Gadget Freak Case #162

Magic Fingers Tickle the Ivories

Imagine, put on a glove and you can play the piano. Students John Bean, Hannah Hudson, Casey O'Farrell, and Ryan Slinger created such a glove that detects the bend of a finger and "plays" a corresponding piano note. A button on the glove shifts between five high and low keys. As Ingrid Bergman might have said in Casablanca, "Play it, Sam. Play 'The Gadget Freak Boogie.'"

The Design

The Magic Piano and electronic glove (Figure 1) lets you play an electric piano without touching the keys. Instead, you simply wear a glove that detect when you bend one or more fingers. Once a finger bends, a signal specific to that finger is sent wirelessly to the piano. When the piano receives the data, it plays the note for the key assigned to the bent finger. The piano can play 10 notes with one glove. When the "pianist" pushes a shift button, the piano plays the lower five notes, otherwise, the piano plays five higher notes.

Figure 1: The Magic Piano.
System Details

The Magic Piano comprises two subassemblies; the glove (Figure 2) and an electronic keyboard (Figure 3). The glove contains five bend sensors (one located on each finger) and the normally-open shift button. The bend sensors act like potentiometers. The more a finger bends, the higher the resistance of the finger's sensor. The sensors connect to a Microchip Technology PIC microcontroller (MCU) that continuously reads the state of the bend sensors and the button. When this MCU--mounted on the glove--detects a bent finger, it transmits wireless information to a second MCU within the piano. The wireless signal from the hand continuously communicates with the piano, so as long as a finger stays bent, the note will play. This capability provides nearly instant reaction of the piano to finger bending.

The second MCU process the information from the glove and plays the key that corresponds to the bend finger. A pianist can bend more than one finger at a time and thus play more than one note at a time.

To each piano key, we attached a linear actuator, basically a solenoid, that employed an inductor coil wound on an open core. An iron rod sits part way into the inductor's core. To play a note, the piano MCU drives current through the coil to pull down the iron core and "play" a note. To switch a high current through each coil, we used a relay controlled by the MCU. (Schematic diagrams and PICBasic code listings appear later in this document.)

The chart in Figure 4 shows the simple flow of the Magic Piano software. Upon start up, the character-output LCD offers a welcome screen (Figure 5) while the graphics LCD draws a treble clef, music staff, and notes. After the piano plays its start-up sequence, the character LCD tells the player he or she can begin to play.

While the piano plays, a solid-state temperature sensor continually monitors the temperature inside the piano box. As long as the temperature stays below a set threshold, the piano will operate. However when the temperature sensor's signal exceeds the set point, the MCU turns off all signals going to the relays that control the solenoids. Then, the character LCD displays a warning message (Figure 6) that tells the player to wait for the system to cool. When the temperature drops below the threshold, the MCU resumes actuating the keys.
The functional diagram (Figure 7) illustrates how the Magic Piano and its major components interact. All of the manual inputs and the wireless transmitter connect to the hand MCU (PIC Microcontroller 1). The rest of the system operates within the piano. A third PIC MCU (PIC Microcontroller 3) controls the graphics LCD. The wireless receiver, inductors, thermistor, and character LCD are all handled by the remaining MCU (PIC Microcontroller 2). The speaker operates directly from the piano and requires no MCU control.

Figure 2: The glove with a bend sensor on each finger and the wireless MCU board attached.
Figure 3: Open view of the Magic Piano. Note the solenoids and the wires that actuate individual keys.
Software Flowchart

Start

- declare variables and constants

- initialize the valid combination and the outputs

Loop

- Read the states of the 5 fingers and the shift button

- Send info to the receiver PIC

- Check temperature sensor for over heating in piano
  - Hot: Turn off solenoids and display error message on character LCD
  - Cold: Turn on the corresponding solenoid(s)

- Display the note(s) being played on the graphic LCD

Figure 4: Software flowchart for the Magic Piano.
Figure 5: The Welcome Screen shown upon startup.

Figure 6: Temperature error message.
Figure 7: Functional diagram of the Magic Piano.
Design Process

Solenoids

We used a solenoid and an iron plunger to actuate each of the 10 piano keys. The coils provided consistent, repeatable linear motion to pull a wire for each key. Through tests on the solenoids we determined how much voltage the coils needed to create a force strong enough to play a key. We added two small, strong magnets to the top of each iron plunger to increase the attractive force between the magnetic field of the coil and the field of the plunger.

Finger Sensors

We used tape and sewing to attach the sensors to the glove fingers. Double-sided tape attached the sensors to the tops of the fingers, and then we used thread and basting stitches (single thread, long stitch) to hold down the sensors for a tight, secure fit.

Figures 8 illustrates the concept of using the glove to control piano keys.
Figure 8: Concept Design of the Magic Piano.

Displays

The character LCD receives serial commands from PIC Microcontroller 2 and displays a "splash screen" welcome message, the notes currently being played, or a warning message when the temperature inside the piano gets too high. The LCD module had a fixed data rate of 9600 bits/sec (bps).

The graphics LCD receives serial commands from PIC Microcontroller 3, which we dedicated to this LCD. An ATMega168 within the LCD module controls the display and displays a treble clef, music scale, and some 8th notes upon startup. We aimed to have this LCD display the notes being played, but problems prevented a full implementation. As a result, a builder might choose to not use the graphics LCD.
The graphics LCD presented another challenge. We set up PIC Microcontroller 3 to send data at 9600 bits/sec but the LCD operated at 115,200 bps. The PIC will not operate at that high a baud rate. We were lucky enough to acquire a developer board that let us change the bit rate of the graphics LCD to 9600.

Adding the wireless component to the PIC to PIC interface presented of a challenge that we expected, perhaps due to the long serial commands that the system transmitted. Sending a continuous wireless signal helped eliminate this problem.

Figure 9: Solenoid coil and iron plunger.
Figure 10: Magnet used to create a magnetic field on each solenoid plunger to increase solenoid actuation force.

Figure 11: Bend sensor.

Lessons Learned

MCU-to-MCU wireless communications

Just because two PIC MCUs can connect via a 1-wire serial link does not mean they can communicate flawless information over a wireless link. The code for a "wired" connection can send the information once and the receiving PIC MCU will pick it up without a problem. For wireless communication, we had to send the data must several times so the receiver would have a better
chance of picking up the data reliably. We used a for-next loop to transmit wireless data three times. When we only sent data only once, the receiver picked up only some of the info some of the time.

Memory Page Boundary:

When writing code for the PIC16F916 that requires a lot of memory, we sometimes got “crossing memory page boundary” messages from the compiler. This results when a program crosses a memory-page boundary. This limitation applies to the PIC14 and PIC16 MCU families, but not to the PIC18 family. Some compilers automatically handle the memory-page-crossing limits.

PIC Basic POT Command

The bend sensors on the glove were easy to connect to PIC Microcontroller 1, but the PICBasic POT command that reads a potentiometer value (resistance) proved difficult to use with the sensors. One side of each sensor connects to ground and the other connects to an input on the PIC MCU. The POT command should measure the change in resistance that occurred in each finger. Each sensor had a different resistance when straight and when bent over the finger's range of motions. So, we had to calibrate each sensor so that the fingers provided consistent and comparable resistance values.

Temperature Sensor

Use the active-low logic OUT signal from the TC622 temperature sensor to indicate an over-temperature condition. The sensor uses an external resistor to set the temperature threshold. Use the formula provided in the TC622/TC624 data sheet from Microchip to set the temperature "trip point." This sensor includes hysteresis, so it will trip when the temperature reaches T and will return to normal when the temperature goes below T-2 degrees C. The data sheet formula uses temperature units of Kelvins. Find the data sheet at: http://ww1.microchip.com/downloads/en/DeviceDoc/21440C.pdf

Fine tuning the potentiometer adjustment while system running is very difficult. Once you change the resistance to pass the trip temperature, you cannot turn it back and un-trip the sensor. You must turn off the system, change the resistance, then test again until you get it just right.

Table 1. Parts List for Allied Electronics
<table>
<thead>
<tr>
<th>Amt</th>
<th>Part Description</th>
<th>Allied Part #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PIC16F916 Microcontroller</td>
<td>383-0503</td>
</tr>
<tr>
<td>2</td>
<td>PIC16F88 Microcontroller</td>
<td>383-0496</td>
</tr>
<tr>
<td>10</td>
<td>R40-11D2-5 Relay, NTE Electronics</td>
<td>235-0036</td>
</tr>
<tr>
<td>5</td>
<td>0.1 μF 50V Capacitor</td>
<td>507-0211</td>
</tr>
<tr>
<td>10</td>
<td>1N4004 Diode</td>
<td>266-0004</td>
</tr>
<tr>
<td>2</td>
<td>L7805CV 5V Regulator</td>
<td>248-0415</td>
</tr>
<tr>
<td>1</td>
<td>L7809CV 9V Regulator</td>
<td>248-0671</td>
</tr>
<tr>
<td>4</td>
<td>1000Ω Resistor 1/4W</td>
<td>648-1443</td>
</tr>
<tr>
<td>1</td>
<td>1000Ω Trimmer Resistor 1/2W</td>
<td>372-0068</td>
</tr>
<tr>
<td>1</td>
<td>Miniature Pushbutton, N.O.</td>
<td>676-0219</td>
</tr>
<tr>
<td>1</td>
<td>Toggle Switch, SPDT</td>
<td>676-3000</td>
</tr>
</tbody>
</table>

Table 2. Parts from Other Suppliers

<table>
<thead>
<tr>
<th>Amt</th>
<th>Description</th>
<th>Vendor</th>
<th>Part #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yamaha electric keyboard, PSS-50</td>
<td>PSS-50</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Solenoid Coil and Iron Core Rod</td>
<td>C&amp;S Sales</td>
<td>6SC N3</td>
</tr>
<tr>
<td>10</td>
<td>Solenoid Iron Plunger</td>
<td>C&amp;S Sales</td>
<td>6SC M3B</td>
</tr>
<tr>
<td>20</td>
<td>Magnet, NdFeB</td>
<td>Force Field</td>
<td>Model: 0003</td>
</tr>
<tr>
<td>1</td>
<td>TC622CPA Temperature Sensor, 8 pin DIP</td>
<td>Microchip</td>
<td>TC622CPA</td>
</tr>
<tr>
<td>1</td>
<td>WRL-08945 Wireless Transmitter, 315 MHz</td>
<td>Sparkfun</td>
<td>WRL-08945</td>
</tr>
<tr>
<td>1</td>
<td>WRL-08947 Wireless Receiver, 315 MHz</td>
<td>Sparkfun</td>
<td>WRL-08947</td>
</tr>
<tr>
<td>1</td>
<td>Graphics LCD, 128x64 pixel</td>
<td>Sparkfun</td>
<td>LCD-09351</td>
</tr>
<tr>
<td>1</td>
<td>Character LCD, 16x2</td>
<td>Sparkfun</td>
<td>LCD-09395</td>
</tr>
<tr>
<td>5</td>
<td>Bend Sensor</td>
<td>Sparkfun</td>
<td>SEN-08606</td>
</tr>
</tbody>
</table>
Appendix

The Hand Circuit

Figure 12: A photo of the hand circuit, as mounted on the glove.
Figure 13: Hand Circuit Wiring Diagram
'Code for PIC16F88 in the hand circuit:

'Identify & set oscillator clock speed to 8 MHz
DEFINE OSC 8
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1

Include "BS2DEFS.BAS" 'needed to allow use of mode names & other syntax for serout/serin

'Turn off A/D converter:
anzel = 0

'Define pin assignments, variables, constants:

finger0  var  PORTB.0  'assign index finger to PORTB.0
finger1  var  PORTB.1  'assign middle finger to PORTB.1
finger2  var  PORTB.2  'assign ring finger to PORTB.2
finger3  var  PORTB.3  'assign pinky finger to PORTB.3
finger4  var  PORTB.4  'assign thumb finger to PORTB.4
out      var  PORTA.0  'send data out on PORTA.0
glove_status  var  byte[8]   'variable to store status of glove (not shifted)
glove_status2  var  byte[8]   'variable to store status of glove (shifted)
SCALE0    con  52   'scaling factors for pot_val on each finger
SCALE1    con  63   'since resistance of each sensor was not the same,
SCALE2    con  54   'each sensor needed a different scaling factor to
SCALE3    con  53   'return consistent and comparable pot values
SCALE4    con  70
pot_val   var  byte   'variable for storage of pot value
shift_button  var  PORTB.5  'assign shift button to PORTB.5
i         var  byte   'variable to control # of times data is sent

'Initialize the I/O pins:
TRISA = %00000000
TRISB = %00111111
inputs, rest as outputs
low out
low shift_button
pot_val = 0
pause 500

WHILE (1)

low shift_button
\reset state of shift button to low

Treset glove status storage variable to zero

for all bits
glove_status = %00000000
for all bits
glove_status2 = %00000000
for all bits
POT, finger0, SCALE0, pot_val
constant of RC circuit,
if (pot_val > 220 && shift_button = 0) then
variable "pot_val"
glove_status0 = 1
not pressed,
glove_status2.0 = 0
corresponding shifted bit low
else
if (pot_val > 220 && shift_button = 1) then
pressed set corresponding,
glove_status2.0 = 1
shifted bit low
else
make sure both
glove_status0 = 0
endif
endif
glove_status2.0 = 0
glove_status.0 = 1
endif

pot_val = 0
not stored properly finger,
key won't get stuck down
POT, finger1, SCALE1, pot_val
if (pot_val > 220 && shift_button = 0) then
glove_status1 = 1
else
if (pot_val > 220 && shift_button = 1) then
glove_status2.1 = 1
else
glove_status2.1 = 0
endif
glove_status.1 = 0
defined
endif

pot_val = 0

POT, finger2, SCALE2, pot_val
if (pot_val > 220 && shift_button = 0) then
glove_status2.2 = 1
else
if (pot_val > 220 && shift_button = 1) then
glove_status2.2 = 1
else
glove_status2.2 = 0
endif
glove_status.2 = 0
defined
endif
pot_val = 0

POT finger3, SCALE3, pot_val
if (pot_val > 220 && shift_button = 0) then
  glove_status.3 = 1
  glove_status2.3 = 0
else
  if (pot_val > 220 && shift_button = 1) then
    glove_status2.3 = 1
    glove_status.3 = 0
  else
    glove_status.3 = 0
    glove_status2.3 = 0
  endif
endif

pot_val = 0

POT finger4, SCALE4, pot_val
if (pot_val > 220 && shift_button = 0) then
  glove_status.4 = 1
  glove_status2.4 = 0
else
  if (pot_val > 220 && shift_button = 1) then
    glove_status2.4 = 1
    glove_status.4 = 0
  else
    glove_status.4 = 0
    glove_status2.4 = 0
  endif
endif

pot_val = 0

so that wireless
for i = 0 to 3
  picking up the signal
  SEROUT out,T2400, ["AB",glove_status]
  variable to be stored
next i
pause 5
status2" three times, so that
for i = 0 to 3
  chance of picking up the signal
  SEROUT out,T2400, ["CD",glove_status2]
  variable to be stored
next i
the same for the receiving pic

WEND

END
The Piano Circuit

Figure 14: This photo of the piano circuit shows the solenoids, relays, and microcontroller placed in solderless breadboards. Note the green wireless receiver in the foreground.
Figure 15: Piano Circuit Wiring Diagram
'Code for PIC16F916 in the piano circuit:

'Identify & set oscillator clock speed to 8 MHz
DEFINE OSC 8
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1

Include "BS2DEFS.BAS" 'needed to allow use of mode
names and other syntax for 'serout/serin commands

'Turn off A/D converter:
anse1 = 0

'Define pin assignments, variables, constants:
in var PORTC.4 'input from 16f88 on hand circuit
glove_status var byte[8] 'storage var for first five keys (not
shifted)
glove_status2 var byte[8] 'storage var for next five keys (shifted)
relay7 var PORTB.1 'relay to play index finger
relay8 var PORTB.0 'relay to play middle finger
relay9 var PORTC.7 'relay to play ring finger
relay10 var PORTC.6 'relay to play pinky finger
relay6 var PORTB.2 'relay to play thumb finger
relay5 var PORTB.3 'relay to play index finger (shifted)
relay4 var PORTB.4 'relay to play middle finger (shifted)
relay3 var PORTB.5 'relay to play ring finger (shifted)
relay2 var PORTB.6 'relay to play pinky finger (shifted)
relay1 var PORTB.7 'relay to play thumb finger (shifted)
LCD var PORTC.3 'LCD to display note played and
messages thermistor var PORTA.6 'temperature sensor

finger0 var relay7 'alias: finger0 corresponds to
relay/solenoid 7
finger1 var relay8 'alias: finger1 corresponds to
relay/solenoid 8
finger2 var relay9 'alias: finger2 corresponds to
relay/solenoid 9
finger3 var relay10 'alias: finger3 corresponds to
relay/solenoid 10
finger4 var relay6 'alias: finger4 corresponds to
relay/solenoid 6
finger0_2 var relay2 'alias: finger0_2 corresponds to
relay/solenoid 2
finger1_2 var relay3 'alias: finger1_2 corresponds to
relay/solenoid 3
finger2_2 var relay4 'alias: finger2_2 corresponds to
relay/solenoid 4
finger3_2 var relay5 'alias: finger3_2 corresponds to
relay/solenoid 5
finger4_2 var relay1 'alias: finger4_2 corresponds to
relay/solenoid 6
'Initialize the I/O pins:
TRISA = %01000000      'designate PORTA.6 as input, rest
as outputs
TRISB = %00000000      'designate all of PORTB as outputs
TRISC = %00010000      'designate PORTC.4 as input, rest
as outputs
low finger0       'initialize all outputs to
low finger0_2
low finger1       relay/solenoids as low
low finger1_2
low finger2       low LCD
low finger2_2
glove_status = %00000000    'start with glove status storage variable at
low finger3       zero (keys off)
glove_status2 = %00000000     'start with glove status2 storage variable at
low finger4       zero (keys off)

PAUSE 2000

'Display startup message:
SEROUT LCD,T9600, [254,1]    'command to clear LCD screen
SEROUT LCD,T9600, [254,2]    'command to move cursor to first character
of line one on LCD
SEROUT LCD,T9600, [" WELCOME TO THE"] 'command to move cursor to beginning of'
SEROUT LCD,T9600, [254,192]    'command to move cursor to beginning of'
next line
SEROUT LCD,T9600, [" MAGIC PIANO"]
PAUSE 2000

'Startup sequence: play all keys in order
high relay3 : pause 250 : low relay3
high relay4 : pause 250 : low relay4
high relay5 : pause 250 : low relay5
high relay6 : pause 250 : low relay6
high relay7 : pause 250 : low relay7
high relay8 : pause 250 : low relay8
high relay9 : pause 250 : low relay9
high relay10 : pause 250 : low relay10

pause 1000

'Indicate that the piano is ready to play:
SEROUT LCD,T9600, [254,1]
SEROUT LCD,T9600, [254,2]
SEROUT LCD,T9600, [" READY TO PLAY "]
PAUSE 1000
SEROUT LCD,T9600, [254,1]
SEROUT LCD,T9600, [254,2]

loop:
'check temperature
If (thermistor = 0) Then
  'Thermistor voltage = 0 when above temp
  trip point
  SEROUT LCD,T9600, [254,1]
  'If too hot, display warning
  message on LCD
  SEROUT LCD,T9600, [254,2]
  SEROUT LCD,T9600, [" TOO HOT! "]
  SEROUT LCD,T9600, [254,192]
  SEROUT LCD,T9600, [" NO MORE MAGIC! "]
  PAUSE 3000
  SEROUT LCD,T9600, [254,1]
  SEROUT LCD,T9600, [254,2]
  SEROUT LCD,T9600, ["PLEASE WAIT FOR "]
  SEROUT LCD,T9600, [254,192]
  SEROUT LCD,T9600, [" PIANO TO COOL "]
  PAUSE 10000
  'wait 10 seconds before checking thermistor
  again
  SEROUT LCD,T9600, [254,1]
  SEROUT LCD,T9600, [254,2]
  SEROUT LCD,T9600, [" CHECKING TEMP "]
  SEROUT LCD,T9600, [254,192]
  SEROUT LCD,T9600, [" AGAIN "]
  PAUSE 2000
  SEROUT LCD,T9600, [254,1]
  SEROUT LCD,T9600, [254,2]
  GOTO loop
  'check thermistor again
Else
  GOTO incoming
Endif
  'if temperature is under trip point, then jump
out of loop & go to line
'labeled "incoming", allowing 'piano to play
incoming:
  SERIN in,T2400, ["AB"], glove_status
  'wait for string "AB" and store next item in variable
  'glove_status to indicate status of
  first five fingers
  if (glove_status = %00010110) then
    'check for special finger combination
    gosub axel_f
    '"Axel F." and jump to that
    subroutine if correct finger
    goto incoming
    'combination is played
  endif
  if (glove_status = %00011100) then
    'check for special finger combination
    gosub jingle_bells
    "Jingle Bells" and jump to that
    subroutine if correct finger
    goto incoming
    'combination is played
  endif
  if (glove_status.0 = 1) then
    'start checking each bit of the transmitted
    binary #'s.
    high finger0
    'a 1 or high bit corresponds to that
    note being played,
SEROUT LCD,T9600, [254,2]  
'SEROUT LCD,T9600, [254,128 +8]  
'order to  
' SEROUT LCD,T9600, ["C"]  
'arranged on LCD in same order  
else  
' low finger0  
'sure the corresponding  
'SEROUT LCD,T9600, [254,2]  
'note played while leaving  
'SEROUT LCD,T9600, [254,128 +8]  
'on the LCD screen  
'SEROUT LCD,T9600, [" "]  
endif  
if (glove_status.1 = 1) then  
' repeat above process for each of the 5  
' high finger1  
'SEROUT LCD,T9600, [254,2]  
'SEROUT LCD,T9600, [254,128 +9]  
'SEROUT LCD,T9600, ["D"]  
else  
' low finger1  
'SEROUT LCD,T9600, [254,2]  
'SEROUT LCD,T9600, [254,128 +9]  
'SEROUT LCD,T9600, [" "]  
endif  
if (glove_status.2 = 1) then  
' high finger2  
'SEROUT LCD,T9600, [254,2]  
'SEROUT LCD,T9600, [254,128 +10]  
'SEROUT LCD,T9600, ["E"]  
else  
' low finger2  
'SEROUT LCD,T9600, [254,2]  
'SEROUT LCD,T9600, [254,128 +10]  
'SEROUT LCD,T9600, [" "]  
endif  
if (glove_status.3 = 1) then  
' high finger3  
'SEROUT LCD,T9600, [254,2]  
'SEROUT LCD,T9600, [254,128 +11]  
'SEROUT LCD,T9600, ["F"]  
else  
' low finger3  
'SEROUT LCD,T9600, [254,2]  
'SEROUT LCD,T9600, [254,128 +11]  
'SEROUT LCD,T9600, [" "]  
endif  
if (glove_status.4 = 1) then  
' high finger4
SEROUT LCD,T9600, [254,2]
SEROUT LCD,T9600, [254,128 +7]
SEROUT LCD,T9600, ["B"]
else
    low finger4
SEROUT LCD,T9600, [254,2]
SEROUT LCD,T9600, [254,128 +7]
SEROUT LCD,T9600, [" "]
endif

variable glove_status2
    SERIN in,T2400, ["CD"], glove_status2
fingers
of 2400, must be same as pic on hand circuit

'wait for string "CD" and store next item in
'to indicate status of second (shifted) five
'note: T2400 = baud rate

if (glove_status2.0 = 1) then
    high finger0_2
    SEROUT LCD,T9600, [254,2]
    SEROUT LCD,T9600, [254,128 +3]
    SEROUT LCD,T9600, ["E"]
else
    low finger0_2
    SEROUT LCD,T9600, [254,2]
    SEROUT LCD,T9600, [254,128 +3]
    SEROUT LCD,T9600, [" "]
endif

if (glove_status2.1 = 1) then
    high finger1_2
    SEROUT LCD,T9600, [254,2]
    SEROUT LCD,T9600, [254,128 +4]
    SEROUT LCD,T9600, ["F"]
else
    low finger1_2
    SEROUT LCD,T9600, [254,2]
    SEROUT LCD,T9600, [254,128 +4]
    SEROUT LCD,T9600, [" "]
endif

if (glove_status2.2 = 1) then
    high finger2_2
    SEROUT LCD,T9600, [254,2]
    SEROUT LCD,T9600, [254,128 +5]
    SEROUT LCD,T9600, ["G"]
else
    low finger2_2
    SEROUT LCD,T9600, [254,2]
    SEROUT LCD,T9600, [254,128 +5]
    SEROUT LCD,T9600, [" "]
endif

if (glove_status2.3 = 1) then
    high finger3_2
    SEROUT LCD,T9600, [254,2]
    SEROUT LCD,T9600, [254,128 +6]
SEROUT LCD,T9600, ["A"]  
else  
low finger3_2  
SEROUT LCD,T9600, [254,2]  
SEROUT LCD,T9600, [254,128 +6]  
SEROUT LCD,T9600, [" "]  
endif

if (glove_status2.4 = 1) then  
high finger4_2  
SEROUT LCD,T9600, [254,2]  
SEROUT LCD,T9600, [254,128 +2]  
SEROUT LCD,T9600, ["D"]  
else  
low finger4_2  
SEROUT LCD,T9600, [254,2]  
SEROUT LCD,T9600, [254,128 +2]  
SEROUT LCD,T9600, [" "]  
endif

GOTO loop  
sensor before  
'return to loop which will check temperature  
'allowing piano to continue to be

played  

axel_f:  
'play song "Axel F."
low finger1 : low finger2 : low finger4  
sequence  
'turn off the keys used to initiate the song

high relay5 : pause 125 : low relay5  
whole note  
pause 250  
is used for each half note  
high relay7 : pause 125 : low relay7  
quarter note  
pause 250  
62.5 in order to sound right,  
high relay5 : pause 125 : low relay5  
system)  
pause 125  
high relay5 : pause 75 : low relay5  
pause 75  
high relay8 : pause 75 : low relay8  
pause 125  
high relay5 : pause 125 : low relay5  
pause 125  
high relay4 : pause 125 : low relay4  
pause 125  
high relay5 : pause 125 : low relay5  
pause 250  
high relay9 : pause 125 : low relay9  
pause 125  
high relay5 : pause 125 : low relay5  
pause 250  
high relay5 : pause 75 : low relay5  
'pause 250 = 0.25 sec is used for each  
'a pause of 250/2 = 125 = 0.125 sec  
'a pause of 75 = 0.075 sec is used for each  
'(needed to be longer than 250/4 =  
'perhaps due to delays in the relay/solenoid
pause 75
high relay10 : pause 75 : low relay10
pause 125
high relay9 : pause 125 : low relay9
pause 125
high relay8 : pause 125 : low relay8
pause 125
high relay5 : pause 125 : low relay5
pause 125
high relay8 : pause 125 : low relay8
pause 125
high relay9 : pause 125 : low relay9
pause 125
high relay5 : pause 75 : low relay5
pause 75
high relay4 : pause 75 : low relay4
pause 125
high relay4 : pause 75 : low relay4
pause 75
high relay2 : pause 125 : low relay2
pause 125
high relay6 : pause 125 : low relay6
pause 125
high relay5 : pause 125 : low relay5
pause 125

glove_status = $00000000

'in the program
pause 500
RETURN

jingle_bells:
'play song "Jingle Bells"
low finger2 : low finger3 : low finger4
sequence

high relay6 : pause 150 : low relay6
whole note
pause 150
is for each half note
high relay6 : pause 150 : low relay6
for each quarter note
pause 150
high relay6 : pause 300 : low relay6
pause 300

high relay6 : pause 150 : low relay6
pause 150
high relay6 : pause 150 : low relay6
pause 150
high relay6 : pause 300 : low relay6
pause 300

high relay6 : pause 150 : low relay6
pause 150
high relay6 : pause 150 : low relay6
pause 150
'
clear the variable glove status, wait 0.5 sec, 'and return to the previous location
'turn off the keys used to initiate the song
'a pause of 300 = 0.30 sec is used for each
'a pause of 300/2 = 150 = 0.150 sec
'a pause of 300/4 = 75 = 0.075 sec is used
high relay8 : pause 150 : low relay8
pause 150
high relay4 : pause 150 : low relay4
pause 150
high relay5 : pause 150 : low relay5
pause 150
high relay6 : pause 400 : low relay6
pause 300
high relay7 : pause 150 : low relay7
pause 150
high relay7 : pause 150 : low relay7
pause 150
high relay7 : pause 150 : low relay7
pause 150
high relay7 : pause 150 : low relay7
pause 150
high relay7 : pause 150 : low relay7
pause 150
high relay7 : pause 150 : low relay7
pause 150
high relay6 : pause 150 : low relay6
pause 150
high relay6 : pause 150 : low relay6
pause 150
high relay6 : pause 75 : low relay6
pause 75
high relay6 : pause 75 : low relay6
pause 75
high relay6 : pause 150 : low relay6
pause 150
high relay5 : pause 150 : low relay5
pause 150
high relay5 : pause 150 : low relay5
pause 150
high relay5 : pause 150 : low relay5
pause 150
high relay5 : pause 150 : low relay5
pause 150
high relay5 : pause 400 : low relay5
pause 150
high relay8 : pause 600 : low relay8
pause 400

glove_status = $00000000

'clear the variable glove status, wait 0.5 sec,
and return to the previous location

in the program
pause 500
RETURN

END
The GLCD Circuit

Figure 16: Graphic LCD Circuit Wiring Diagram
'Code for PIC16F88 in the piano that controls the GLCD:

'Identify & set oscillator clock speed
DEFINE OSC 8
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1

DEFINE CHAR_PACING 1000

'PIC16F88

'Turn off the A/D converter
ansel = 0

'Define pin assignments, variables, constants:
LCD   var  PORTA.3  'output to LCD display

'initialize the I/O pins:
TRISB = %11111111  'designate all of PORTB
    as inputs
TRISA = %00000000  'designate all of PORTA
    as outputs

Include "BS2DEFS.BAS"

'main:
pause 2000  'wait 2 seconds for the
LCD to startup

'Draw Treble Clef
SEROUT LCD,T9600, [7C,0C,0A,06,0A,08,01]  'Sending "<control>l"
(S00C)" followed by two sets of
SEROUT LCD,T9600, [7C,0C,0B,04,0B,08,01]  '(x, y)
coordinates defining the line's start
SEROUT LCD,T9600, [7C,0C,06,0C,06,08,01]  'and stop,
followed by a 0 or 1 determines
SEROUT LCD,T9600, [7C,0C,03,0C,03,04,01]  'whether to draw
or erase the line
SEROUT LCD,T9600, [7C,0C,0D,02,0D,03,01]
SEROUT LCD,T9600, [7C,0C,0E,02,10,02,01]
SEROUT LCD,T9600, [7C,0C,11,02,11,03,01]
SEROUT LCD,T9600, [7C,0C,12,03,12,04,01]
SEROUT LCD,T9600, [7C,0C,13,04,13,05,01]
SEROUT LCD,T9600, [7C,0C,14,05,14,0B,01]
SEROUT LCD,T9600, [7C,0C,13,0B,13,11,01]
SEROUT LCD,T9600, [7C,0C,12,10,12,15,01]
SEROUT LCD,T9600, [7C,0C,11,15,11,1E,01]
SEROUT LCD,T9600, [7C,0C,10,1D,10,1F,01]
SEROUT LCD,T9600, [7C,0C,0F,1D,0F,23,01]
SEROUT LCD,T9600, [7C,0C,0E,23,0E,29,01]
SEROUT LCD,T9600, [7C,0C,0D,29,0D,37,01]
SEROUT LCD,T9600, [7C,0C,0C,2D,0C,36,01]
SEROUT LCD,T9600, [$7C,$0C,$0E,$37,$0E,$38,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0F,$38,$0F,$39,$01]  
SEROUT LCD,T9600, [$7C,$0C,$10,$39,$10,$3A,$01]  
SEROUT LCD,T9600, [$7C,$0C,$11,$39,$11,$3A,$01]  
SEROUT LCD,T9600, [$7C,$0C,$12,$38,$12,$39,$01]  
SEROUT LCD,T9600, [$7C,$0C,$13,$2E,$13,$38,$01]  
SEROUT LCD,T9600, [$7C,$0C,$14,$30,$14,$36,$01]  
SEROUT LCD,T9600, [$7C,$0C,$12,$2C,$12,$2E,$01]  
SEROUT LCD,T9600, [$7C,$0C,$11,$2B,$11,$2D,$01]  
SEROUT LCD,T9600, [$7C,$0C,$10,$29,$10,$2B,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0F,$28,$0F,$2B,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0D,$26,$0D,$28,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0C,$25,$0C,$27,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0B,$24,$0B,$26,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0A,$23,$0A,$25,$01]  
SEROUT LCD,T9600, [$7C,$0C,$09,$22,$09,$24,$01]  
SEROUT LCD,T9600, [$7C,$0C,$08,$21,$08,$23,$01]  
SEROUT LCD,T9600, [$7C,$0C,$07,$20,$07,$22,$01]  
SEROUT LCD,T9600, [$7C,$0C,$06,$1E,$06,$20,$01]  
SEROUT LCD,T9600, [$7C,$0C,$05,$16,$05,$18,$01]  
SEROUT LCD,T9600, [$7C,$0C,$04,$18,$04,$1E,$01]  
SEROUT LCD,T9600, [$7C,$0C,$06,$14,$06,$16,$01]  
SEROUT LCD,T9600, [$7C,$0C,$07,$13,$07,$15,$01]  
SEROUT LCD,T9600, [$7C,$0C,$08,$12,$08,$13,$01]  
SEROUT LCD,T9600, [$7C,$0C,$09,$11,$09,$12,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0A,$10,$0A,$11,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0B,$11,$0B,$12,$01]  
SEROUT LCD,T9600, [$7C,$0C,$14,$10,$14,$12,$01]  
SEROUT LCD,T9600, [$7C,$0C,$15,$11,$15,$13,$01]  
SEROUT LCD,T9600, [$7C,$0C,$16,$12,$16,$14,$01]  
SEROUT LCD,T9600, [$7C,$0C,$17,$13,$17,$15,$01]  
SEROUT LCD,T9600, [$7C,$0C,$18,$14,$18,$16,$01]  
SEROUT LCD,T9600, [$7C,$0C,$19,$15,$19,$17,$01]  
SEROUT LCD,T9600, [$7C,$0C,$06,$1C,$06,$1E,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0D,$1B,$0D,$1D,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0B,$1A,$0B,$1C,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0A,$19,$0A,$21,$01]  
SEROUT LCD,T9600, [$7C,$0C,$09,$18,$09,$20,$01]  
SEROUT LCD,T9600, [$7C,$0C,$08,$17,$08,$19,$01]  
SEROUT LCD,T9600, [$7C,$0C,$07,$16,$07,$18,$01]  
SEROUT LCD,T9600, [$7C,$0C,$06,$15,$06,$17,$01]  
SEROUT LCD,T9600, [$7C,$0C,$05,$14,$05,$16,$01]  
SEROUT LCD,T9600, [$7C,$0C,$04,$13,$04,$15,$01]  
SEROUT LCD,T9600, [$7C,$0C,$14,$12,$14,$14,$01]  
SEROUT LCD,T9600, [$7C,$0C,$15,$13,$15,$15,$01]  
SEROUT LCD,T9600, [$7C,$0C,$16,$14,$16,$16,$01]  
SEROUT LCD,T9600, [$7C,$0C,$17,$15,$17,$17,$01]  
SEROUT LCD,T9600, [$7C,$0C,$18,$16,$18,$18,$01]  
SEROUT LCD,T9600, [$7C,$0C,$19,$17,$19,$19,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0E,$1C,$0E,$1E,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0D,$1B,$0D,$1D,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0B,$1A,$0B,$1C,$01]  
SEROUT LCD,T9600, [$7C,$0C,$0A,$19,$0A,$21,$01]  
SEROUT LCD,T9600, [$7C,$0C,$09,$18,$09,$20,$01]  
SEROUT LCD,T9600, [$7C,$0C,$08,$17,$08,$19,$01]  

pause 2000  
'Send 2 seconds to draw
staff lines

'Draw Lines
SEROUT LCD,T9600, [$7C,$0C,$00,$10,$7A,$10,$01]  
'Send G line
SEROUT LCD,T9600, [$7C,$0C,$00,$18,$7A,$18,$01]  
'Send D line
SEROUT LCD,T9600, [$7C,$0C,$00,$20,$7A,$20,$01]  
'Send F line
SEROUT LCD,T9600, [$7C,$0C,$00,$28,$7A,$28,$01]  

pause 2000  
'Send 2 seconds to draw
eighth note

'Draw Eighth Notes
SEROUT LCD,T9600, [$7C,$03,$3F,$18,$03,$01]  
'Send "<control>c  
(0x03)" followed by x and y
SEROUT LCD,T9600, [$7C,$03,$3F,$18,$02,$01]  
'defining the center of the circle,
SEROUT LCD,T9600, [$7C,$03,$3F,$18,$01,$01]  
'followed by a number representing the
SEROUT LCD,T9600, [$7C,$0C,$42,$1A,$42,$2D,$01]  
'radius of the circle, followed by a 0 or 1
SEROUT LCD,T9600, [$7C,$03,$52,$10,$03,$01]  
'determines whether to draw or erase the
SEROUT LCD,T9600, [$7C,$03,$52,$10,$02,$01]  
'circle.
SEROUT LCD,T9600, [$7C,$03,$52,$10,$01,$01]
SEROUT LCD,T9600, [$7C,$0C,$54,$12,$54,$24,$01]
SEROUT LCD,T9600, [$7C,$0C,$53,$25,$43,$2D,$01]
SEROUT LCD,T9600, [$7C,$0C,$53,$24,$43,$2C,$01]
SEROUT LCD,T9600, [$7C,$0C,$53,$1D,$43,$25,$01]

pause 2000

end

---End of Gadget Freak Case #162------