

Build instructions

Name: James Kinney

Project: Air-Suspension System

School: Colorado State University

Class: MECH307 Mechatronics

This gadget is a Mechatronic Microcontroller. It is an air-suspension setup (for a car, but built a test bed for it for demonstration) that will set its height to a specified value of your choice.

Basically, you use a touchscreen to select a ride height on an LCD, then you send the information serially to another microcontroller. The "controls" microcontroller then sets the height to the correct value you choose.

DETAILS:

We took a resistive touch-screen out of a Sudoku game and made it into a 1-9 keypad touchscreen. We used a set of transistors to turn a change in resistance into a change in voltage and output that to a PIC16f88 microcontroller. This in turn sent a binary number to a 7447 7-seg driver to display what key was pressed.

A menu-PIC16f88 sends information to an LCD serially, as well as having 4 analog to digital converters on it. The menu-driven software allows you to see your current height and pressure (A/D conversions, 2 potentiometers and 2 pressure sensors) and select a ride height. The ride height you set is saved on EEPROM and when restarted, the value will still be what you left it at.

Once a ride height is chosen, the menu-PIC sends serial data to the controller-PIC and then goes into a pseudo-closed-loop system to lower/raise the frame. Since the air-tank that holds the pressure is held at 105psi, and the air suspension needs only 10 to 30 psi, we incorporated a stepper motor to control a pressure regulator. So when you need to raise, the stepper sets the regulator to the correct setting and then switches a solenoid open, thus raising the suspension.

Upon turning the device ON, there is a Welcome screen followed by "London Bridges are Falling Down" intro-tune. Afterwards, the main menu comes on screen and lets you choose what you want to do.

That code is for the assembler to setup the PIC16f88's for the correct settings. The OSCON statements set the internal oscillator in the microcontroller to 8 MHz, the DEFINE ADC***** set the analog to digital conversion for 8 bits and a certain sample time, and the ANSEL and TRIS statements set the in/out ports (ansel sets the analog to digital ports on/off, tris just makes input/output).

```
DEFINE OSC 8
OSCCON.4 = 1 'Sets the internal oscillator frequency to 8 MHz
OSCCON.5 = 1
OSCCON.6 = 1
```

```
DEFINE ADC_BITS 8
DEFINE ADC_CLOCK 3
DEFINE ADC_SAMPLEUS 50
```

```
ansel = %00000011
TRISA = %01001111
TRISB = %00000000
```

```
LINCH VAR PORTA.0
RINCH VAR PORTA.1
RPSI VAR PORTA.2
LPSI VAR PORTA.3
```

```
STEP1 VAR PORTB.0 '4 STEPS
STEP2 VAR PORTB.1
STEP3 VAR PORTB.2
STEP4 VAR PORTB.3
```

```
R_UP VAR PORTB.4 '4 SOLENOIDS
R_DN VAR PORTB.5
L_UP VAR PORTB.6
L_DN VAR PORTB.7
```

```
INCHES_L VAR BYTE
INCHES_R VAR BYTE
L_ADJ VAR BYTE : L_ADJ = 0 'Serin value from MENU PIC
R_ADJ VAR BYTE : L_ADJ = 0 'Serin value from MENU PIC
I VAR BYTE
LAST_L VAR BYTE
```

```
LOW R_UP
LOW R_DN
LOW L_UP
LOW L_DN
LOW STEP1
LOW STEP2
LOW STEP3
```

LOW STEP4

PAUSE 500

WHILE 1 'infinite loop begin

WHILE (L_ADJ == 0) OR (R_ADJ == 0) 'wait until values
SERIN PORTA.6, 2, ["A"], L_ADJ 'are serially received
SERIN PORTA.6, 2, ["B"], R_ADJ
WEND

ADCIN 0, INCHES_L 'A/D CONVERSION
ADCIN 1, INCHES_R
INCHES_L = INCHES_L/5 'MATH TO MAKE A/D
INCHES_R = INCHES_R/5 'INTO INCHES

PAUSE 25

IF (L_ADJ < INCHES_L) THEN 'COMPARE MENU VARIABLE TO
PAUSE 100 'A/D VARIABLE
FOR I = 1 TO 20 'ALLOW PRESSURE IN WITH STEPPER.
GOSUB BACKWARD
NEXT I

PAUSE 250

WHILE (L_ADJ < INCHES_L) 'LOOP TO ADJUST PRESSURE SOLENOID
HIGH L_DN
PAUSE 250
LOW L_DN
PAUSE 250
ADCIN 0, INCHES_L
ADCIN 1, INCHES_R
INCHES_L = INCHES_L/5
INCHES_R = INCHES_R/5
WEND

FOR I = 1 TO 20 'PUT STEPPER/REGULATOR BACK
GOSUB FORWARD 'TO ORIGINAL POSITION.
NEXT I

L_ADJ = 0

ENDIF

```
IF (L_ADJ > INCHES_L) THEN
PAUSE 100
FOR I = 1 TO 20
GOSUB FORWARD
NEXT I
```

PAUSE 250

```
WHILE (L_ADJ > INCHES_L)
HIGH L_UP
PAUSE 250
LOW L_UP
PAUSE 250
ADCIN 0, INCHES_L
ADCIN 1, INCHES_R
INCHES_L = INCHES_L/5
INCHES_R = INCHES_R/5
WEND
```

```
FOR I = 1 TO 20
GOSUB BACKWARD
NEXT I
```

L_ADJ = 0
ENDIF

```
IF (R_ADJ < INCHES_R) THEN
PAUSE 100
FOR I = 1 TO 20
GOSUB BACKWARD
NEXT I
```

PAUSE 250

```
WHILE (R_ADJ < INCHES_R)
HIGH R_DN
PAUSE 250
LOW R_DN
```

```
PAUSE 250
ADCIN 0, INCHES_L
ADCIN 1, INCHES_R
INCHES_L = INCHES_L/5
INCHES_R = INCHES_R/5
WEND
```

```
FOR I = 1 TO 20
GOSUB FORWARD
NEXT I
```

```
R_ADJ = 0
ENDIF
```

```
IF (R_ADJ > INCHES_R) THEN
PAUSE 100
FOR I = 1 TO 20
GOSUB FORWARD
NEXT I
```

```
PAUSE 250
```

```
WHILE (R_ADJ > INCHES_R)
HIGH R_UP
PAUSE 250
LOW R_UP
PAUSE 250
ADCIN 0, INCHES_L
ADCIN 1, INCHES_R
INCHES_L = INCHES_L/5
INCHES_R = INCHES_R/5
WEND
```

```
FOR I = 1 TO 20
GOSUB BACKWARD
NEXT I
```

```
R_ADJ = 0
ENDIF
```

WEND

FORWARD: 'ALLOW PRESSURE TO
HIGH STEP1 'INCREASE IN THE
PAUSE 50 'REGULATOR, STEPPER
LOW STEP1 'SEQUENCE.

HIGH STEP2
PAUSE 50
LOW STEP2
HIGH STEP3
PAUSE 50
LOW STEP3
HIGH STEP4
PAUSE 50
LOW STEP4
RETURN

BACKWARD: 'SAME AS FORWARD,
HIGH STEP4 'BUT OBVIOUSLY
PAUSE 50 'IN REVERSE

LOW STEP4
HIGH STEP3
PAUSE 50
LOW STEP3
HIGH STEP2
PAUSE 50
LOW STEP2
HIGH STEP1
PAUSE 50
LOW STEP1
RETURN

```
DEFINE OSC 8
OSCCON.4 = 1 'Sets the internal oscillator frequency to 8 MHz
OSCCON.5 = 1
OSCCON.6 = 1
```

```
ansel = 0 'Turns off analog to digital conversion
```

```
TRISB = %11110000 'B.0,1,2,3 ARE OUTPUTS, B.4,5,6,7 ARE INPUTS
TRISA = %11111111 'A.1,0,7,6 ARE INPUTS
```

```
'BELOW 4 LINES REFER TO OUTPUT TO 7447
'B.0 IS "A," LEAST SIGNIFICANT DIGIT
'B.3 IS "B," SENCOND LEAST SIGNIFICANT
'B.2 IS "C," SECOND HIGHEST SIGNIFICANT
'B.1 IS "D," HIGHEST SIGNIFICANT DIGIT
```

```
PAUSE 500 'INTRO PAUSE TO LET EVERYTHING WARM-UP
LOW PORTB.0
LOW PORTB.1
LOW PORTB.2
LOW PORTB.3
```

```
WHILE 1 'INFINITE LOOP
```

```
LOW PORTB.0
LOW PORTB.1
LOW PORTB.2
LOW PORTB.3
```

```
'OUTPUTS ARE TO 7447, AND ARE IN BINARY
'RELATING TO WHAT KEY WAS PRESSED.
```

```
IF (PORTA.1 == 0) THEN 'BUTTON #1 PRESSED
HIGH PORTB.0
LOW PORTB.1
LOW PORTB.2
LOW PORTB.3
PAUSE 500
LOW PORTB.0
WHILE (PORTA.1 == 0)
```

```
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTA.0 == 0) THEN 'BUTTON #2 PRESSED
HIGH PORTB.3
LOW PORTB.0
LOW PORTB.1
LOW PORTB.2
PAUSE 500
LOW PORTB.3
WHILE (PORTA.0 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTA.7 == 0) THEN 'BUTTON #3 PRESSED
HIGH PORTB.0
HIGH PORTB.3
LOW PORTB.1
LOW PORTB.2
PAUSE 500
LOW PORTB.0
LOW PORTB.3
WHILE (PORTA.7 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTA.6 == 0) THEN 'BUTTON #4 PRESSED
HIGH PORTB.2
LOW PORTB.0
LOW PORTB.1
LOW PORTB.3
PAUSE 500
LOW PORTB.2
WHILE (PORTA.6 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTB.7 == 0) THEN 'BUTTON #5 PRESSED
HIGH PORTB.0
HIGH PORTB.2
LOW PORTB.1
LOW PORTB.3
```

```
PAUSE 500
LOW PORTB.0
LOW PORTB.2
WHILE (PORTB.7 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTB.6 == 0) THEN 'BUTTON #6 PRESSED
HIGH PORTB.2
HIGH PORTB.3
LOW PORTB.0
LOW PORTB.1
PAUSE 500
LOW PORTB.2
LOW PORTB.3
WHILE (PORTB.6 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTB.5 == 0) THEN 'BUTTON #7 PRESSED
HIGH PORTB.0
HIGH PORTB.2
HIGH PORTB.3
LOW PORTB.1
PAUSE 500
LOW PORTB.0
LOW PORTB.2
LOW PORTB.3
WHILE (PORTB.5 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTB.4 == 0) THEN 'BUTTON #8 PRESSED
HIGH PORTB.1
LOW PORTB.0
LOW PORTB.2
LOW PORTB.3
PAUSE 500
LOW PORTB.1
WHILE (PORTB.4 == 0)
WEND
PAUSEUS 50
ENDIF
```

```

IF (PORTA.4 == 0) THEN 'BUTTON #9 PRESSED
HIGH PORTB.1
HIGH PORTB.0
LOW PORTB.2
LOW PORTB.3
PAUSE 500
LOW PORTB.1
LOW PORTB.0
WHILE (PORTA.4 == 0)
WEND
PAUSEUS 50
ENDIF

WEND
END
DEFINE OSC 8
OSCCON.4 = 1 'Sets the internal oscillator frequency to 8 MHz
OSCCON.5 = 1
OSCCON.6 = 1

DEFINE ADC_BITS 8
DEFINE ADC_CLOCK 3
DEFINE ADC_SAMPLEUS 50

ansel = %00001111
TRISB = %11111111
TRISA = %00001111

INCHES_L VAR BYTE
INCHES_R VAR BYTE
PRS1RAW VAR BYTE
PRS1 VAR BYTE
PRS1DEC VAR BYTE
PRS2RAW VAR BYTE
PRS2 VAR BYTE
PRS2DEC VAR BYTE
KEYIN VAR BYTE
MENU_STATE VAR BYTE : MENU_STATE = 0 'CHOOSE NEW MENU
L_ADJ VAR BYTE : READ 0, L_ADJ 'CHECK EEPROM
R_ADJ VAR BYTE : READ 1, R_ADJ

PAUSE 500 'INTRO SCREEN
SEROUT PORTA.7, 2, [$FE, 1]
SEROUT PORTA.7, 2, [$FE, 1, " WELCOME TO NJZZ"]

```

```
SEROUT PORTA.7, 2, [$FE, $C0, " AIR "]
SEROUT PORTA.7, 2, [$FE, $94, " SUSPENSION"]
SEROUT PORTA.7, 2, [$FE, $D4, " SYSTEM"]
PAUSE 1000
```

```
SOUND PORTB.4, [106, 30, 109, 10, 106, 20, 102, 20, 104, 20, 106, 40, 99, 20, 102, 20, 104,
40, 102, 20, 104, 20, 106, 40, 106, 30, 109, 10, 106, 20, 104, 20, 102, 20, 104, 20, 106, 40, 99,
40, 106, 40, 102, 20, 96, 60]
```

```
'LONDON BRIDGE IS FALLING DOWN ^^^^^^^^
```

```
WHILE 1 'INFINITE LOOP
```

```
IF (MENU_STATE == 0) THEN 'MAIN MENU
GOSUB MENU0
ENDIF
```

```
IF (MENU_STATE == 1) THEN 'DISPLAY MENU
GOSUB MENU1
ENDIF
```

```
IF (MENU_STATE == 2) THEN 'CHANGE HEIGHT MENU
GOSUB MENU2:
ENDIF
```

```
MENU_STATE = 0
PAUSE 150
```

```
WEND
END
```

```
'GOSUBS:.....
```

```
MENU0:
SEROUT PORTA.7, 2, [$FE, 1] '12345678901234567890
SEROUT PORTA.7, 2, [$FE, 1, " *****MAIN MENU*****"]
SEROUT PORTA.7, 2, [$FE, $C0, "1: DISPLAY INCH/PSI"]
SEROUT PORTA.7, 2, [$FE, $94, "2: CHANGE SETTINGS"]
SEROUT PORTA.7, 2, [$FE, $D4, " "]
WHILE (MENU_STATE == 0)
KEYIN = PORTB & $0F
MENU_STATE = KEYIN
```

```
PAUSE 100
WEND
RETURN
```

```
MENU1: 'LOGIC TO KEEP 1 OR 2 DIGIT NUMBERS CENTER
PAUSE 600 'CORRECTLY. ALSO HAS DECIMAL PLACE.
```

```
WHILE (MENU_STATE == 1)
```

```
ADCIN 0, INCHES_L
```

```
ADCIN 1, INCHES_R
```

```
ADCIN 3, PRS1RAW
```

```
ADCIN 2, PRS2RAW
```

```
GOSUB MATH
```

```
'12345678901234567890" <
```

```
'SEROUT PORTA.7, 2, [$FE, 1, " LEFT RIGHT"] '<
```

```
'INCHES: 10 10 <
```

```
'PSI : 10.2 10.2 <
```

```
IF (INCHES_L >= 10) & (INCHES_R >= 10) THEN
```

```
SEROUT PORTA.7, 2, [$FE, 1, " LEFT RIGHT"]
```

```
SEROUT PORTA.7, 2, [$FE, $C0, "INCHES: ", #INCHES_L, " ", #INCHES_R ]
```

```
ENDIF
```

```
IF (INCHES_L < 10) & (INCHES_R >= 10) THEN
```

```
SEROUT PORTA.7, 2, [$FE, 1, " LEFT RIGHT"]
```

```
SEROUT PORTA.7, 2, [$FE, $C0, "INCHES: ", #INCHES_L, " ", #INCHES_R]
```

```
ENDIF
```

```
IF (INCHES_L >= 10) & (INCHES_R < 10) THEN
```

```
SEROUT PORTA.7, 2, [$FE, 1, " LEFT RIGHT"]
```

```
SEROUT PORTA.7, 2, [$FE, $C0, "INCHES: ", #INCHES_L, " ", #INCHES_R]
```

```
ENDIF
```

```
IF (INCHES_L < 10) & (INCHES_R < 10) THEN
```

```
SEROUT PORTA.7, 2, [$FE, 1, " LEFT RIGHT"]
```

```
SEROUT PORTA.7, 2, [$FE, $C0, "INCHES: ", #INCHES_L, " ", #INCHES_R]
```

```
ENDIF
```

```
SEROUT PORTA.7, 2, [$FE, $94, "PSI : ", #PRS1, ".", #PRS1DEC, " ", #PRS2, ".",  
#PRS2DEC]
```

```
SEROUT PORTA.7, 2, [$FE, $D4, "ANY KEY TO RETURN"]
```

```
KEYIN = PORTB & $0F
```

```
IF (KEYIN > 0) THEN
```

```
MENU_STATE = 0 'GOTO MAIN MENU
```

ENDIF

PAUSE 150

WEND

RETURN

MENU2:

PAUSE 600

WHILE (MENU_STATE == 2)

ADCIN 0, INCHES_L

ADCIN 1, INCHES_R

GOSUB MATH

'12345678901234567890" <

'SEROUT PORTA.7, 2, [\$FE, 1, "1MAIN- LEFT RIGHT"] '<

'INCHES : 10 10 <

'NEW SET: 10 10 <

'2:SET L:6U7D R:8U9D

SEROUT PORTA.7, 2, [\$FE, 1, " 1MAIN- LEFT RIGHT"]

IF (INCHES_L >= 10) & (INCHES_R >= 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$C0, "INCHES : ", #INCHES_L, " ", #INCHES_R]

ENDIF

IF (INCHES_L < 10) & (INCHES_R >= 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$C0, "INCHES : ", #INCHES_L, " ", #INCHES_R]

ENDIF

IF (INCHES_L >= 10) & (INCHES_R < 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$C0, "INCHES : ", #INCHES_L, " ", #INCHES_R]

ENDIF

IF (INCHES_L < 10) & (INCHES_R < 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$C0, "INCHES : ", #INCHES_L, " ", #INCHES_R]

ENDIF

IF (L_ADJ >= 10) & (R_ADJ >= 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$94, "NEW SET: ", #L_ADJ, " ", #R_ADJ]

ENDIF

IF (L_ADJ < 10) & (R_ADJ >= 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$94, "NEW SET: ", #L_ADJ, " ", #R_ADJ]

ENDIF

IF (L_ADJ >= 10) & (R_ADJ < 10) THEN

```
SEROUT PORTA.7, 2, [$FE, $94, "NEW SET: ", #L_ADJ, " ", #R_ADJ]
ENDIF
```

```
IF (L_ADJ < 10) & (R_ADJ < 10) THEN
SEROUT PORTA.7, 2, [$FE, $94, "NEW SET: ", #L_ADJ, " ", #R_ADJ]
ENDIF
```

```
SEROUT PORTA.7, 2, [$FE, $D4, "2:SET L:6U7D R:8U9D"]
```

```
GOSUB KEY_MATH
```

```
PAUSE 150
WEND
```

```
MATH: 'MATH CONVERTS ABSTRACT A/D SIGNALS TO INCHES AND PSI, DECIMAL
PSI AS WELL
```

```
INCHES_L = INCHES_L/5
INCHES_R = INCHES_R/5
```

```
PRS1 = PRS1RAW/7
PRS1DEC = PRS1//7
PRS1DEC = PRS1DEC*10 / 7
```

```
PRS2 = PRS2RAW/7
PRS2DEC = PRS2//7
PRS2DEC = PRS2DEC*10 / 7
RETURN
```

```
KEY_MATH:
```

```
KEYIN = PORTB & $0F
IF (KEYIN == 6) THEN
PAUSE 400
L_ADJ = L_ADJ +1
ENDIF
```

```
IF (KEYIN == 7) THEN
PAUSE 400
L_ADJ = L_ADJ -1
ENDIF
```

```
IF (KEYIN == 8) THEN
PAUSE 400
R_ADJ = R_ADJ +1
```

ENDIF

```
IF (KEYIN == 9) THEN
  PAUSE 400
  R_ADJ = R_ADJ -1
ENDIF
```

```
IF (L_ADJ > 38) THEN 'VALUE CANNOT EXCEED 37
L_ADJ = 37
ENDIF
```

```
IF (L_ADJ < 26) THEN 'VALUE CANNOT GO BELOW 26
L_ADJ = 26
ENDIF
```

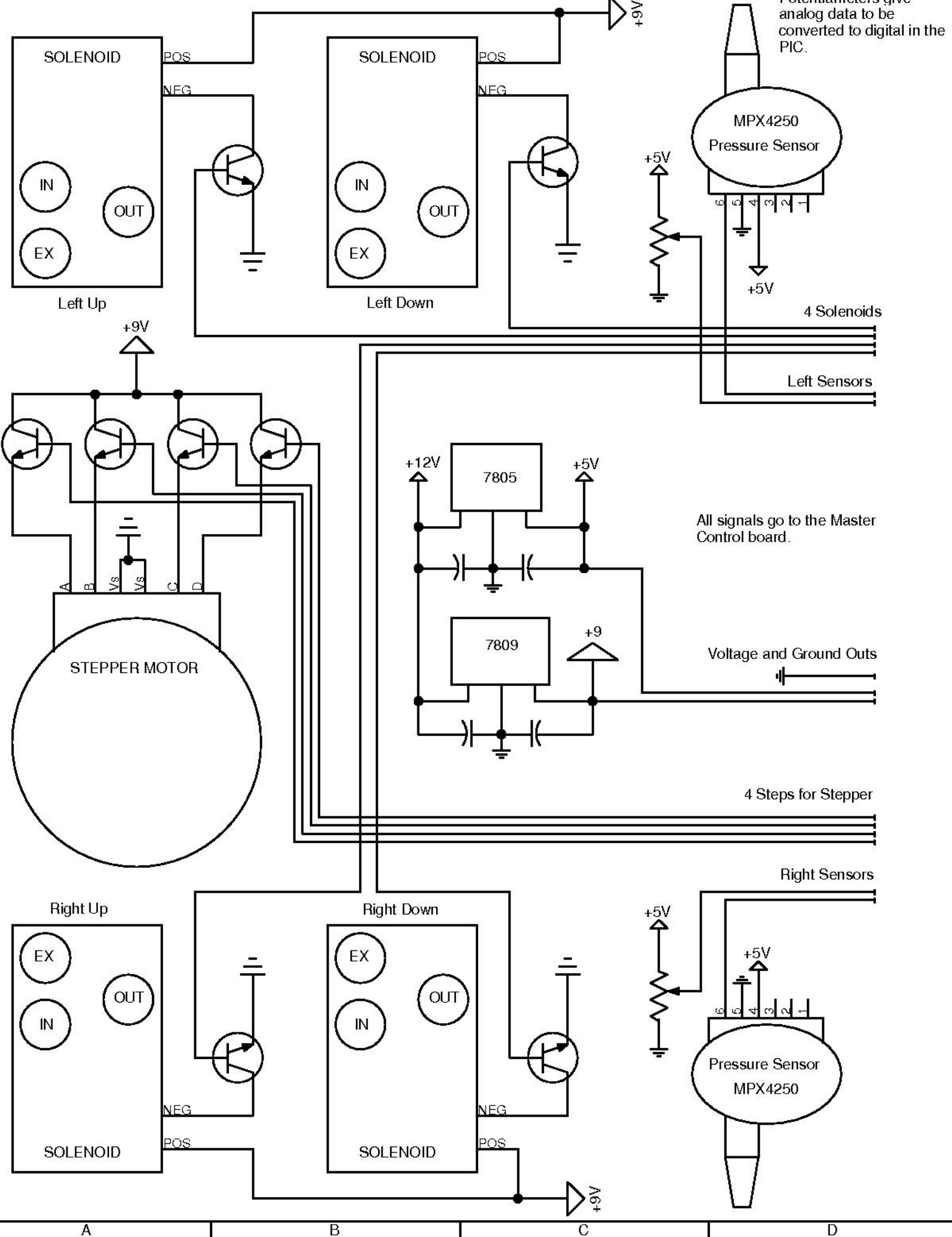
```
IF (R_ADJ > 38) THEN
  R_ADJ = 37
ENDIF
```

```
IF (R_ADJ < 26) THEN
  R_ADJ = 26
ENDIF
```

```
IF (KEYIN == 1) THEN '1 TO RETURN TO MAIN MENU
  PAUSE 400
  MENU_STATE = 0
ENDIF
```

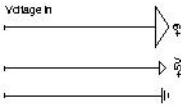
```
IF (KEYIN == 2) THEN 'SERIAL TO CONTROLS PIC, IF 2 PRESSED
  PAUSE 400
  SEROUT PORTA.6, 2, ["A", L_ADJ]
  SEROUT PORTA.6, 2, ["B", R_ADJ]
ENDIF
WRITE 0, L_ADJ 'SAVE VALUES TO EEPROM
WRITE 1, R_ADJ
RETURN
```

CONTROLLER BOARD



MASTER MENU/CONTROLS BOARD

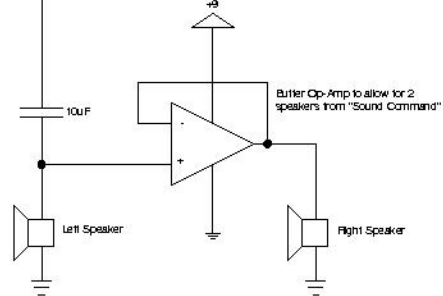
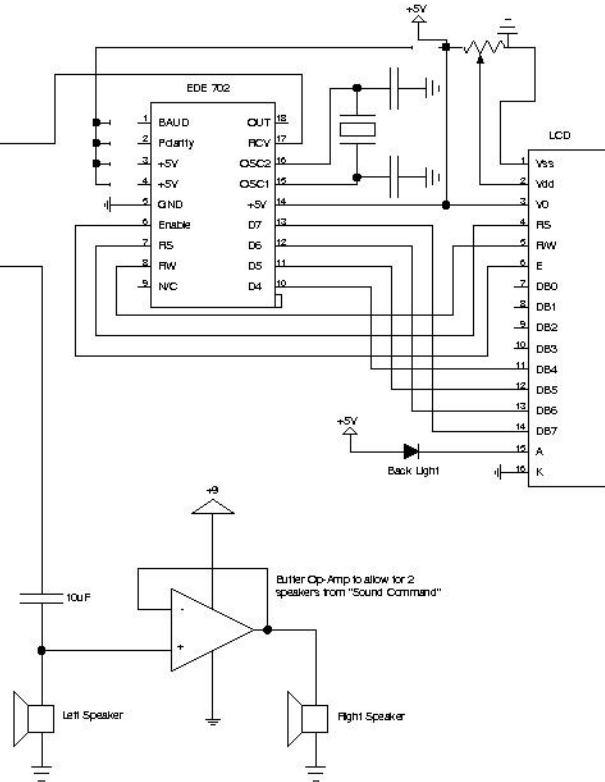
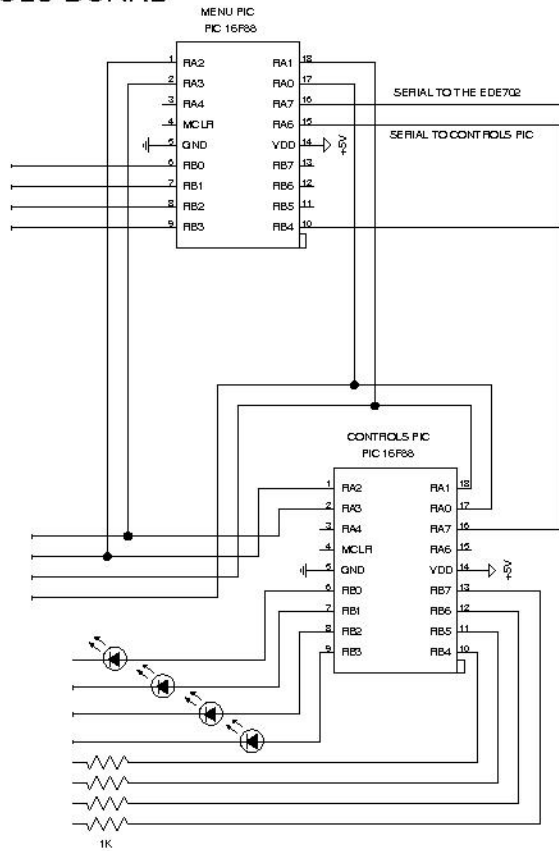
Parallel binary number from TOUCHSCREEN circuit.



4 ANALOG SIGNALS ARE SENT FROM THE 2 POTENTIOMETERS AND 2 PRESSURE SENSORS. THE SIGNALS ARE SENT TO TWO PICs, AND FOR SIMPLICITY, A SENSOR WILL OUTPUT TO THE SAME PORT ON EACH. FOR INSTANCE, PORTA.0 ON ONE PIC WILL READ THE SAME AS PORTA.0 ON THE OTHER PIC.

Four steps for stepper motor. Oddly enough, the LEDs are in the correct location. The transistors would over-heat with resistors in series to the stepper, and often fail. The best guess as to why these work is they set a low voltage (2.5V drop across LED) and this is enough to saturate the transistor in an efficient manner.

These 4 cuts are for the solenoids. Internal resistance in the solenoids was a lot larger than the stepper (36 Ohms vs. ~3 Ohms in the stepper), so smaller transistors were required for these.



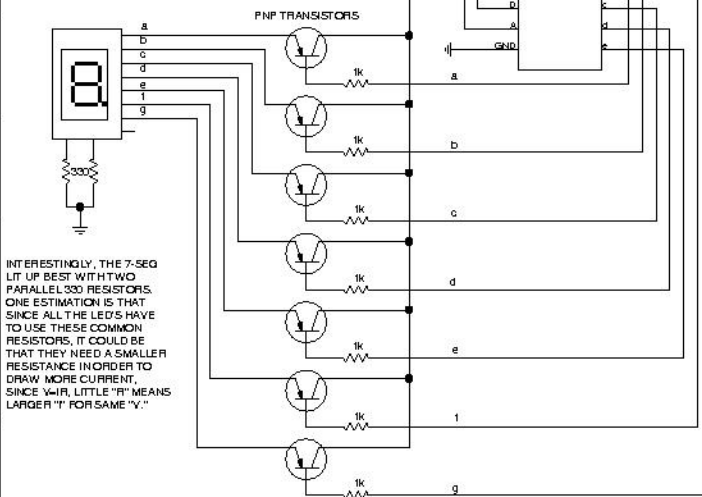
TOUCHSCREEN SCHEMATIC

BASICALLY, WHEN A ROW IS PUSHED ON THE TOUCHSCREEN (ROWS 1 THROUGH 9), THEY CAUSE A NPN TO SEND A LOW SIGNAL TO A PIC16F88. THE PROGRAM THEN OUTPUTS THE CORRECT BINARY SIGNAL TO A 7447 (FOR EXAMPLE, IF 3 WERE PRESSED, 0011 WOULD BE SENT TO THE 7447), THIS CAUSES THE DIGIT TO APPEAR ON THE 7-SEG FOR A SET AMOUNT OF TIME, FOR INSTANCE 1/2 SECOND.

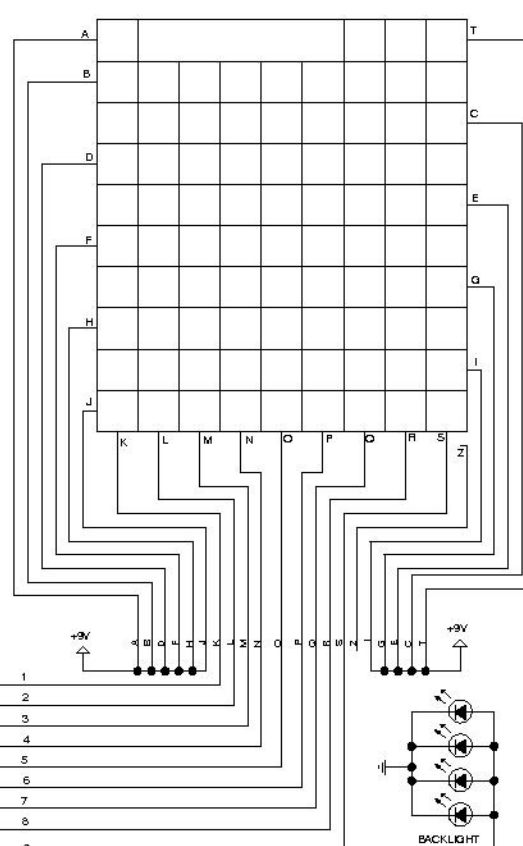
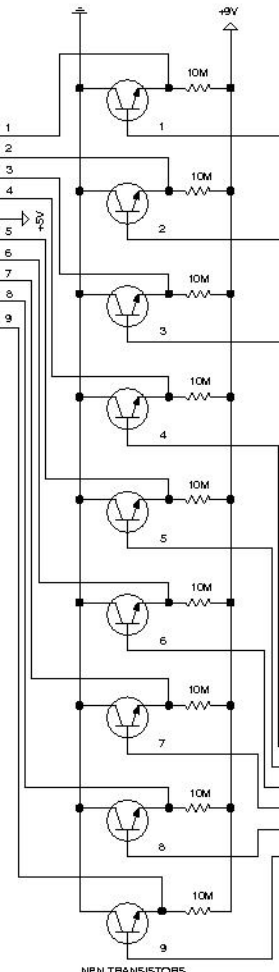
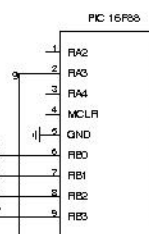
THE PIC SENDS A BINARY NUMBER TO A 7447 TO BE DISPLAYED ON A 7-SEG DISPLAY. THIS SAME PARALLEL INFORMATION IS ALSO SENT TO THE "HOST" MENU PIC ON THE MENU/CONTROLLER CIRCUIT

A COMMON GROUND IS NEEDED FROM THE CONTROL BOARD, AS WELL AS THE +5 AND +9 VOLTAGES. THE +9 IS ONLY USED ON THE SCREEN ITSELF.

THE 7-SEG DISPLAY NEEDS A POSITIVE SIGNAL, AND THE 7447 GOUNDS AS ITS SIGNAL. THE SOLUTION IS TO USE A SET OF PNP TRANSISTORS. IN THIS CONFIGURATION, THE 7447 GOUNDS OUT THE "BASE" AND THUS SATURATES THE TRANSISTOR. THIS IN TURN CAUSES CURRENT TO PASS AND SEND VOLTAGE TO THE 7-SEG DISPLAY.



INTERESTINGLY, THE 7-SEG LIT UP BEST WITH TWO PARALLEL 330 RESISTORS. ONE ESTIMATION IS THAT SINCE ALL THE LED'S HAVE TO USE THESE COMMON RESISTORS, IT COULD BE THAT THEY NEED A SMALLER RESISTANCE IN ORDER TO DRAW MORE CURRENT, SINCE "4"R, LITTLE "F" MEANS LARGER "I" FOR SAME "V."



THE SCREEN OPERATES ON RESISTANCE. WITHOUT TOUCHING IT, THE RESISTANCE IS SIGNIFICANT. ONCE A "BUTTON" IS PRESSED, THE RESISTANCE BETWEEN THE LEAD-OUTS OF THE ROW AND COLUMN BECOME ~10M OHMS. A 10M OHM RESISTOR WAS THUS PAIRED ON AN NPN TRANSISTOR TO PUT IT INTO SATURATION. THE PIC16F88 WILL SEE A "HIGH" SIGNAL WITHOUT A BUTTON PRESSED, AND "LOW" ONCE THE BUTTON IS PRESSED. IN THIS CONFIGURATION, THE ROWS ARE THE "BUTTONS" AND THE COLUMNS ARE SET TO +9V. VOLTAGE DROP ON THE RESISTOR AND NPN RESULT IN ~+5V AND -0V SIGNALS, RESPECTIVELY.

Build instructions

Name: James Kinney

Project: Air-Suspension System

School: Colorado State University

Class: MECH307 Mechatronics

This gadget is a Mechatronic Microcontroller. It is an air-suspension setup (for a car, but built a test bed for it for demonstration) that will set its height to a specified value of your choice.

Basically, you use a touchscreen to select a ride height on an LCD, then you send the information serially to another microcontroller. The "controls" microcontroller then sets the height to the correct value you choose.

DETAILS:

We took a resistive touch-screen out of a Sudoku game and made it into a 1-9 keypad touchscreen. We used a set of transistors to turn a change in resistance into a change in voltage and output that to a PIC16f88 microcontroller. This in turn sent a binary number to a 7447 7-seg driver to display what key was pressed.

A menu-PIC16f88 sends information to an LCD serially, as well as having 4 analog to digital converters on it. The menu-driven software allows you to see your current height and pressure (A/D conversions, 2 potentiometers and 2 pressure sensors) and select a ride height. The ride height you set is saved on EEPROM and when restarted, the value will still be what you left it at.

Once a ride height is chosen, the menu-PIC sends serial data to the controller-PIC and then goes into a pseudo-closed-loop system to lower/raise the frame. Since the air-tank that holds the pressure is held at 105psi, and the air suspension needs only 10 to 30 psi, we incorporated a stepper motor to control a pressure regulator. So when you need to raise, the stepper sets the regulator to the correct setting and then switches a solenoid open, thus raising the suspension.

Upon turning the device ON, there is a Welcome screen followed by "London Bridges are Falling Down" intro-tune. Afterwards, the main menu comes on screen and lets you choose what you want to do.

That code is for the assembler to setup the PIC16f88's for the correct settings. The OSCON statements set the internal oscillator in the microcontroller to 8 MHz, the DEFINE ADC***** set the analog to digital conversion for 8 bits and a certain sample time, and the ANSEL and TRIS statements set the in/out ports (ansel sets the analog to digital ports on/off, tris just makes input/output).

```
DEFINE OSC 8
OSCCON.4 = 1 'Sets the internal oscillator frequency to 8 MHz
OSCCON.5 = 1
OSCCON.6 = 1
```

```
DEFINE ADC_BITS 8
DEFINE ADC_CLOCK 3
DEFINE ADC_SAMPLEUS 50
```

```
ansel = %00000011
TRISA = %01001111
TRISB = %00000000
```

```
LINCH VAR PORTA.0
RINCH VAR PORTA.1
RPSI VAR PORTA.2
LPSI VAR PORTA.3
```

```
STEP1 VAR PORTB.0 '4 STEPS
STEP2 VAR PORTB.1
STEP3 VAR PORTB.2
STEP4 VAR PORTB.3
```

```
R_UP VAR PORTB.4 '4 SOLENOIDS
R_DN VAR PORTB.5
L_UP VAR PORTB.6
L_DN VAR PORTB.7
```

```
INCHES_L VAR BYTE
INCHES_R VAR BYTE
L_ADJ VAR BYTE : L_ADJ = 0 'Serin value from MENU PIC
R_ADJ VAR BYTE : L_ADJ = 0 'Serin value from MENU PIC
I VAR BYTE
LAST_L VAR BYTE
```

```
LOW R_UP
LOW R_DN
LOW L_UP
LOW L_DN
LOW STEP1
LOW STEP2
LOW STEP3
```

LOW STEP4

PAUSE 500

WHILE 1 'infinite loop begin

WHILE (L_ADJ == 0) OR (R_ADJ == 0) 'wait until values
SERIN PORTA.6, 2, ["A"], L_ADJ 'are serially received
SERIN PORTA.6, 2, ["B"], R_ADJ
WEND

ADCIN 0, INCHES_L 'A/D CONVERSION
ADCIN 1, INCHES_R
INCHES_L = INCHES_L/5 'MATH TO MAKE A/D
INCHES_R = INCHES_R/5 'INTO INCHES

PAUSE 25

IF (L_ADJ < INCHES_L) THEN 'COMPARE MENU VARIABLE TO
PAUSE 100 'A/D VARIABLE
FOR I = 1 TO 20 'ALLOW PRESSURE IN WITH STEPPER.
GOSUB BACKWARD
NEXT I

PAUSE 250

WHILE (L_ADJ < INCHES_L) 'LOOP TO ADJUST PRESSURE SOLENOID
HIGH L_DN
PAUSE 250
LOW L_DN
PAUSE 250
ADCIN 0, INCHES_L
ADCIN 1, INCHES_R
INCHES_L = INCHES_L/5
INCHES_R = INCHES_R/5
WEND

FOR I = 1 TO 20 'PUT STEPPER/REGULATOR BACK
GOSUB FORWARD 'TO ORIGINAL POSITION.
NEXT I

L_ADJ = 0

ENDIF

```
IF (L_ADJ > INCHES_L) THEN
PAUSE 100
FOR I = 1 TO 20
GOSUB FORWARD
NEXT I
```

PAUSE 250

```
WHILE (L_ADJ > INCHES_L)
HIGH L_UP
PAUSE 250
LOW L_UP
PAUSE 250
ADCIN 0, INCHES_L
ADCIN 1, INCHES_R
INCHES_L = INCHES_L/5
INCHES_R = INCHES_R/5
WEND
```

```
FOR I = 1 TO 20
GOSUB BACKWARD
NEXT I
```

L_ADJ = 0
ENDIF

```
IF (R_ADJ < INCHES_R) THEN
PAUSE 100
FOR I = 1 TO 20
GOSUB BACKWARD
NEXT I
```

PAUSE 250

```
WHILE (R_ADJ < INCHES_R)
HIGH R_DN
PAUSE 250
LOW R_DN
```

```
PAUSE 250
ADCIN 0, INCHES_L
ADCIN 1, INCHES_R
INCHES_L = INCHES_L/5
INCHES_R = INCHES_R/5
WEND
```

```
FOR I = 1 TO 20
GOSUB FORWARD
NEXT I
```

```
R_ADJ = 0
ENDIF
```

```
IF (R_ADJ > INCHES_R) THEN
PAUSE 100
FOR I = 1 TO 20
GOSUB FORWARD
NEXT I
```

```
PAUSE 250
```

```
WHILE (R_ADJ > INCHES_R)
HIGH R_UP
PAUSE 250
LOW R_UP
PAUSE 250
ADCIN 0, INCHES_L
ADCIN 1, INCHES_R
INCHES_L = INCHES_L/5
INCHES_R = INCHES_R/5
WEND
```

```
FOR I = 1 TO 20
GOSUB BACKWARD
NEXT I
```

```
R_ADJ = 0
ENDIF
```

WEND

FORWARD: 'ALLOW PRESSURE TO
HIGH STEP1 'INCREASE IN THE
PAUSE 50 'REGULATOR, STEPPER
LOW STEP1 'SEQUENCE.

HIGH STEP2

PAUSE 50

LOW STEP2

HIGH STEP3

PAUSE 50

LOW STEP3

HIGH STEP4

PAUSE 50

LOW STEP4

RETURN

BACKWARD: 'SAME AS FORWARD,
HIGH STEP4 'BUT OBVIOUSLY
PAUSE 50 'IN REVERSE

LOW STEP4

HIGH STEP3

PAUSE 50

LOW STEP3

HIGH STEP2

PAUSE 50

LOW STEP2

HIGH STEP1

PAUSE 50

LOW STEP1

RETURN

```
DEFINE OSC 8
OSCCON.4 = 1 'Sets the internal oscillator frequency to 8 MHz
OSCCON.5 = 1
OSCCON.6 = 1
```

```
ansel = 0 'Turns off analog to digital conversion
```

```
TRISB = %11110000 'B.0,1,2,3 ARE OUTPUTS, B.4,5,6,7 ARE INPUTS
TRISA = %11111111 'A.1,0,7,6 ARE INPUTS
```

```
'BELOW 4 LINES REFER TO OUTPUT TO 7447
'B.0 IS "A," LEAST SIGNIFICANT DIGIT
'B.3 IS "B," SENCOND LEAST SIGNIFICANT
'B.2 IS "C," SECOND HIGHEST SIGNIFICANT
'B.1 IS "D," HIGHEST SIGNIFICANT DIGIT
```

```
PAUSE 500 'INTRO PAUSE TO LET EVERYTHING WARM-UP
LOW PORTB.0
LOW PORTB.1
LOW PORTB.2
LOW PORTB.3
```

```
WHILE 1 'INFINITE LOOP
```

```
LOW PORTB.0
LOW PORTB.1
LOW PORTB.2
LOW PORTB.3
```

```
'OUTPUTS ARE TO 7447, AND ARE IN BINARY
'RELATING TO WHAT KEY WAS PRESSED.
```

```
IF (PORTA.1 == 0) THEN 'BUTTON #1 PRESSED
HIGH PORTB.0
LOW PORTB.1
LOW PORTB.2
LOW PORTB.3
PAUSE 500
LOW PORTB.0
WHILE (PORTA.1 == 0)
```

```
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTA.0 == 0) THEN 'BUTTON #2 PRESSED
HIGH PORTB.3
LOW PORTB.0
LOW PORTB.1
LOW PORTB.2
PAUSE 500
LOW PORTB.3
WHILE (PORTA.0 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTA.7 == 0) THEN 'BUTTON #3 PRESSED
HIGH PORTB.0
HIGH PORTB.3
LOW PORTB.1
LOW PORTB.2
PAUSE 500
LOW PORTB.0
LOW PORTB.3
WHILE (PORTA.7 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTA.6 == 0) THEN 'BUTTON #4 PRESSED
HIGH PORTB.2
LOW PORTB.0
LOW PORTB.1
LOW PORTB.3
PAUSE 500
LOW PORTB.2
WHILE (PORTA.6 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTB.7 == 0) THEN 'BUTTON #5 PRESSED
HIGH PORTB.0
HIGH PORTB.2
LOW PORTB.1
LOW PORTB.3
```

```
PAUSE 500
LOW PORTB.0
LOW PORTB.2
WHILE (PORTB.7 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTB.6 == 0) THEN 'BUTTON #6 PRESSED
HIGH PORTB.2
HIGH PORTB.3
LOW PORTB.0
LOW PORTB.1
PAUSE 500
LOW PORTB.2
LOW PORTB.3
WHILE (PORTB.6 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTB.5 == 0) THEN 'BUTTON #7 PRESSED
HIGH PORTB.0
HIGH PORTB.2
HIGH PORTB.3
LOW PORTB.1
PAUSE 500
LOW PORTB.0
LOW PORTB.2
LOW PORTB.3
WHILE (PORTB.5 == 0)
WEND
PAUSEUS 50
ENDIF
```

```
IF (PORTB.4 == 0) THEN 'BUTTON #8 PRESSED
HIGH PORTB.1
LOW PORTB.0
LOW PORTB.2
LOW PORTB.3
PAUSE 500
LOW PORTB.1
WHILE (PORTB.4 == 0)
WEND
PAUSEUS 50
ENDIF
```

```

IF (PORTA.4 == 0) THEN 'BUTTON #9 PRESSED
HIGH PORTB.1
HIGH PORTB.0
LOW PORTB.2
LOW PORTB.3
PAUSE 500
LOW PORTB.1
LOW PORTB.0
WHILE (PORTA.4 == 0)
WEND
PAUSEUS 50
ENDIF

WEND
END
DEFINE OSC 8
OSCCON.4 = 1 'Sets the internal oscillator frequency to 8 MHz
OSCCON.5 = 1
OSCCON.6 = 1

DEFINE ADC_BITS 8
DEFINE ADC_CLOCK 3
DEFINE ADC_SAMPLEUS 50

ansel = %00001111
TRISB = %11111111
TRISA = %00001111

INCHES_L VAR BYTE
INCHES_R VAR BYTE
PRS1RAW VAR BYTE
PRS1 VAR BYTE
PRS1DEC VAR BYTE
PRS2RAW VAR BYTE
PRS2 VAR BYTE
PRS2DEC VAR BYTE
KEYIN VAR BYTE
MENU_STATE VAR BYTE : MENU_STATE = 0 'CHOOSE NEW MENU
L_ADJ VAR BYTE : READ 0, L_ADJ 'CHECK EEPROM
R_ADJ VAR BYTE : READ 1, R_ADJ

PAUSE 500 'INTRO SCREEN
SEROUT PORTA.7, 2, [$FE, 1]
SEROUT PORTA.7, 2, [$FE, 1, " WELCOME TO NJZZ"]

```

```
SEROUT PORTA.7, 2, [$FE, $C0, " AIR "]
SEROUT PORTA.7, 2, [$FE, $94, " SUSPENSION"]
SEROUT PORTA.7, 2, [$FE, $D4, " SYSTEM"]
PAUSE 1000
```

```
SOUND PORTB.4, [106, 30, 109, 10, 106, 20, 102, 20, 104, 20, 106, 40, 99, 20, 102, 20, 104,
40, 102, 20, 104, 20, 106, 40, 106, 30, 109, 10, 106, 20, 104, 20, 102, 20, 104, 20, 106, 40, 99,
40, 106, 40, 102, 20, 96, 60]
```

```
'LONDON BRIDGE IS FALLING DOWN ^^^^^^^^
```

```
WHILE 1 'INFINITE LOOP
```

```
IF (MENU_STATE == 0) THEN 'MAIN MENU
GOSUB MENU0
ENDIF
```

```
IF (MENU_STATE == 1) THEN 'DISPLAY MENU
GOSUB MENU1
ENDIF
```

```
IF (MENU_STATE == 2) THEN 'CHANGE HEIGHT MENU
GOSUB MENU2:
ENDIF
```

```
MENU_STATE = 0
PAUSE 150
```

```
WEND
END
```

```
'GOSUBS:.....
```

```
MENU0:
SEROUT PORTA.7, 2, [$FE, 1] '12345678901234567890
SEROUT PORTA.7, 2, [$FE, 1, " *****MAIN MENU*****"]
SEROUT PORTA.7, 2, [$FE, $C0, "1: DISPLAY INCH/PSI"]
SEROUT PORTA.7, 2, [$FE, $94, "2: CHANGE SETTINGS"]
SEROUT PORTA.7, 2, [$FE, $D4, " "]
WHILE (MENU_STATE == 0)
KEYIN = PORTB & $0F
MENU_STATE = KEYIN
```

```
PAUSE 100
WEND
RETURN
```

```
MENU1: 'LOGIC TO KEEP 1 OR 2 DIGIT NUMBERS CENTER
PAUSE 600 'CORRECTLY. ALSO HAS DECIMAL PLACE.
```

```
WHILE (MENU_STATE == 1)
```

```
ADCIN 0, INCHES_L
```

```
ADCIN 1, INCHES_R
```

```
ADCIN 3, PRS1RAW
```

```
ADCIN 2, PRS2RAW
```

```
GOSUB MATH
```

```
'12345678901234567890" <
```

```
'SEROUT PORTA.7, 2, [$FE, 1, " LEFT RIGHT"] '<
```

```
'INCHES: 10 10 <
```

```
'PSI : 10.2 10.2 <
```

```
IF (INCHES_L >= 10) & (INCHES_R >= 10) THEN
```

```
SEROUT PORTA.7, 2, [$FE, 1, " LEFT RIGHT"]
```

```
SEROUT PORTA.7, 2, [$FE, $C0, "INCHES: ", #INCHES_L, " ", #INCHES_R ]
```

```
ENDIF
```

```
IF (INCHES_L < 10) & (INCHES_R >= 10) THEN
```

```
SEROUT PORTA.7, 2, [$FE, 1, " LEFT RIGHT"]
```

```
SEROUT PORTA.7, 2, [$FE, $C0, "INCHES: ", #INCHES_L, " ", #INCHES_R]
```

```
ENDIF
```

```
IF (INCHES_L >= 10) & (INCHES_R < 10) THEN
```

```
SEROUT PORTA.7, 2, [$FE, 1, " LEFT RIGHT"]
```

```
SEROUT PORTA.7, 2, [$FE, $C0, "INCHES: ", #INCHES_L, " ", #INCHES_R]
```

```
ENDIF
```

```
IF (INCHES_L < 10) & (INCHES_R < 10) THEN
```

```
SEROUT PORTA.7, 2, [$FE, 1, " LEFT RIGHT"]
```

```
SEROUT PORTA.7, 2, [$FE, $C0, "INCHES: ", #INCHES_L, " ", #INCHES_R]
```

```
ENDIF
```

```
SEROUT PORTA.7, 2, [$FE, $94, "PSI : ", #PRS1, ".", #PRS1DEC, " ", #PRS2, ".",  
#PRS2DEC]
```

```
SEROUT PORTA.7, 2, [$FE, $D4, "ANY KEY TO RETURN"]
```

```
KEYIN = PORTB & $0F
```

```
IF (KEYIN > 0) THEN
```

```
MENU_STATE = 0 'GOTO MAIN MENU
```

ENDIF

PAUSE 150

WEND

RETURN

MENU2:

PAUSE 600

WHILE (MENU_STATE == 2)

ADCIN 0, INCHES_L

ADCIN 1, INCHES_R

GOSUB MATH

'12345678901234567890" <

'SEROUT PORTA.7, 2, [\$FE, 1, "1MAIN- LEFT RIGHT"] '<

'INCHES : 10 10 <

'NEW SET: 10 10 <

'2:SET L:6U7D R:8U9D

SEROUT PORTA.7, 2, [\$FE, 1, " 1MAIN- LEFT RIGHT"]

IF (INCHES_L >= 10) & (INCHES_R >= 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$C0, "INCHES : ", #INCHES_L, " ", #INCHES_R]

ENDIF

IF (INCHES_L < 10) & (INCHES_R >= 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$C0, "INCHES : ", #INCHES_L, " ", #INCHES_R]

ENDIF

IF (INCHES_L >= 10) & (INCHES_R < 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$C0, "INCHES : ", #INCHES_L, " ", #INCHES_R]

ENDIF

IF (INCHES_L < 10) & (INCHES_R < 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$C0, "INCHES : ", #INCHES_L, " ", #INCHES_R]

ENDIF

IF (L_ADJ >= 10) & (R_ADJ >= 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$94, "NEW SET: ", #L_ADJ, " ", #R_ADJ]

ENDIF

IF (L_ADJ < 10) & (R_ADJ >= 10) THEN

SEROUT PORTA.7, 2, [\$FE, \$94, "NEW SET: ", #L_ADJ, " ", #R_ADJ]

ENDIF

IF (L_ADJ >= 10) & (R_ADJ < 10) THEN

```
SEROUT PORTA.7, 2, [$FE, $94, "NEW SET: ", #L_ADJ, " ", #R_ADJ]
ENDIF
```

```
IF (L_ADJ < 10) & (R_ADJ < 10) THEN
SEROUT PORTA.7, 2, [$FE, $94, "NEW SET: ", #L_ADJ, " ", #R_ADJ]
ENDIF
```

```
SEROUT PORTA.7, 2, [$FE, $D4, "2:SET L:6U7D R:8U9D"]
```

```
GOSUB KEY_MATH
```

```
PAUSE 150
WEND
```

```
MATH: 'MATH CONVERTS ABSTRACT A/D SIGNALS TO INCHES AND PSI, DECIMAL
PSI AS WELL
```

```
INCHES_L = INCHES_L/5
INCHES_R = INCHES_R/5
```

```
PRS1 = PRS1RAW/7
PRS1DEC = PRS1//7
PRS1DEC = PRS1DEC*10 / 7
```

```
PRS2 = PRS2RAW/7
PRS2DEC = PRS2//7
PRS2DEC = PRS2DEC*10 / 7
RETURN
```

```
KEY_MATH:
```

```
KEYIN = PORTB & $0F
IF (KEYIN == 6) THEN
PAUSE 400
L_ADJ = L_ADJ +1
ENDIF
```

```
IF (KEYIN == 7) THEN
PAUSE 400
L_ADJ = L_ADJ -1
ENDIF
```

```
IF (KEYIN == 8) THEN
PAUSE 400
R_ADJ = R_ADJ +1
```

ENDIF

```
IF (KEYIN == 9) THEN
  PAUSE 400
  R_ADJ = R_ADJ -1
ENDIF
```

```
IF (L_ADJ > 38) THEN 'VALUE CANNOT EXCEED 37
L_ADJ = 37
ENDIF
```

```
IF (L_ADJ < 26) THEN 'VALUE CANNOT GO BELOW 26
L_ADJ = 26
ENDIF
```

```
IF (R_ADJ > 38) THEN
  R_ADJ = 37
ENDIF
```

```
IF (R_ADJ < 26) THEN
  R_ADJ = 26
ENDIF
```

```
IF (KEYIN == 1) THEN '1 TO RETURN TO MAIN MENU
  PAUSE 400
  MENU_STATE = 0
ENDIF
```

```
IF (KEYIN == 2) THEN 'SERIAL TO CONTROLS PIC, IF 2 PRESSED
  PAUSE 400
  SEROUT PORTA.6, 2, ["A", L_ADJ]
  SEROUT PORTA.6, 2, ["B", R_ADJ]
ENDIF
WRITE 0, L_ADJ 'SAVE VALUES TO EEPROM
WRITE 1, R_ADJ
RETURN
```