that a touch has occurred. This line is normally high. When you touch the pad, the line goes low. It remains low until the CLEAR line is pulsed.

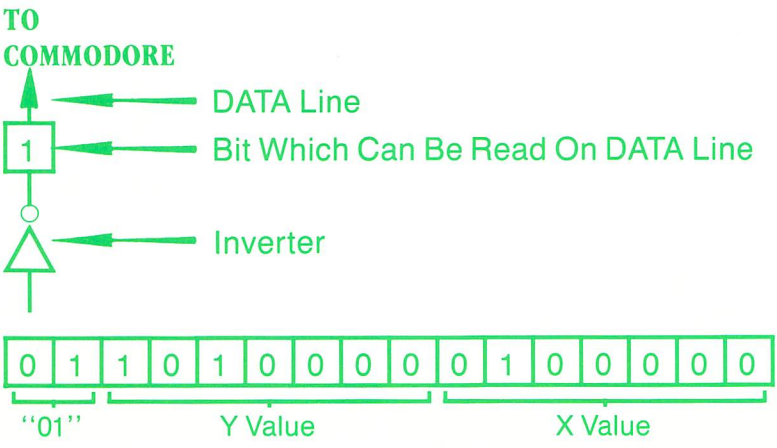
When PowerPad senses a touch, it stores the X and Y coordinate values in a 16-bit shift register. Figure 3a shows an example of what the shift register might contain immediately after the pad has sensed a touch. The left-most two bits are always loaded with "01." The next seven bits to the right contain the value for Y, and the right-most seven bits hold the value for X. The X and Y values both are stored with their least significant bit to the left. So in Figure 3a, the shift register is shown holding the coordinates (X=2, Y=5).

At any time the computer can read only the left-most bit of the shift register. This bit is inverted (a 1 becomes a 0, and a 0 becomes a 1) and then sent out on PowerPad's DATA line.

The CLOCK line enables the computer to read the other bits in the shift register. Each time the CLOCK line is pulsed (set from low to high and back to low again), the shift register moves its contents one position to the left. The left-most bit is lost and a "0" is shifted into the right-most position. Figure 3b shows the contents of the shift register after the CLOCK line has been pulsed once. Figure 3c shows the register after a second shift. The first two shifts are always necessary to remove the "01" in the left-most bit positions. At this point the computer can begin reading the seven bits corresponding to the Y value, followed by the seven bits for the X value. Remember that since the bits coming out of the DATA line are inverted, the computer must invert them again.

The CLOCK line should be held low while PowerPad is scanning. If the line is pulsed before the pad finds a point, the shift register immediately loads garbage while the pad continues to scan for a point. Then when the pad finds the next point, it is unable to put the correct data into the shift register. At this point, pulsing the CLEAR line clears the garbage from the register and causes the pad to resume normal operation.

Figure 3a. Initial Contents of Shift Register



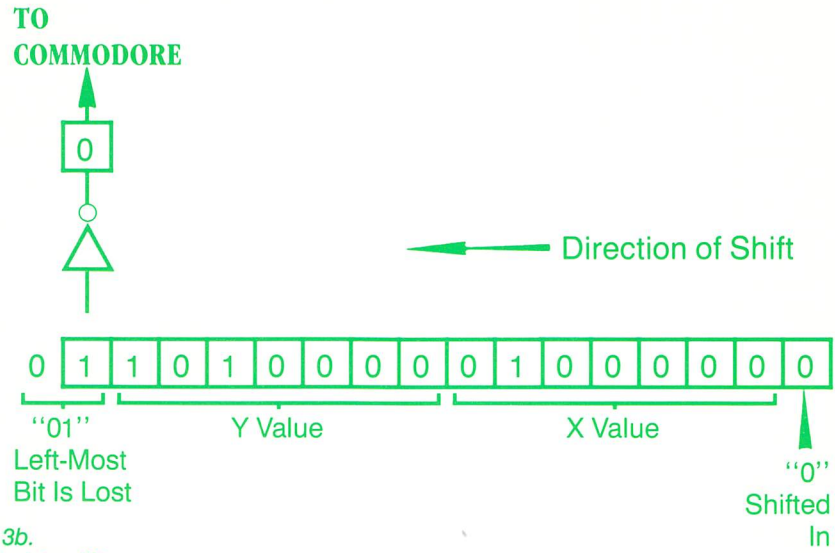


Figure 3b.
Shift Register After
One Clock Pulse

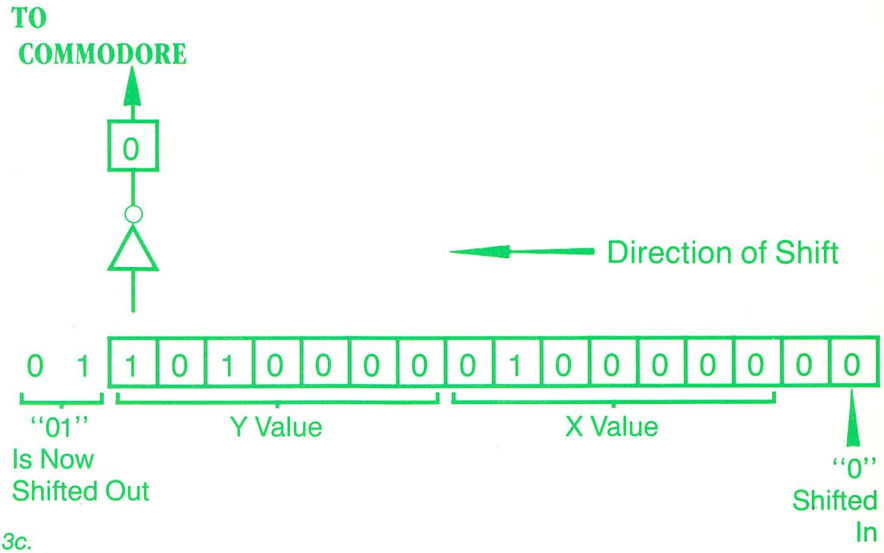


Figure 3c.
Shift Register After
Two Clock Pulses

Programming Considerations

As PowerPad scans its matrix of switches, it starts at coordinates (X=0, Y=0), then looks at (X=0, Y=1), etc., up to (X=0, Y=119). It then looks at the next column, (X=1, Y=0) up to (X=1, Y=119), and continues scanning each successive column until it reaches (X=119, Y=119). At this point PowerPad scans again, starting with (X=0, Y=0). If more than one point is closed in the switch matrix, PowerPad scans to the first point, lowers the SENSE line and waits until the computer reads the coordinates. When the computer acknowledges reading the coordinates (by pulsing the CLEAR line), PowerPad resumes scanning the switch matrix at the next point following the one it just reported. When it finds another closed switch in the matrix, it again lowers the SENSE line and waits for the computer to read the coordinates before continuing to scan.

Each time PowerPad scans through the point (X=0, Y=0), it reports a switch closure at that point, even if nothing is touching the pad. This is a very useful feature. If a program reads the pad and finds coordinates (X=0, Y=0) two times in succession, then it knows that PowerPad has scanned all the way through its switch matrix without finding any closed switch points. In other words, this indicates that nothing is touching the pad. Programs can use this feature to detect when you have lifted your finger off PowerPad.

It is possible that some points in PowerPad's switch matrix may close permanently due to extraordinary wear and tear.

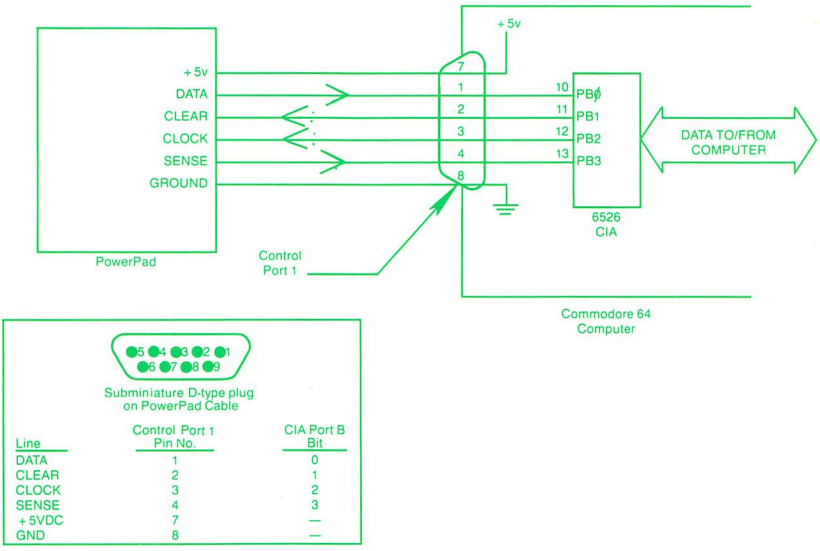
When a point becomes shorted, PowerPad reports its coordinates each time it scans through the matrix, just as if you were applying continuous pressure to that point. Writing a routine to detect shorted points and to ignore them is not difficult. At the start of a program, create a table of all shorted points (points

reported as being touched while nothing is really touching PowerPad) with the exception of (X=0, Y=0). Later in the program, whenever a location read from the pad matches any of the points in the table, the program ignores that location.

Commodore 64's Control Port Interface

When you plug PowerPad into Commodore the 64 control port 1, the CLOCK, CLEAR, SENSE, and DATA lines from the PowerPad are connected to Port B of a 6526 Complex Interface Adapter chip (CIA) inside the Commodore 64 (see Figure 4). All PowerPad software expects to find PowerPad in control port 1.

Figure 4. Commodore Control Port Interface



Subminiature D-type plug on PowerPad Cable

Line	Control Port 1 Pin No.	CIA Port B Bit
DATA	1	0
CLEAR	2	1
CLOCK	3	2
SENSE	4	3
+5VDC	7	—
GND	8	—

The four CIA lines used by PowerPad are also used by the Commodore 64's keyboard. This sharing of lines makes normal access to the keyboard impossible while PowerPad is connected. However, all of the example programs in this manual **are** able to examine the space bar and allow you to use that key to halt the programs. If you wish to obtain additional information concerning how to read the keyboard while PowerPad is plugged in, send your questions along with your PadMasters Guild membership application (see last page).

The registers of the 6526 CIA are memory mapped in the Commodore 64 to locations $DC00 (decimal 56320) through $DC0F (decimal 56335). You can access the registers by loading and storing (or PEEKing and POKEing) data at these addresses. There are two particular registers in the 6526 with which we are concerned. They are: the Port B Data Direction Register, or DDRB, at $DC03; and the Port B Register, or PRB, at $DC01.

The individual bits of DDRB determine whether a line on the control port is an input to or an output from the Commodore 64. Setting a bit to 0 assigns the corresponding line as an input, and setting it to 1 assigns the line as an output. PowerPad requires that lines 0 and 3 be inputs, and lines 1 and 2 be outputs. So you should store the value 6 (binary 0110) as the low four bits of DDRB.

Once you have set up DDRB, you can read the DATA and SENSE lines and write to the CLOCK and CLEAR lines through PRB. Setting a bit of PRB to 0 or 1 sets the corresponding output line low or high, respectively. Setting an input line low or high sets its corresponding PRB bit to 0 or 1, respectively. Given below are machine language examples of how to set up and talk to PowerPad:

```
LDA #$06        ;  binary 0000 0110
STA $DC03       ;  set up DDRB (2 inputs & 2 outputs)

LDA #$00        ;  binary 0000 0000
STA $DC01       ;  set CLEAR and CLOCK low

LDA #$02        ;  binary 0000 0010
STA $DC01       ;  set CLEAR high and CLOCK low

LDA #$04        ;  binary 0000 0100
STA $DC01       ;  set CLOCK high and CLEAR low

LDA $DC01       ;  read the DATA line
AND $#01        ;  binary 0000 0001
                ;  giving: A = 0 for DATA = low
                ;           A = 1 for DATA = high

LDA $DC01       ;  read the SENSE line
AND $#08        ;  binary 0000 1000
                ;  giving: A = 0 for SENSE = low
                ;           A = 8 for SENSE = high
```

While entering Program 2, make certain that the DATA statements are **exactly** as shown here. Otherwise the program will not run properly and could even cause the computer to "freeze". If you have a disk or cassette, save the program before running it.

These Data statements contain the machine language subroutines taken from the listings in Appendix B.

Program 2:
PowerPad Test (BASIC with machine language subroutines. See Appendix B for an assembly language listing of the subroutines).

```
1 REM ******** POWERPAD TEST ********
2 REM POWERPAD DEMONSTRATION PROGRAM
3 REM IN BASIC WITH MACHINE LANGUAGE
4 REM SUBROUTINES FOR COMMODORE 64.
5 REM NOTE: USE THE SUBROUTINES IN
6 REM       LINES 10000-15520
7 REM       FOR YOUR OWN PROGRAMS
8 REM  (YOU CAN LEAVE OUT ALL REMS)
9 REM CHALK BOARD, INC. 1983
10 REM*****************************
100 PRINT CHR$(147): REM CLEAR SCREEN
110 PRINT "POWERPAD TEST"
120 GOSUB 10010: REM SET UP M.L.
130 GOSUB 11010: REM CALL INITPAD
140 GOSUB 12010: REM CALL GETSENSE
150 GOSUB 13010: REM CALL GETXY
200 REM USE X AND Y
210 IF X=0 AND Y=0 THEN 140
220 PRINT"X=";X; "  Y=";Y
230 GOTO 140
240 REM *****************************
10000 REM SET UP MACH. LANG. SUBROUTINES
10010 PRINT "PLUG POWERPAD INTO CONTROL PORT 1."
10020 GOSUB 14010: REM POKE IN M.L.
10030 IC=0: REM COUNTS CALLS OF INIT
10040 PRINT "DONE INSTALLING M.L."
10050 RETURN
10060 REM *****************************
11000 REM CALL INIT PAD
11010 IC=IC+1: IF IC=5 THEN 11110
11020 SYS 49152: REM $C000 = INITPAD
11030 RETURN
11040 REM ==========
11100 REM INIT PAD HAS BEEN CALLED 5 TIMES
11110 PRINT "UNABLE TO TALK TO POWERPAD.  DID"
11120 PRINT "YOU PLUG PAD INTO CONTROL PORT 1?"
11130 REM ==========
11200 REM RE-ENABLE KEYBOARD
11210 PRINT "UNPLUG POWERPAD BEFORE TYPING."
11220 SYS 49161: REM $C009 = RESTORE
11230 END
11240 REM *****************************
12000 REM CALL GETSENSE
12010 SYS 49155: REM $C003 = GETSENSE
12020 SENSE=PEEK(49166): REM $C00E = SNS
12030 IF SENSE<2 THEN 12100: REM SPACE NOT PRESSED
12040 PRINT "PROGRAM STOPPED.": GOTO 11210
12100 IF SENSE=1 THEN 12120: REM POINT NOT FOUND
```

```
12110 IC=0: RETURN: REM RESET INIT COUNTER
12120 GOSUB 11010: REM CALL INIT AGAIN
12130 GOTO.12010
12140 REM ****************************
13000 REM CALL GETXY
13010 SYS 49158: REM $C006 = GETXY
13020 X=PEEK(49165): REM $C00D
13030 Y=PEEK(49164): REM $C00C
13040 RETURN
13050 REM ****************************
14000 REM INSTALL MACH. LANG. SUBROUTINES
14010 FOR L=49152 TO 49307: REM $C000-C09B
14020 READ BYTE$
14030 K=ASC(MID$(BYTE$,1,1))-48
14040 IF K>9 THEN K=K-7
14050 BYTE=K*16
14060 K=ASC(MID$(BYTE$,2,1))-48
14070 IF K>9 THEN K=K-7
14080 BYTE=BYTE+K
14090 POKE L,BYTE
14100 NEXT L
14110 RETURN
14120 REM READ HEXADECIMAL BYTE FROM DATA
14130 REM STATEMENTS; CONVERT BYTE TO
14140 REM DECIMAL; STORE VALUE IN MEMORY
14150 REM ****************************
15000 REM SUBROUTINE VECTORS & VARS
15010 DATA 4C,0F,C0,4C,27,C0,4C,4F
15020 DATA C0,4C,8E,C0,00,00,00
15100 REM INITPAD
15110 DATA A9,16,8D,03,DC,A9,00,8D
15120 DATA 02,DC,8D,01,DC,A9,02,8D
15130 DATA 01,DC,A9,00,8D,01,DC,60
15200 REM GETSENSE
15210 DATA A2,FF,A0,0F,AD,00,DC,29
15220 DATA 80,D0,06,A9,02,8D,0E,C0
15230 DATA 60,AD,01,DC,29,08,D0,04
15240 DATA 8D,0E,C0,60,CA,D0,E5,88
15250 DATA D0,E2,A9,01,8D,0E,C0,60
15300 REM GETXY
15310 DATA A9,04,8D,01,DC,A9,00,8D
15320 DATA 01,DC,20,71,C0,8D,0C,C0
15330 DATA 20,71,C0,8D,0D,C0,A9,02
15340 DATA 8D,01,DC,A9,00,8D,01,DC
15350 DATA 60
15400 REM GETBYTE
15410 DATA 00,A2,07,A9,04,8D,01,DC
15420 DATA A9,00,8D,01,DC,AD,01,DC
15430 DATA 4A,6E,70,C0,CA,D0,EC,AD
15440 DATA 70,C0,49,FF,4A,60
15500 REM RESTORE
15510 DATA A9,FF,8D,02,DC,8D,00,DC
15520 DATA A9,00,8D,03,DC,60
```

Creating Your Own Command Buttons

A command button is an area on the PowerPad surface which, when pushed, calls into action a subroutine within your program. You can design your own command buttons anywhere on the surface of PowerPad. Program 3 assists you by providing the range of values for X and Y corresponding to your command button. You can use the information Program 3 provides to determine when your program will call its command button subroutines.

Program 3:
Command Button Mapper. You must also enter the machine language subroutines from Program 2 (lines 10000-15520)

```
1 REM **** COMMAND BUTTON MAPPER ****
2 REM GIVES YOU THE RANGE OF X AND Y
3 REM VALUES FOR YOUR COMMAND BUTTONS
4 REM !REQUIRES THE MACHINE LANGUAGE
5 REM SUBROUTINES (LINES 10000-15520)
6 REM FROM PROGRAM 2!
8 REM  (YOU CAN LEAVE OUT ALL REMS)
9 REM CHALK BOARD, INC. 1983
10 REM******************************
100 PRINT CHR$(147): REM CLEAR SCREEN
110 GOSUB 1010: REM INSTRUCTIONS & INIT VARS
120 GOSUB 10010: REM SET UP M.L.
130 GOSUB 11010: REM CALL INITPAD
140 GOSUB 12010: REM CALL GETSENSE
150 GOSUB 13010: REM CALL GETXY
200 REM USE X AND Y
210 IF X+Y=0 THEN 140
220 GOSUB 2010: REM UPDATE RANGE
230 GOTO 140
240 REM ******************************
1000 REM INSTRUCTIONS & INIT VARS
1010 XHI=0:   REM LEFTMOST VALUE
1020 XLO=119: REM RIGHTMOST VALUE
1030 YHI=0:   REM BOTTOM VALUE
1040 YLO=119: REM TOP VALUE
1100 PRINT "COMMAND BUTTON MAPPER": PRINT
```

- Insert the grid overlay onto the surface of PowerPad.
- Choose an area of PowerPad's surface to represent your button.
- Use your marking pen to draw a square to identify the location of your button. You can label or color your button any way you choose.
- If you need to change a button's location, refer to Appendix A for instructions concerning erasing marks on your overlay.
- Enter Program 3 into your computer.
- RUN.

```
1110 PRINT "TO USE THIS PROGRAM:"
1120 PRINT "~ MOVE YOUR FINGER OVER THE"
1130 PRINT "    ENTIRE AREA OF YOUR BUTTON"
1140 PRINT "    STAYING *INSIDE* ITS OUTLINE"
1150 PRINT"~ THE MAXIMUM AND MINUMUM X AND"
1160 PRINT "    Y VALUES OF YOUR BUTTON"
1170 PRINT "    WILL BE CONTINUOUSLY"
1180 PRINT "    PRINTED."
1190 PRINT "~ CONTINUE TO PRESS THE BUTTON"
1200 PRINT "    UNTIL THE X AND Y VALUES"
1210 PRINT "    NO LONGER CHANGE."
1220 PRINT "    (THESE ARE THE VALUES YOU"
1230 PRINT "    SHOULD USE IN YOUR PROGRAM)"
1240 PRINT "~ PRESS THE SPACE BAR TO STOP"
1250 PRINT "    THIS PROGRAM."
1260 PRINT
1270 INPUT "PRESS RETURN WHEN READY";A$
1280 PRINT CHR$(147)
1290 RETURN
1300 REM ****************************
2000 REM UPDATE & PRINT RANGE OF X & Y
2010 IF X<XLO THEN XLO=X
2020 IF X>XHI THEN XHI=X
2030 IF Y<YLO THEN YLO=Y
2040 IF Y>YHI THEN YHI=Y
2050 PRINT CHR$(147)
2060 PRINT "X>";XLO;"AND X<";XHI;
2070 PRINT "AND Y>";YLO;"AND Y<";YHI
2080 PRINT: PRINT: PRINT
2090 RETURN
```

Once you have determined the range of values for X and Y (for example, X might range between 20 and 30, and Y between 40 and 50), you can set up your program to know when you are pressing your command button. Assume that you want your command button to activate a routine which begins at line 3000 in your own program. To call the routine you must insert a line in your original PowerPad Test program to say the following: if I touch PowerPad at this spot, then go to line 3000 and do the routine there. This can be represented in BASIC in the following way:

```
225  IF X>20 AND X<30 AND Y>40 AND Y<50 THEN GOTO 3000
```

In this example **line 3000** is the beginning of the routine you want executed whenever your command button is pressed.

When the program returns from the routine and reads the pad again, it might find that you are still pressing the command button. If the routine is called each time the pad reports a switch closure within your pushbutton area, then the routine is likely to be run several times for each push of the button.

If you do not want the routine running several times for each push of the button, there is an easy way to avoid re-running the routine. Use a variable to indicate whether or not the button has been pressed. When the pad reports a point within your button area, your program checks the variable to see if you have already pressed the button. If not, then it calls the routine to perform the function associated with that button and sets the variable to show that that function has been used.

If the variable shows that you previously pressed the button, then the program ignores that button until you either lift your finger from the pad or touch another button. In both cases the program can then reset the variable so that the button will work the next time you press it. When your program reads the pad and finds two (X=0, Y=0) points in succession, it knows that nothing is touching the pad.

Program 4, "Making Change," uses this method to insure that when you hold down a coin button, the value of that coin is added to your total only once.

Sample Programs

Program 4 is an educational game in which you make change for random amounts of money (1–100 cents). Try changing some of the PRINT statements to have fun with the responses.

Creating Your Overlay For "Making Change"

- Fit the overlay on PowerPad's surface.
- Using a marking pen, divide the overlay into four equal vertical columns. See Figure 5.
- Mark each section with a drawing of its appropriate coin value. From left to right show: Quarter, Dime, Nickel, Penny.

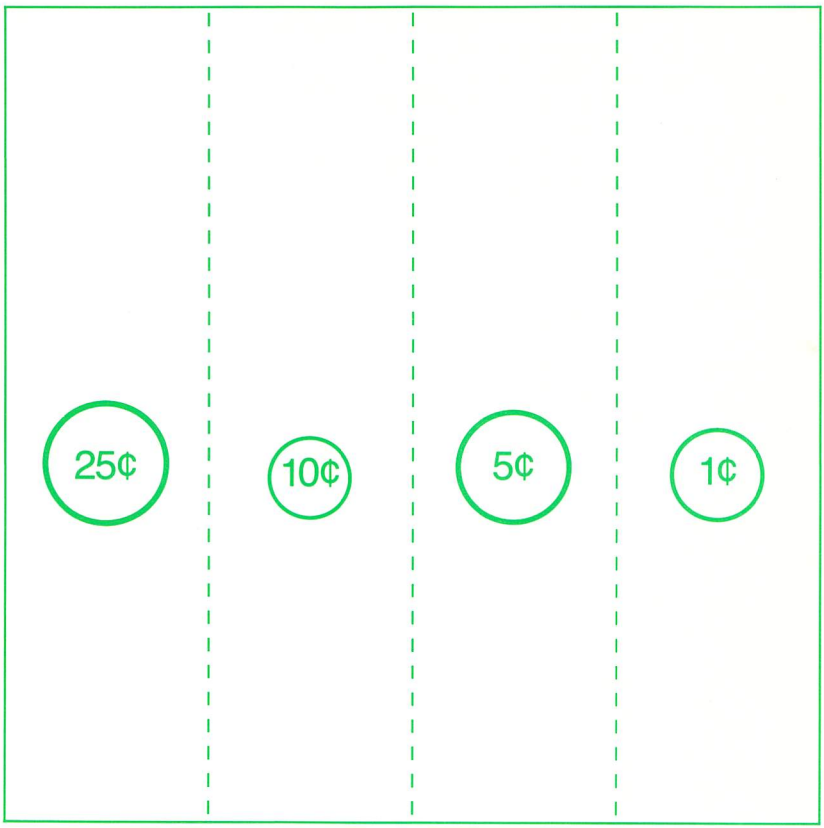


Fig. 5.
Overlay For
"Making Change" Program

Program 4:

Making Change. You must also enter the machine language subroutines from Program 2 (lines 10000-15520)

```
1 REM ******** MAKING CHANGE ********
2 REM AN EDUCATIONAL GAME USING THE
3 REM POWERPAD
4 REM !REQUIRES THE MACHINE LANGUAGE
5 REM SUBROUTINES (LINES 10000-15520)
6 REM FROM PROGRAM 2!
8 REM  (YOU CAN LEAVE OUT ALL REMS)
9 REM CHALK BOARD, INC. 1983
10 REM*******************************
100 PRINT CHR$(147): REM CLEAR SCREEN
110 GOSUB 10010: REM SET UP M.L.
120 GOSUB 1010: REM INIT VARS & PICK AMOUNT
130 GOSUB 11010: REM CALL INITPAD
140 GOSUB 12010: REM CALL GETSENSE
150 GOSUB 13010: REM CALL GETXY
200 REM USE X AND Y
210 GOSUB 2010
220 GOTO 140
230 REM *******************************
1000 REM INIT VARIABLES
1010 PS=0: NS=0: DS=0: QS=0: REM # OF COINS
1020 TTL=0: REM TOTOL VALUE OF COINS
1030 AMT=INT(RND(0)*99)+1: REM # TO MATCH
1040 PF=1: NF=1: DF=1: QF=1: REM "TOUCHED" FLAGS
1050 PRINT
1060 PRINT "============================="
1070 PRINT
1080 PRINT "LET'S MAKE CHANGE FOR";AMT;"CENTS"
1090 PRINT
1100 RETURN
1110 REM *******************************
2000 REM CONVERT X & Y TO A COIN
2010 IF X+Y=0 THEN 2100: REM NO POINT?
2020 ZF=0: REM POINT WAS NOT (0,0)
2030 IF X>=0 AND X<30 THEN 5010: REM PENNY
2040 IF X>=30 AND X<60 THEN 6010: REM NICKEL
2050 IF X>=60 AND X<90 THEN 7010: REM DIME
2060 IF X>=90 AND X<120 THEN 8010: REM QUARTER
2100 ZF=ZF+1: REM COUNT CONSECUTIVE (0,0)'S
2110 IF ZF<2 THEN RETURN: ONLY 1 SO FAR
2120 GOSUB 3010: REM 2 SO CLEAR "TOUCHED" FLAGS
2130 RETURN
2140 REM *******************************
3000 REM CLEAR ALL "BUTTON TOUCHED" FLAGS
3010 PF=0: NF=0: DF=0: QF=0
3020 RETURN
3030 REM *******************************
```

```
4000 REM TOTAL UP COINS
4010 PRINT
4020 PRINT "MAKING CHANGE FOR";AMT;"CENTS"
4030 PRINT
4040 IF PS>0 THEN PRINT "    PENNIES:";PS
4050 IF NS>0 THEN PRINT "    NICKELS:";NS
4060 IF DS>0 THEN PRINT "      DIMES:";DS
4070 IF QS>0 THEN PRINT "   QUARTERS:";QS
4080 PRINT
4090 PRINT "YOUR TOTAL SO FAR IS";TTL;"CENTS"
4100 PRINT
4110 IF TTL>=AMT THEN 4130
4120 RETURN: REM NOT ENOUGH YET
4130 IF TTL>AMT THEN 4160
4140 PRINT"CONGRATULATIONS! YOU GOT IT!
4150 GOTO 4170
4160 PRINT "WHOOPS! THAT WAS TOO MUCH."
4170 GOSUB 1010: REM DO IT AGAIN
4180 RETURN
4190 REM ******************************
5000 REM PENNY WAS TOUCHED
5010 IF PF=1 THEN RETURN: REM WAS JUST TOUCHED
5020 GOSUB 3010: REM ALLOW ALL OTHER BUTTONS
5030 PF=1: REM SHOW THAT PENNY TOUCHED
5040 PS=PS+1: REM INCREMENT PENNY COUNTER
5050 TTL=TTL+1: REM UPDATE TOTAL
5060 GOTO 4010
5070 REM ******************************
6000 REM NICKEL WAS TOUCHED
6010 IF NF=1 THEN RETURN: REM WAS JUST TOUCHED
6020 GOSUB 3010: REM ALLOW ALL OTHER BUTTONS
6030 NF=1: REM SHOW THAT NICKEL TOUCHED
6040 NS=NS+1: REM INCREMENT NICKEL COUNTER
6050 TTL=TTL+5: REM UPDATE TOTAL
6060 GOTO 4010
6070 REM ******************************
7000 REM DIME WAS TOUCHED
7010 IF DF=1 THEN RETURN: REM WAS JUST TOUCHED
7020 GOSUB 3010: REM ALLOW ALL OTHER BUTTONS
7030 DF=1: REM SHOW THAT DIME TOUCHED
7040 DS=DS+1: REM INCREMENT DIME COUNTER
7050 TTL=TTL+10: REM UPDATE TOTAL
7060 GOTO 4010
7070 REM ******************************
8000 REM QUARTER WAS TOUCHED
8010 IF QF=1 THEN RETURN: REM WAS JUST TOUCHED
8020 GOSUB 3010: REM ALLOW ALL OTHER BUTTONS
8030 QF=1: REM SHOW THAT QUARTER TOUCHED
8040 QS=QS+1: REM INCREMENT QUARTER COUNTER
8050 TTL=TTL+25: REM UPDATE TOTAL
8060 GOTO 4010
8070 REM ******************************
```

Program 5 is a small sample of the graphics capabilities of the Commodore 64 in combination with your PowerPad. The program draws a waving man on the screen when you touch the pad.

Program 5:

Waving Man. You must also enter the machine language subroutines from Program 2 (lines 10000-15520)

```
1 REM ********** WAVING MAN **********
2 REM POWERPAD DEMONSTRATION PROGRAM
4 REM !REQUIRES THE MACHINE LANGUAGE
5 REM SUBROUTINES (LINES 10000-15520)
6 REM FROM PROGRAM 2!
8 REM  (YOU CAN LEAVE OUT ALL REMS)
9 REM CHALK BOARD, INC. 1983
10 REM*******************************
100 PRINT CHR$(147): REM CLEAR SCREEN
110 GOSUB 1010 : REM SET UP GRAPHICS CHARS
120 GOSUB 10010: REM SET UP M.L.
130 GOSUB 11010: REM CALL INITPAD
140 GOSUB 12010: REM CALL GETSENSE
150 GOSUB 13010: REM CALL GETXY
200 REM USE X AND Y
210 IF X+Y=0 THEN 140
220 GOSUB 2010
230 GOTO 140
240 REM *******************************
1000 REM INIT WAVING MAN
1010 HD$=CHR$( 32)+CHR$(113)+CHR$( 32)
1020 TP$=CHR$( 99)+CHR$(123)+CHR$( 99)
1030 TR$=CHR$( 99)+CHR$(123)+CHR$(189)
1040 TL$=CHR$(173)+CHR$(123)+CHR$( 99)
1050 MD$=CHR$( 32)+CHR$(125)+CHR$( 32)
1060 LG$=CHR$(110)+CHR$( 32)+CHR$(109)
1070 PRINT "WAVING MAN"
1080 RETURN
1090 REM *******************************
```

```
2000 REM DISPLAY WAVING MAN
2010 XP=((119-X)*36)/119: REM COMPUTE LOCATION
2020 YP=(Y*21)/119:  REM FROM X AND Y
2030 A=1024+(INT(YP)*40)+INT(XP)
2040 AH=INT(A/256): REM LOCATION FOR HEAD
2050 AL=A-(AH*256)
2060 A=A+40
2070 BH=INT(A/256): REM LOCATION FOR ARMS
2080 BL=A-(BH*256)
2090 A=A+40
2100 CH=INT(A/256): REM LOCATION FOR WAIST
2110 CL=A-(CH*256)
2120 A=A+40
2130 DH=INT(A/256): REM LOCATION FOR LEGS
2140 DL=A-(DH*256)
2150 PRINT CHR$(147): REM CLEAR SCREEN
2160 FOR M=1 TO 2
2170 POKE 209,AL
2180 POKE 210,AH
2190 PRINT HD$: REM DRAW HEAD
2200 POKE 209,BL
2210 POKE 210,BH
2220 PRINT TP$: REM DRAW STRAIGHT ARMS
2230 POKE 209,CL
2240 POKE 210,CH
2250 PRINT MD$: REM DRAW WAIST
2260 POKE 209,DL
2270 POKE 210,DH
2280 PRINT LG$: REM DRAW LEGS
2290 FOR M1=1 TO 10: NEXT M1: REM DELAY
2300 POKE 209,BL
2310 POKE 210,BH
2320 IF X>60 THEN PRINT TL$: REM DRAW WAVING
2330 IF X<61 THEN PRINT TR$: REM    ARMS
2340 FOR M1=1 TO 10: NEXT M1: REM DELAY
2350 NEXT M
2360 RETURN
2370 REM ****************************
```

Questions to Ponder

A. How would you modify "Making Change" so that you must touch the pad only within the outline of each coin?

B. How would you design a STOP command button that, when pressed, would stop the program, replacing the function of the space bar?

C. How could you change the PowerPad Test program to print out the value (1 or 0) for each data bit read in from the pad? How could you change the PowerPad Test program to print the state of the SENSE line (high or low)?

D. On your PowerPad overlay draw a medium-sized rectangle with sides parallel to the sides of PowerPad. Then RUN the "Waving Man" program (Program 5). Next touch any three of the four corners of your rectangle simultaneously. Continue touching the three corners until the waving man has appeared at all positions several times. What do you notice about the route of the waving man?

The waving man appears at the fourth corner of the rectangle—even though no pressure was applied to that point. This "false-image" is also known as "aliasing." What would explain this occurrence? How can you avoid this in your own programs? How can you take advantage of this in your own programs?

Do you have any ideas that you would like to share with us about your experiences with the PowerPad Programming Kit? Do you want to know more about what other people are doing with PowerPad and Chalk Board products? If so, please write to us at the address given at the front of this booklet.

Your name will be placed on a list of users who receive our company newsletter. If you send us some of your ideas, you might also appear in our newsletter.

Watch for coming issues of our newsletter, as well as upcoming additions to Leonardo's Library of quality software for your home computer.

Appendix A. Care of Overlays, Replacement Overlays

The overlay is designed for use with colored marking pens of the sort included in the Programming Kit. Marks made by these pens should not smudge or smear significantly with normal use. The use of colored marking pens of another make is not recommended because they can smear or be more difficult to erase.

To clean an overlay, use a damp cloth or paper towel. **Always** remove the overlay from PowerPad before cleaning since water may damage the surface of PowerPad.

Additional overlays may be ordered from:

Customer Support
Chalk Board, Inc.
Suite 140
3772 Pleasantdale Rd.
Atlanta, Georgia 30340

The charge for each replacement overlay is $6.00, which includes $3.00 for the overlay itself and $3.00 for postage and handling.

Appendix B. Machine Language Subroutines Used By Programs 2, 3, 4 and 5

```
                ;***********************************************
                ;     Machine language subroutines to interface
                ;           PowerPad to the Commodore 64.
                ;              Chalk Board, Inc. 1983
                ;***********************************************

                ;Define constants:

DC03            DDRB     =   $DC03      ;CIA Port B Data Dir. Reg.
DC01            PRB      =   $DC01      ;CIA Port B
DC02            DDRA     =   $DC02      ;CIA Port A Data Dir. Reg.
DC00            PRA      =   $DC00      ;CIA Port A

0002            CLRHIGH  =   $02        ;to pulse CLEAR bit (0000 0010)
0004            CLKHIGH  =   $04        ;to pulse CLOCK bit (0000 0100)
0008            SNSMASK  =   $08        ;mask for SENSE bit (0000 1000)
0080            SPSMASK  =   $80        ;mask for space bit (1000 0000)

0016            INTPRB   =   $16        ;to init Port B DDR (0001 1001)

C000                    .=   $C000      ;starting address

```

```
                        23 ;Vectors to subroutine entry points:
                        24
C000 4C 0F C0           25         JMP INITPAD
C003 4C 27 C0           26         JMP GETSENSE
C006 4C 4F C0           27         JMP GETXY
C009 4C 8E C0           28         JMP RESTORE
                        29
                        30 ;Returned values:
                        31
C00C                    32 YPOS:                    ;Y coordinate returned by GETXY
C00D                    33         .=.+1
C00D                    34 XPOS:                    ;X coordinate returned by GETXY
C00E                    35         .=.+1
C00E                    36 SNS:                     ;status returned by GETSENSE
C00F                    37         .=.+1
```

```
                        ;*************************************************
                        ;Subroutine INITPAD:
                        
                        ;Initialize the Commodore 64 CIA as follows:
                        
                        ;   Port B bit 0 is DATA line input
                        ;          bit 1 is CLEAR line output
                        ;          bit 2 is CLOCK line output
                        ;          bit 3 is SENSE line output
                        ;          bit 4 is output for space check (always 0)
                        
                        ;   Port A all bits are inputs
                        
                        ;Initialize PowerPad:
                        
                        ;   Reset CLEAR and CLOCK lines to 0
                        ;   Pulse CLEAR line to begin scanning
                        ;*************************************************
                        
C00F                    INITPAD:
C00F A9 16              LDA #INTPRB     ;set up Port B (2 inputs,
C011 8D 03 DC           STA DDRB        ;   3 outputs)
C014 A9 00              LDA #$00        ;set up Port A (8 inputs)
C016 8D 02 DC           STA DDRA
C019 8D 01 DC           STA PRB         ;reset all lines
C01C A9 02              LDA #CLRHIGH    ;pulse CLEAR line:
C01E 8D 01 DC           STA PRB         ;   high (begin scanning)
C021 A9 00              LDA #$00        ;   bring CLEAR back low
C023 8D 01 DC           STA PRB
C026 60                 RTS             ;return to BASIC
```

```
                        68 ;***************************************************
                        69 ;Subroutine GETSENSE
                        70
                        71 ;Examine the SENSE line from the PowerPad
                        72 ;   and the space bar from the keyboard.
                        73
                        74 ;Store result in SNS:
                        75 ;   0 if SENSE low (point found),
                        76 ;   1 if SENSE high (still scanning), or
                        77 ;   2 if space bar pressed
                        78 ;***************************************************
                        79
C027                    80 GETSENSE:
C027 A2 FF              81          LDX #$FF        ;init loop counters
C029 A0 0F              82          LDY #$0F        ;   (do $FFF tries)
C02B AD 00 DC           83 GSLOOP:  LDA PRA         ;check for space bar
C02E 29 80              84          AND #SPSMASK    ;   mask out all but bit 7
C030 D0 06              85          BNE NOSPACE     ;branch if not pressed
C032 A9 02              86          LDA #$02        ;was pressed, so set
C034 8D 0E C0           87          STA SNS         ;   SNS to 2, and
C037 60                 88          RTS             ;   return to BASIC
C038                    89 NOSPACE:
C038 AD 01 DC           90          LDA PRB         ;check SENSE line
C03B 29 08              91          AND #SNSMASK    ;   mask out all but bit 3
C03D D0 04              92          BNE NOSENSE     ;branch if no point avail.
C03F 8D 0E C0           93          STA SNS         ;   point avail.; set SNS to 0
C042 60                 94          RTS             ;   return to BASIC
C043                    95 NOSENSE:
C043 CA                 96          DEX             ;no space or point yet, so
C044 D0 E5              97          BNE GSLOOP      ;   keep checking
C046 88                 98          DEY
C047 D0 E2              99          BNE GSLOOP
C049 A9 01             100          LDA #$01        ;no space or point,  so set
C04B 8D 0E C0          101          STA SNS         ;   SNS to 1, and
C04E 60                102          RTS             ;   return to BASIC
```

```
                    103 ;************************************************
                    104 ;Subroutine GETXY
                    105
                    106 ;Strip off the initial two bits ("01"), read Y,
                    107 ;   read X, and resume scanning.  Return X and Y
                    108 ;   in XPOS and YPOS.
                    109 ;************************************************
                    110
C04F                111 GETXY:
C04F A9 04          112         LDA #CLKHIGH    ;pulse CLOCK line (shift off
C051 8D 01 DC       113         STA PRB         ;   left shift register bit)
C054 A9 00          114         LDA #$00        ;   bring CLOCK back low
C056 8D 01 DC       115         STA PRB
                    116
C059 20 71 C0       117         JSR GETBYTE     ;get Y value,
C05C 8D 0C C0       118         STA YPOS        ;   store in YPOS
                    119
C05F 20 71 C0       120         JSR GETBYTE     ;get X value,
C062 8D 0D C0       121         STA XPOS        ;   store in XPOS
                    122
C065 A9 02          123         LDA #CLRHIGH    ;pulse CLEAR line (resume
C067 8D 01 DC       124         STA PRB         ;   scanning)
C06A A9 00          125         LDA #$00        ;   bring CLEAR back low
C06C 8D 01 DC       126         STA PRB
C06F 60             127         RTS             ;return to BASIC
```

```
                    128 ;***************************************************
                    129 ;Subroutine GETBYTE
                    130 ;   (called by GETXY)
                    131
                    132 ;Read and ignore 1 bit, then assemble next 7 bits
                    133 ;   (X or Y value).  Return result in A reg.
                    134 ;***************************************************
                    135
C070                136 BYTE:                   ;used to assemble X and Y
C071                137         .=.+1
                    138
C071                139 GETBYTE:
C071 A2 07          140         LDX #$07        ;count 7 bits
C073                141 GBLOOP:
C073 A9 04          142         LDA #CLKHIGH    ;pulse CLOCK line (shift in
C075 8D 01 DC       143         STA PRB         ;   new shift register bit)
C078 A9 00          144         LDA #$00        ;   bring CLOCK back low
C07A 8D 01 DC       145         STA PRB
                    146
C07D AD 01 DC       147         LDA PRB         ;read DATA line (bit 0)
C080 4A             148         LSR A           ;shift bit 0 into carry
C081 6E 70 C0       149         ROR BYTE        ;shift carry into bit 7
C084 CA             150         DEX             ;decrement bit counter
C085 D0 EC          151         BNE GBLOOP      ;done 7 yet?  if not, branch
C087 AD 70 C0       152         LDA BYTE        ;get assembled byte
C08A 49 FF          153         EOR #$FF        ;invert all of the bits
C08C 4A             154         LSR A           ;do 8th shift (right justify)
C08D 60             155         RTS             ;return to GETXY
```

```
                    156 ;************************************************
                    157 ;Procedure RESTORE
                    158
                    159 ;Restore PRA and PRB to re-enable the keyboard
                    160 ;************************************************
                    161
C08E                162 RESTORE:
C08E A9 FF          163         LDA #$FF        ;restore DDRA to all outputs
C090 8D 02 DC       164         STA DDRA
C093 8D 00 DC       165         STA PRA         ;reset PRA to all 1's
C096 A9 00          166         LDA #$00        ;reset DDRB to all inputs
C098 8D 03 DC       167         STA DDRB
C09B 60             168         RTS             ;return to BASIC
```

Bibliography

Brannon, Charles. Using joysticks on the Commodore 64: a basic tutorial. *Computer Gazette*, July 1983, pp. 90-93.

Commodore 64 Programmer's Reference Guide. Howard W. Sams and Company, Inc.

Commodore 64 User's Guide. Howard W. Sams and Company, Inc.

Golden, Neal. *Computer Programming in the BASIC Language.* New York: Harcourt, Brace, Jovanovich, 1975.

Hampshire, Nick. *VIC Revealed.* Rochelle Park, New Jersey: Hayden Book Company, Inc., 1982.

Leventhal, Lance. *6502 Assembly Language Programming.* Berkeley, California: Osborne/McGraw Hill, 1979.

Mansfield, Richard. A hidden world. *Computer's Gazette*, July 1983, pp. 84-86.

Papert, Seymour. *Mind Storms.* New York, New York: Basic Books, Inc., 1980.

PadMasters Guild

Chalk Board, Inc. is making a special offer to you as an owner of a PowerPad and the PowerPad Programming Kit. You are invited to become a member of the PadMasters Guild. For a full year, your membership is free. After that, the annual membership and renewal fee is $9.95. Your membership includes:

- Home delivery of quarterly publications containing new programs to run on your PowerPad. These publications serve as a clearing house for programs submitted by programmers like yourself.
- Home delivery of issues of the Chalk Board newsletter.
- Access to a special "Hot Line" telephone number which you can use for technical assistance.
- Special issues and releases about new product data from Chalk Board, Inc.

If you would like to become a member of the PadMasters Guild, complete this membership form and send it to:

Chalk Board, Inc.
3772 Pleasantdale Road
Atlanta, Georgia 30340

PadMasters Guild
Membership Form

Name_____

Address_____

City_____ State_____ Zip_____

Age_____

Type of Computer You Own/Use_____

Type of Storage Device Used

_____diskette _____cassette _____none

Purpose for which you use a Computer (check all that apply.)

_____Games _____Business _____Education

_____Word Processing _____Homes Finances

Product Design:
Ben Satterfield (author, educational applications)
Don Higgins (programming engineer)
David Carter (programming engineer)

Support:
Margaret Gorley (educational applications)
Tim Cope (programming engineer)
Margaret Walsh (editor)

Package & User's Guide Design:
Taylor & Taylor, Inc./Atlanta, GA

A touch of genius.

PowerLog™

CHALK BOARD INC.

DISK/TAPE NAME _____

SOFTWARE NAME _____

	DATE	NAME/MODE	DESCRIPTION
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			

Part No. 11004A

Chalk Board™ Warranty and Registration Card

A touch of genius.

Chalk Board, Inc.
LIMITED WARRANTY

Chalk Board, Inc., warrants this PowerPad™ touch-sensitive pad, connector cable, or software (including overlay, diskette or cassette and manual) against defects in material and workmanship for a period of 90 days from the date of purchase. This warranty is made only to the original consumer purchaser and only if such purchaser completes and returns the attached warranty registration card within 15 days after the date of purchase. Completion and return of the warranty registration card are conditions precedent to the effectiveness of this warranty.

If this product appears to be defective within 30 days after the date of purchase, return it to the retail store where it was purchased with your sales receipt and your retailer will replace your defective product with a new Chalk Board product.

If during the next 60 days the product appears to be defective, send the product, your name and address, a check or money order for $5.00 to cover mailing and handling, and complete description of the problem to: Customer Support Department, Chalk Board, Inc., 3772 Pleasantdale Rd., Atlanta, Ga. 30340. If Chalk Board, Inc. determines that the product is defective in either materials or workmanship, it will (at its option) repair or replace the product. Under no circumstances will Chalk Board, Inc. be responsible for damage resulting from improper use or installation of this product.

CHALK BOARD, INC., MAKES NO OTHER EXPRESS WARRANTIES AND LIMITS THE DURATION OF ALL IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MERCHANTABILITY, TO DURATION OF THE ABOVE WARRANTY (90 DAYS AFTER THE PURCHASE DATE). Some states do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you. CHALK BOARD, INC. WILL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OF THIS PRODUCT (EXCEPT FOR CONSEQUENTIAL DAMAGES FOR PERSONAL INJURY). Some states do not allow the exclusion or limitations of incidental or consequential damages, so the above limitation may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Chalk Board Warranty Registration Card:

Thank you for purchasing a Chalk Board product. To help us serve you better in the future, please take a few moments to complete this warranty registration card.

You must complete and return this card within 15 days after date of purchase to qualify for warranty protection.

Name: _____ Age: _____

Address: _____ City: _____

State: _____ Zip: _____

Country: _____

Telephone: _____

Product(s) Purchased: ☐ Chalk Board PowerPad™
☐ PowerPad™ software

Serial Number: _____

Date of Purchase: _____

Place of Purchase: _____ City: _____

The following questions are for market research purposes ONLY. PLEASE PROVIDE US WITH AS MUCH INFORMATION AS YOU FEEL IS APPROPRIATE.

Questions:

1. Do you or your family currently own a video game or computer? ☐ Video Game ☐ Computer ☐ Both

2. If yes, what type?
   Computer:
   ☐ Commodore ☐ VIC 20 ☐ 64
   ☐ Atari ☐ 400 ☐ 600 ☐ 800 ☐ 1200
   ☐ Apple ☐ II ☐ IIe
   ☐ IBM PC
   ☐ Texas Instruments 99/4A
   ☐ Other _____
   Game:
   ☐ Atari ☐ 2600 ☐ 5200
   ☐ Intellivision
   ☐ ColecoVision
   Other _____

3. How did you hear about Chalk Board?
   ☐ From a friend ☐ Used in school ☐ Newspaper article
   ☐ Magazine article ☐ Newspaper ad (specify) _____
   ☐ Magazine ad (specify) _____
   ☐ From my children _____
   ☐ Other (Please Specify) _____

(continued on other side)

4. Where did you purchase your Chalk Board product?

☐ Department Store ☐ Catalogue Showroom
☐ Discount Store ☐ Video/Stereo Store
☐ Toy Store ☐ Computer Store
☐ Business Equipment Dealer ☐ Other_____

5. User's age(s) (Please check all that apply):

☐ 0-6 yrs. ☐ 7-12 yrs. ☐ 13-18 yrs.
☐ 19-30 yrs. ☐ 30-50 yrs. ☐ Over 50

6. Product will be used by:

☐ Male(s) ☐ Female(s) ☐ Both

7. Annual Household Income:

☐ $15,000-30,000 ☐ $30,000-50,000 ☐ Over $50,000

8. Did you purchase any software when you purchased your PowerPad?

☐ yes ☐ no

9. If yes, how many packages?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 or more.

10. I would like to see more types of the following Chalk Board software:

☐ Visual Arts ☐ Music ☐ Math
☐ Science ☐ Language Arts ☐ Games
☐ Social Studies ☐ Sports ☐ Other_____

11. General Comments:_____

_____

_____

_____

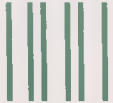
_____

_____

_____

Please detach and send this warranty card in the enclosed envelope.

Part No. 11002A



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 13575 ATLANTA, GA.

POSTAGE WILL BE PAID BY ADDRESSEE

Chalk Board, Inc.
3772 Pleasantdale Road
Atlanta, Georgia 30340

Part No. 11002A