THE INFRARED TOUCH-PAD

ENG 421 MANUAL

Written by Gregory F. Welch
Submitted to Professor Norman Harris

February 26, 1986

THE INFRARED TOUCH-PAD

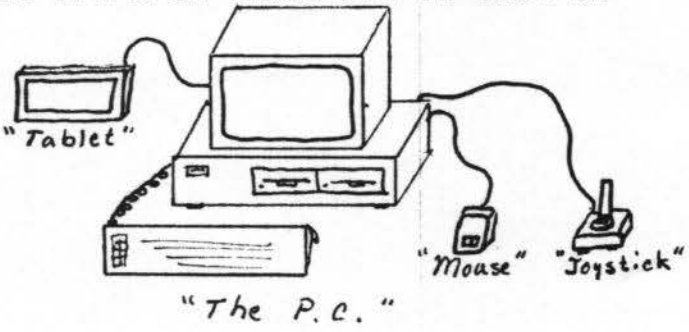
TABLE OF CONTENTS

THE INFRARED TOUCH-PAD

FIGURE LIST

INTRODUCTION

Since the introduction of the personal computer in the late 1970's, there has been a growing need for different types of input devices. Devices such as the joystick, the mouse, and the graphics tablet have made great advances in recent years. Within just the past year, however, came the introduction of several types of touch-sensitive pads. The Infrared Touch-Pad is such an input device.



This manual is designed to guide the user in the setup and operation of the Infrared Touch-Pad. It contains step-by-step instructions which will allow the user to interface the touch-pad to virtually any computer.

First of all, this manual will provide you with some background in the theory of operation. Secondly, it will guide you through the electrical connections required to interface the touch-pad to a computer. Thirdly, it will describe the software needed for normal operation, and give examples and suggestions for use. Finally, it will provide a guide for troubleshooting, in the event that the system should fail.

THEORY OF OPERATION

The touch-pad uses several infrared light beams to detect an obstruction on the pad. These beams are transmitted from one side of the pad to the other, where they are then detected by infrared receivers. Therefore, if you place an object such as a finger on the pad, a beam is broken and the computer can detect that the pad is being touched.

The pad has 16 rows and 16 columns of transmitter-receiver pairs (see figure 1 on the next page). This creates a 16 by 16 grid giving you 256 seperate locations which you can monitor with the computer.

With software you will write, you will output a six bit word which selects a light beam from one of the 16 rows or 16 columns. Then, you can look at the return line from the pad to see if that selected light beam is being broken. Later in this manual, you will see how this process can be implemented into a software loop which will continuously repeat a check of all 256 locations on the touch-pad.

Once this software loop is complete and functioning, you can define it as a ***scan subroutine***. Now, any time you wish to see if the pad is being touched, one call to the subroutine will return to you the location of the touch (if there is one). For example, if you wanted to write your own computer game, you could use the touch-pad for an input device. Imagine a game where you control a helicopter by moving your hand around on the surface of the touch-pad! The possibilities are limited only by your imagination.

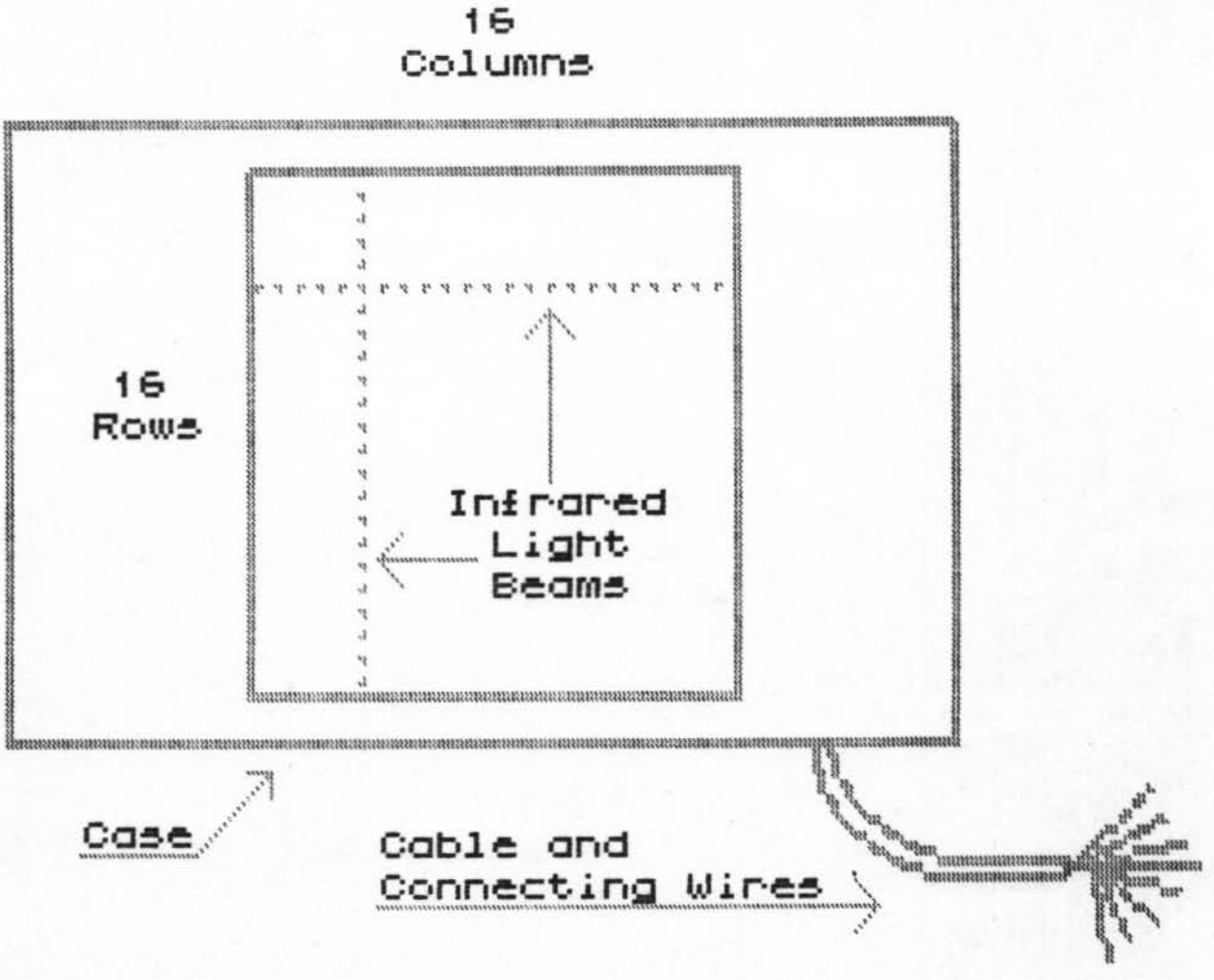


Figure 1: The Infrared Touch-Pad

SETUP INSTRUCTIONS

The following 11 steps will guide you through the initial one-time setup of the touch-pad. All of the steps should be read carefully before attempting to make any electrical connections. As each step is completed, a check mark should be placed in the appropriate spot next to each step (see figure 2 on the next page).

[✓] Check your work, it will pay off!

[ ] 1) Begin by identifying the following nine individual wires from the touch-pad:

	COLOR	BIT #	NAME	FUNCTION
[ ]	BLACK	– B_0 (Bit 0)	2^0 Select	Output
[ ]	BROWN	– B_1 (Bit 1)	2^1 Select	Output
[ ]	RED	– B_2 (Bit 2)	2^2 Select	Output
[ ]	ORANGE	– B_3 (Bit 3)	2^3 Select	Output
[ ]	YELLOW	– B_4 (Bit 4)	Row Select	Output
[ ]	GREEN	– B_5 (Bit 5)	Column Select	Output
[ ]	PURPLE	–	Return	Input
[ ]	GREY	–	Ground	Power
[ ]	WHITE	–	+5 Volts	Power

[ ] 2) Locate the parallel ports in the computer (see computer manual if necessary). Identify the ports to be defined as both output (six bits needed), and input (one bit).

[ ] 3) Connect the BLACK wire (2^0 select) to bit 0 of the computer port to be defined for ***output***.

[ ] 4) Connect the BROWN wire (2^1 select) to bit 1 of the same port.

[ ] 5) Connect the RED wire (2^2 select) to bit 2 of the same port.

[ ] 6) Connect the ORANGE wire (2^3 select) to bit 3 of the same port.

[ ] 7) Connect the YELLOW wire (2^4 select) to bit 4 of the same port.

[ ] 8) Connect the GREEN wire (2^5 select) to bit 5 of the same port.

[ ] 9) Connect the PURPLE wire (Return) to bit 0 of the computer port to be defined for ***input***.

[ ] 10) Connect the GREY wire (Ground) to any *ground* terminal of the computer.

[ ] 11) Connect the WHITE wire (+5 Volts) to any *+5 Volt* terminal on the power supply of the computer.

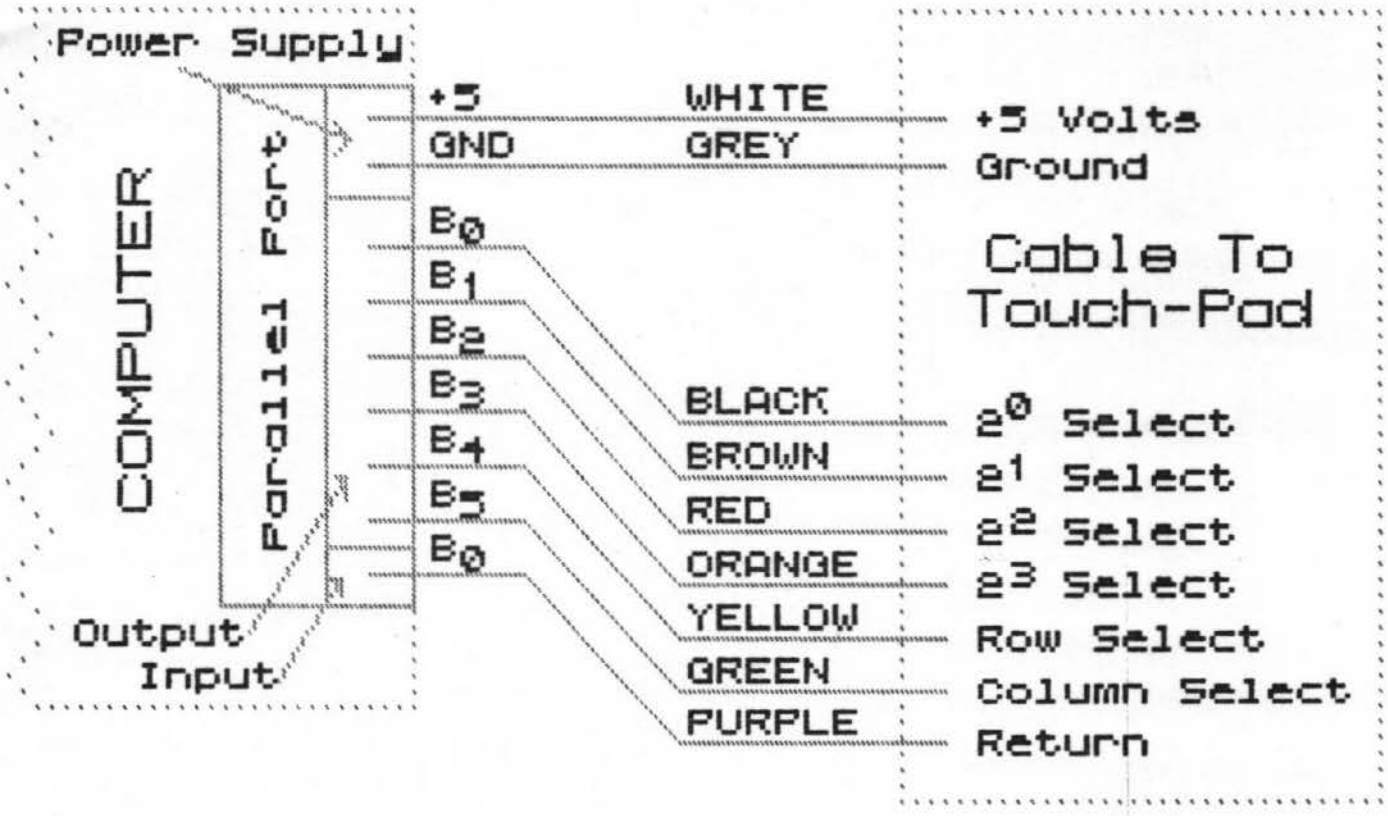


Figure 2: Touch-Pad Connections To Parallel Port

TESTING THE HARDWARE

To test the hardware connections to the touch-pad, you must write some simple software. This is most easily done with an ***interpreter*** based language such as BASIC. Throughout this manual, BASIC computer code will be used in examples. However, with slight changes, any language could be used to accomplish the same objectives.

But why do we need to do all of this?

The purpose of this hardware testing is to see if the touch-pad is connected correctly, so that you can use software without worrying about the hardware. This section will describe how to use the computer to send a beam from one side of the pad to the other. In the future, this process will be referred to as "turning on" a row or a column. To "turn on" a row or column can be compared to turning on a flashlight. The only differences are first of all with the touch-pad you must instruct the ***computer*** to "turn on" the beam. Secondly, once it's on, ***you can't see it.***

To "turn on" a row or column, you must send a number to the parallel port which is being used for output. This number is then interpreted by the touch-pad to turn on the correct beam. Once the beam is on, you can input a value (with software), from the port defined as input. This value will reflect the status of the RETURN wire. If the value returned is 1, then the beam is being broken. Likewise, if the beam was transmitted and received without interruption, the returned value will be 0.

As mentioned above, you must send a number to the parallel port defined as output. But what should this number be? By following the simple guidelines below, any of the 16 rows or columns can be "turned on".

DESIRED BEAM	***NUMBER SENT TO OUTPUT PORT***	
n^{th} Row	31 + n	(1 < n < 16)
n^{th} Column	63 + n	(1 < n < 16)

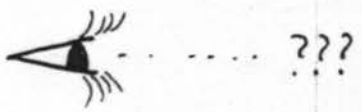
EXAMPLES:

1) To "turn on" the first row, send 31 + 1 or 32
2) ..... sixth row, send 31 + 6 or 37
3) ..... second column, send 63 + 2 or 65
4) ..... sixteenth column, send 63 + 16 or 79

To use BASIC to send these values to the touch-pad, you must first determine the corresponding numbers of the ports defined as input and output (refer to your computer manual to determine the port numbers you are using). Remember, there are six wires connected to the port defined for output, and one wire connected to the input port.

Now you will see how to use BASIC to send the values in the above examples. To do this, you must first be in the BASIC interpreter (see your computer manual if necessary). Once in the BASIC interpreter, try typing the following commands.

ABOVE EXAMPLES USING BASIC:



YOU TYPE

1) OUT outport,32 <ENTER> (Turns on the first row)
2) OUT outport,37 <ENTER> (Turns on the sixth row)
3) OUT outport,65 <ENTER> (Turns on the second column)
4) OUT outport,79 <ENTER> (Turns on the sixteenth column)

(Where ***outport*** is the number of the port defined as output)

Now what happened? Did you see anything? Of course not! All you did was turn on several different infrared beams. Remember, ***you can't see these beams.*** To see if the pad is working, you need to check the status of the RETURN line. This can be done by using BASIC to input a value from the port with the RETURN wire connected to it (the port defined as input).

Now, you should have just "turned on" the sixteenth column. By following the examples given below, you should now finally be able to see some results. Before proceeding, make sure that there is nothing on the touch-pad. You want to test the beam ***without*** an obstruction first.

USING BASIC TO DETERMINE BEAM STATUS:

```
  YOU TYPE
/-----------\
A=INP(inport) <ENTER>   (Places RETURN status in the variable A)
PRINT A       <ENTER>   (Prints either 1 or 0 on the screen)
```

(Where ***inport*** is the number of the port defined as input)

After typing the two statements shown above, you should have seen a 0 displayed on the screen. Once again, this means that the beam was not obstructed. If some number other than 0 was printed, refer to the section called Troubleshooting.

Now for some fun! Remember, the sixteenth column is still turned on. Find an object such as pencil, and position it on the touch-pad so that it is in the sixteenth column. The columns are numbered from 1 to 16, from left to right. They are divided by the 15 vertical lines marked on the pad. Once again, try typing the same statements as you just did above.

Now what number was printed on the screen? You should now see a 1, where as before there was a 0. Of course this means that the beam ***was*** obstructed! If a 0 appeared again, then try choosing a larger object than the pencil, and make sure that you are really placing it in the sixteenth column. If you still can't get a 1 to appear, refer to the section called Troubleshooting. Otherwise, if all seems well, you are now ready to proceed to the next section.

DEVELOPING A STATUS SUBROUTINE

As mentioned earlier, the process of scanning the pad is done by sending a light beam from one side, and checking to see if it was received at the other. We will now start to show you how you can use the touch-pad to do what you really want. Recalling the previous section, it should be noted that only one beam can be transmitted at a time; e.g. the first column, and ***then*** maybe the third row. In other words, the first row and the third column can ***not*** be "turned on" at the same time.

Because of this restriction, we must develop a system to continuously check all of the beams, one at a time. To assist you in doing this, it is suggested that you use a ***subroutine*** to do an individual beam check for you. This way, one call to this subroutine can return the status of the beam. This will eliminate a lot of redundancy. An example of a BASIC subroutine is given on the next page. Try typing it in, and we will then test it.

EXAMPLE OF A BASIC SUBROUTINE:

```
10000 REM This subroutine will use the variable BEAM to
          turn on a beam, and will return the status in
          the variable STATUS
10010 OUTPORT=100 REM Use your output port number
10020 INPORT=101 REM Use your input port number
10030 OUT OUTPORT,BEAM
10040 REM This turns on the beam, where the value of BEAM
          was determined as at the top of page 5.
10050 STATUS=INP(INPORT)
10060 REM This places the status of the beam in the
          variable STATUS.  A 0 means that it is
          unobstructed, a 1 means it is blocked.
```

Now, you have just created a subroutine which you can call at any time to determine the status of any beam. Try typing the following statements to test your subroutine:

```
  YOU TYPE
 /----------\
BEAM=79       <ENTER>  (Remember, 79 turns on the sixteenth column)
RUN 10000     <ENTER>  (This calls your subroutine)
PRINT STATUS <ENTER>  (This prints the beam status)
```

Now, if you still have an obstruction in the sixteenth column, you should have a 1 on the screen. Try removing the obstruction (pencil or whatever), and re-type the above statements. Did the 1 change to a 0? It should have! Congratulations, you are now ready to write a second ***scan*** subroutine. This subroutine can use the previous one to scan the entire touch-pad for you!

DEVELOPING A SCAN SUBROUTINE

Now, if you were going to try to explain to someone (in plain English) how you would scan the touch-pad, how would you explain it? You might end up saying something generally like this:

1) "Check all sixteen rows using the ***status subroutine.***"
2) "If none of the row beams were broken, repeat the check of all sixteen rows as in step 1."
3) "If any row beam was broken, remember the row number."
4) "Now check all sixteen columns, again using the ***status subroutine.***"
5) "If by chance there is no column beam being broken, return to step 1 and start all over."
6) "If any column beam was broken, remember the column number."
7) "Now that we know the pad was touched, print the location on the screen. Maybe we could print the row and the column of the touch."

Did that seem to make sense? If not, try to pretend that you are the computer, and try to choose a logical algorithm. You should arrive at about the same steps. Appendix B contains a sample BASIC ***scan subroutine***. This subroutine was written by translating those seven steps into actual BASIC code.

CLOSING COMMENTS

As mentioned earlier, the BASIC language is not the only computer language which can utilize the touch-pad. By consulting a book on the language you wish to use, you should have no trouble converting the sample program in appendix B. In fact, ***most*** computer language books use BASIC code in their program examples.

Remember, the sky is the limit! Use the touch-pad for games, graphics, control, or whatever you can dream of.

TELEPHONE TECHNICAL SUPPORT

Should you encounter any problems which are not addressed in this manual, all ***registered*** users are entitled to telephone support. This technical support is available at a charge of $45.00 per hour, with a one hour minimum. When you need to call, please have the following information at hand:

1) Your name as it appears on the warranty registration.
2) Your billing address.
3) The touch-pad registration number (on the warranty card).
4) The brand and model of computer you are using.
5) A description of the problem; be as specific as possible.

(317) 247-4919

APPENDIX A

TROUBLESHOOTING

##	SYMTOM	POSSIBLE CAUSE	SOLUTION	SEE ALSO
1)	Status is always 0.	Wrong beam select number.	See SETUP.	2
2)		Wrong input port number.	Check computer manual.	3
3)		Wrong output port number.	(Same)	4
4)		Bad wiring connection.	See SETUP.	5
5)		Defective touch-pad.	See authorized dealer.	6
6)		Defective computer port.	See computer manual.	
7)	Status is always 1.	Wrong input port number.	Check computer manual.	8
8)		Wrong output port number.	(Same)	9
9)		Defective touch-pad.	See authorized dealer.	
10)	One location always appears touched.	Defective touch-pad.	See authorized dealer.	
11)	Software returns a row or column greater than 16.	Wrong input port number.	Check computer manual.	12
12)		Input port being used by other devices.	Select a different input port.	13
13)			Use a software mask, see computer manual.	

APPENDIX B

Sample BASIC Scan Subroutine

```
5000  REM ********************************
5010  REM * Touch-Pad Scan Subroutine
5020  REM ********************************
5030  REM Scan all 16 rows to detect a touch
5040  FOR ROWNUMBER=1 TO 16
5050  BEAM=31+ROWNUMBER
5060  GOSUB 10000        REM Check status of rownumber
5070  IF STATUS=1 THEN GOTO 5100    REM Check columns
5080  NEXT ROWNUMBER     REM No touch, check next row
5090  GOTO 5030     REM Rows scanned, no touch; repeat
5100  REM Scan all 16 columns to detect a touch
5110  FOR COLNUMBER=1 TO 16
5120  BEAM=63+COLNUMBER
5130  GOSUB 10000        REM Check status of colnumber
5140  IF STATUS=1 THEN GOTO 5170    REM Print location
5150  NEXT COLNUMBER     REM No touch, check next column
5160  GOTO 5030     REM Col's scanned, no touch; repeat
5170  REM Print location of touch, and return from sub.
5180  PRINT "TOUCH AT ROW: ";ROWNUMBER
5190  PRINT "       COLUMN: ";COLNUMBER
5200  RETURN     REM Return from subroutine
5210  REM ********************************
5220  REM * END Touch-Pad Scan Subroutine
5230  REM ********************************
10000 REM This subroutine will use the variable BEAM to
          turn on a beam, and will return the status in
          the variable STATUS
10010 OUTPORT=100 REM Use your output port number
10020 INPORT=101 REM Use your input port number
10030 OUT OUTPORT,BEAM
10040 REM This turns on the beam, where the value of BEAM
          was determined as at the top of page 5.
10050 STATUS=INP(INPORT)
10060 REM This places the status of the beam in the
          variable STATUS.  A 0 means that it is
          unobstructed, a 1 means it is blocked.
10070 RETURN     REM Return from subroutine
```

APPENDIX C

STANDARDS & SPECIFICATIONS

Input Grid Size	16 x 16 (256)
Grid Resolution	0.5 cm
Supply Voltage	5 Volts
Supply Current	75 milliamps
I/O Bit Requirements	6 output, 1 input
I/O Signal Levels	TTL Logic Levels
	> 3.5 Volts = "1"
	< 1.5 Volts = "0"

APPENDIX D

GLOSSARY

ALGORITHM A procedure for solving a problem.

BINARY Having two values or states. The binary number system has two digits, 1 and 0.

BIT Binary digit. A 1 or a 0.

BYTE A group of eight bits.

HARDWARE Computer equipment such as the computer itself, a printer, a circuit board, or any mechanical (physical) components.

INFRARED LIGHT Light having wavelengths greater than that of visible light, but shorter than microwaves. This light can not be seen by the human eye.

INTERPRETER A computer language interface which translates each statement when it is encountered in real time, requiring no compiling of the code.

I/O A term refering to input and output in a computer.

MASK A technique used to remove unwanted data from a variable.

PARALLEL PORT A computer interface which transfers binary information in byte form; eight bits at a time.

SERIAL PORT A computer interface which transfers binary information in bit form; one bit at a time.

SOFTWARE Any computer program code (such as BASIC code).

SUBROUTINE A section of computer software which is set aside from the main section of code, and can be used at any time by simply jumping to it. Such code is characterized by a required RETURN statement which resumes processing imediately following the call.

WORD A group of bits representing a complete piece of digital information.