599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallaxinc.com
Technical: javelintech@parallaxinc.com
Web Site: www.javelinstamp.com
Other: www.parallaxinc.com

Javelin Application Note

Using an Analog Joystick

Introduction

Back before the popularity of USB ports, computers often came equipped with a game port designed to read an analog joystick. The Javelin, combined with a bit of code and a few passive components can be used to read the position and switch values from this type of joystick. While less common than they once were, these joysticks are still available and generally at a very low cost ($10 - $20).

Library Classes Used

- Joystick.java

Background

For each axis of the joystick movement (x and y), there is a potentiometer. The output of each pot is fed into a standard rcTime() circuit for the Javelin so that the value (position) of the pot can be measured. The switch outputs are normally open and connected to pulled-up input pins on the Javelin.

Program Explanation

The program starts by creating a joystick object on pins 0, 1, 2 and 3 (see schematic). An integer is created to hold the current axis reading and, finally, a string buffer is created to hold data for the message window.

The rest of the program is a simple loop that takes advantage of standard joystick methods: rawX(), rawY(), buttonX() and buttonY(). The rawX() and rawY() methods return the corresponding potentiometer reading from the joystick. If the value is less than zero, the measurement timed-out. This could mean that the capacitor used in the rcTime() circuit is too large, or you're using a non-standard joystick. The circuit shown the schematic is designed for joysticks with 100K – 200K potentiometers.

A joystick axis value is read into the variable *axisVal*. This way, the program can check for a timeout error (–1) from the joystick object. Under normal circumstance, there will be no error and the current axis value is appended to the output message.

The buttonX() and buttonY() values return True if the corresponding button is pressed, otherwise they will return False. If the button under test is not being pressed, the word "not" is inserted into the [active] sentence "... button is pressed."

Code Listing

```
// Joystick demonstration program
// -- by Jon Williams
// -- Applications Engineer, Parallax
// -- jwilliams@parallaxinc.com
// -- Updated: 22 July 2002

/* Joystick connections:
 *
 *   X Pot    --> Javelin.P0 (rcTime() circuit -- 0.1 uF connected to Vdd)
 *   Y Pot    --> Javelin.P1 (rcTime() circuit -- 0.1 uF connected to Vdd)
 *   X Switch --> Javelin.P2 (active low [pulled up through 10K])
 *   Y Switch --> Javelin.P3 (active low [pulled up through 10K])
 */

import stamp.core.*;
import stamp.peripheral.Joystick;

public class demoJoystick {

  final static char CLR_SCR = '\u0010';

  public static void main() {

    // create joystick object
    joystick controller = new joystick(CPU.pin0, CPU.pin1, CPU.pin2, CPU.pin3);
    // intermediate holder for each pot reading
    int axisVal;
    // create message buffer for screen display
    StringBuffer msg = new StringBuffer();

    while (true) {
      // create and display measurement message
      msg.clear();
      msg.append(CLR_SCR);
      msg.append("Joystick Demo\n\n");

      // X Pot
      msg.append("X position = ");
      axisVal = controller.rawX();
      if (axisVal >= 0)
        msg.append(axisVal);
      else
        msg.append("Error");
      msg.append("\n");

      // Y Pot
      msg.append("Y position = ");
      axisVal = controller.rawY();
      if (axisVal >= 0)
        msg.append(axisVal);
      else
        msg.append("Error");
      msg.append("\n\n");

      // X Button
      msg.append("X button is ");
      if (!controller.buttonX()) msg.append("not ");
```

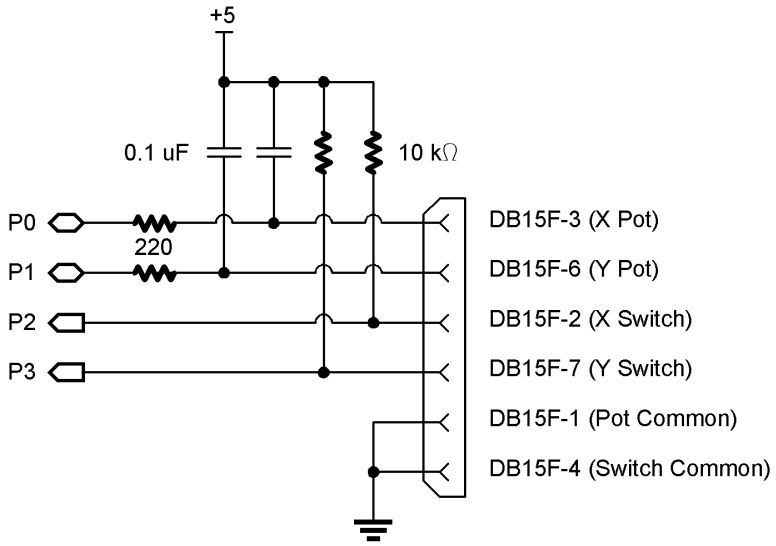
```
      msg.append("pressed.\n");

      // Y Button
      msg.append("Y button is ");
      if (!controller.buttonY()) msg.append("not ");
      msg.append("pressed.\n\n");

      System.out.print(msg.toString());

      // wait 0.1 seconds between readings
      CPU.delay(1000);
    }
  }
}
```

Schematic



For Advanced Users

The joystick class provides a means of scaling the X and Y output values. For each axis there is a [private] multiplier and divisor. These values are set to one when the object is created. By changing these values, the programmer can cause the scaleX() and scaleY() values to have a top-end value different from the raw reading. The scale value is calculated with this formula:

```
scale_value = raw_value * multiplier / divisor
```

The programmer must use caution with the multiplier to prevent the intermediate result (raw * multiplier) from exceeding 32,767. Exceeding this value will cause a negative result (an error) in the scaled output.

Scaling parameters can be set with the following methods: setXMult(m), setXDiv(d), setYMult(m), setYDiv(d), setXScale(m, d), setYScale(m, d).

Sources

- Gravis "Destroyer" analog joystick purchased from CompUSA

Reference Documents and Information

- "The Nuts & Volts of BASIC Stamps" – Volume 2
  Experiment #70

- http://www.epanorama.net/documents/joystick/pc_joystick.html

- http://www.seetron.com/guest/berg/berg_joyssc.htm

Author Information

- Jon Williams
  Applications Engineer, Parallax
  jwilliams@parallaxinc.com

- Published: July 22, 2002