(The activities in this document are excerpts from the forthcoming Stamps in Class book *Smart Sensors and Applications*, by Andy Lindsay.)

ACTIVITY #4: REMOTE DATALOGGING ACCELERATION

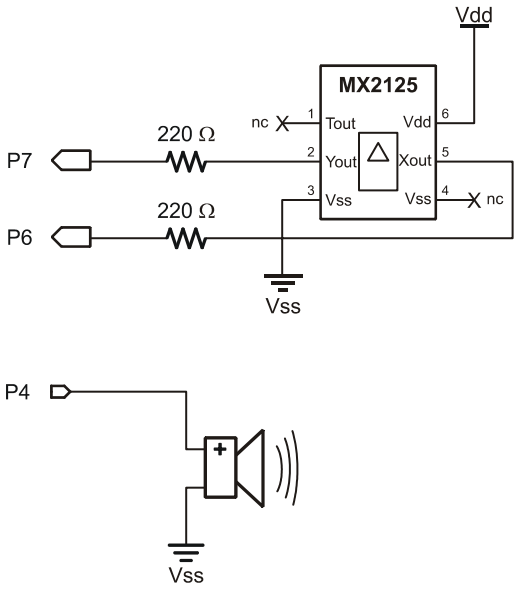
In this activity, you will add a piezospeaker to the existing accelerometer circuit. Then, you will modify the program so that it provides you with a remote datalogging tool that's easy to operate. The piezospeaker will be be handy for indicating countdown, start, and stop. The accelerometer circuit will the same one used in Chapter #3, and the piezospeaker will be added below it on the breadboard.

Parts Required

(1) Memsic 2125 Accelerometer
(2) 220 Ω resistors
(1) piezospeaker
(7) jumper wires

Circuit

√ Build it the accelerometer and piezospeaker circuits shown in Figure 6-3.



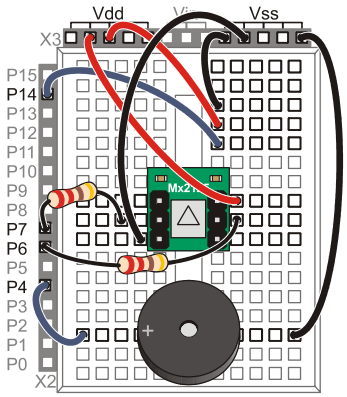


Figure 6-3
Accelerometer and Piezospeaker
Schematic (left) and
Wiring Diagram (below)

Program Modifications

The next example program, DataloggingAccleration.bs2, is an expansion of EepromDataStorageWithReset.bs2. It has been modified so that you can start, stop and restart datalogging with your board's Reset button. You can disconnect the board from your computer to perform the datalogging, and reconnect it to display the measurements in the Debug Terminal. This is a crucial feature for taking field measurements and then displaying them later.

DataloggingAcceleration.bs2 has a modified initialization section that makes the piezospeaker beep every second for ten seconds before it starts datalogging.

```
' -----[ Initialization ]-------------------------------------------

Init:
  .
  .
  .
  FOR char = 10 TO 0
  .
  .
  .
    FREQOUT 4, 50, 3750
    PAUSE 950
  NEXT
  .
  .
  .
```

DataloggingAccleration.bs2 also has a modified **Record_Data** subroutine that gets the x and y values from the accelerometer, scales them to (-100 to 100), and writes both of them to EEPROM. The **FOR**...**NEXT** loop increments in steps of 2 with the **STEP 2** argument since each time through the loop, the routine saves two bytes. The **Display_Data** subroutine has similar modifications so that it displays both the x and y values in a table.

```
Record_Data:

  FREQOUT 4, 75, 4000
  PAUSE 200
  FREQOUT 4, 75, 4000

  DEBUG CLS, "Recording..."
```

```
FOR eeIndex = Records TO RecordsEnd STEP 2

  PULSIN 6, 1, x
  PULSIN 7, 1, y

  x = (x MIN 1875 MAX 3125) - 1875 ** 10538
  y = (y MIN 1875 MAX 3125) - 1875 ** 10538

  WRITE eeIndex, x
  WRITE eeIndex + 1, y

NEXT

FREQOUT 4, 200, 4000

DEBUG CR, "End of records.",
      CR, "Press Enter for menu..."
DEBUGIN char
RETURN
```

The piezospeaker also beeps twice at a higher pitch right at the beginning of the datalogging. One important feature of this ten-second countdown is that you can stop the datalogging before it starts by simply pressing and releasing your board's Reset button. Then, to restart the countdown, just press and release the Reset button again.

Example Program: DatalogAcceleration.bs2

Free Download! This program is available as a free .bs2 file download from the Smart Sensors and Applications Product Page at www.parallax.com.

This program takes and stores 500 accelerometer x and y-axis measurements in about 15 seconds. This equates to a sampling rate of about 33 measurements per second. This is good for a variety of measurements. To measure longer and slower processes, the **`Record_Data`** subroutine can be slowed down with a **`PAUSE`** command.

- √ Open and run DatalogAcceleration.bs2.
- √ Click the Debug Terminal's transmit windowpane.
- √ Type R to start recording, and tilt your accelerometer this way and that for fifteen seconds.
- √ When prompted, press Enter to return to the program's menu.

- √ Type D to display the measurements. Review them and verify that they correspond to how you tilted the accelerometer.
- √ Disconnect your board from the serial cable. If it starts beeping as you do so, press and release the reset button to make it stop.

When you are ready to start tilting the accelerometer for fifteen seconds, press and release the Reset button. The datalogger will beep for a ten second countdown, then end with two higher-pitched beeps signaling the start of the datalogging. It will make a single high-pitched beep when it's finished.

- √ Press and release the reset button. Wait the ten seconds, then tilt your accelerometer in a pattern that you can remember for 15 seconds.
- √ Plug your accelerometer back into your computer. If it starts beeping, press and release the reset button to stop the countdown.
- √ Click the BASIC Stamp Editor's Run button to download the program to the BASIC Stamp and refresh the Debug Terminal's Menu display.
- √ Type D to display the datalogged measurements.
- √ Compare them to the directions you tilted the board and make sure they correspond.

```
' -----[ Title ]-----------------------------------------------------------
' Smart Sensors and Applications - DatalogAcceleration.bs2
' Datalogs 500 x and y-axis acceleration measurements.

'{$STAMP BS2}
'{$PBASIC 2.5}

' -----[ DATA Directives ]-------------------------------------------------

Reset          DATA    0
Records        DATA    (1000)
RecordsEnd     DATA

' -----[ Variables ]-------------------------------------------------------

char           VAR     Byte
eeIndex        VAR     Word
value          VAR     Word
x              VAR     value
y              VAR     Word

' -----[ Initialization ]--------------------------------------------------

Init:
```

```
READ Reset, value
value = value + 1
WRITE Reset, value

IF value // 2 = 0 THEN

  FOR char = 10 TO 0
    DEBUG CLS, "Datalogging starts", CR,
               "in ", DEC2 char, " seconds",
               CR, CR,
               "Press/release Reset", CR,
               "for menu..."
    FREQOUT 4, 50, 3750
    PAUSE 950
  NEXT

  GOSUB Record_Data

ENDIF

' -----[ Main Routine ]-----------------------------------------------------

DO

  DEBUG CLS,
        "Press/Release Reset", CR,
        "to arm datalogger ", CR, CR,
        " - or - ", CR, CR,
        "Type C, R or D", CR,
        "C - Clear records", CR,
        "R - Record records", CR,
        "D - Display records", CR,
        ">"

  DEBUGIN char
  DEBUG CR

  SELECT char
    CASE "C", "c"
      GOSUB Clear_Data
    CASE "R", "r"
      GOSUB Record_Data
    CASE "D", "d"
      GOSUB Display_Data
    CASE ELSE
      DEBUG CR, "Not a valid entry.",
            CR, "Try again."
      PAUSE 1500
  ENDSELECT
```

```
LOOP

' -----[ Subroutine - Clear_Data ]----------------------------------------

Clear_Data:
  DEBUG CR, "Clearing..."
  FOR eeIndex = Records TO RecordsEnd
    WRITE eeIndex, 0
  NEXT
  DEBUG CR, "Records cleared."
  PAUSE 1000
  RETURN

' -----[ Subroutine - Record_Data ]----------------------------------------

Record_Data:

  FREQOUT 4, 75, 4000
  PAUSE 200
  FREQOUT 4, 75, 4000

  DEBUG CLS, "Recording..."

  FOR eeIndex = Records TO RecordsEnd STEP 2

    PULSIN 6, 1, x
    PULSIN 7, 1, y

    x = (x MIN 1875 MAX 3125) - 1875 ** 10538
    y = (y MIN 1875 MAX 3125) - 1875 ** 10538

    WRITE eeIndex, x
    WRITE eeIndex + 1, y

  NEXT

  FREQOUT 4, 200, 4000

  DEBUG CR, "End of records.",
        CR, "Press Enter for menu..."
  DEBUGIN char

  RETURN

' -----[ Subroutine - Display_Data ]---------------------------------------

Display_Data:

  DEBUG CR, "Index  x-axis  y-axis",
        CR, "-----  ------  ------",
        CR
```

```
FOR eeIndex = Records TO RecordsEnd STEP 2
  READ eeIndex, x
  x = x - 100
  READ eeIndex + 1, y
  y = y - 100
  DEBUG DEC eeIndex, CRSRX, 7, SDEC x, CRSRX, 14, SDEC y, CR
NEXT
DEBUG CR, "Press Enter for menu..."
DEBUGIN char
RETURN
```

Your Turn - Datalogging Rotation Angle

Chapter 3, Activity #5 introduced vertical rotation measurements with the accelerometer. Since binary radians are values from 0 to 255, you can store a single angle measurement in one EEPROM byte. This will double the number of measurements the application will take. It only takes a few modifications to DatalogAcceleration.bs2 to make it store rotation angle instead. Here's how:

- √ Save DatalogAccleration.bs2 as DatalogAngle.bs2.
- √ Update the comments in the Title section.
- √ Remove the **`STEP 2`** arguments from the **`FOR`**...**`NEXT`** loops in the **`Record_Data`** and **`Display_Data`** subroutines.
- √ In the **`Record_Data`** subroutine, replace these two **`WRITE`** commands:

```
WRITE eeIndex, x
WRITE eeIndex + 1, y
```

  ...with this **`ATN`** operation and **`WRITE`** command:

```
value = x ATN y
WRITE eeIndex, value
```

- √ Modify the display heading in the **`Display_Data`** subroutine so that it looks like this:

```
DEBUG CR, "Index  angle ",
      CR, "-----  ------",
      CR
```

- √ Replace these four commands:

```
READ eeIndex, x
x = x - 100
```

```
READ eeIndex + 1, y
y = y - 100
DEBUG DEC eeIndex, CRSRX, 7, SDEC x, CRSRX, 14, SDEC y, CR
```

...with these:

```
READ eeIndex, value
value = value */ 361
DEBUG DEC eeIndex, CRSRX, 7, SDEC x, CRSRX, 14, SDEC y, CR
```

√ Save your changes and test the modified program.