

# Rudder Controller & Wiper Controller

## Embedded System Design, Lab 2

Ben Lorenzetti

September 25, 2015

### Contents

<b>1</b>	<b>Objectives and Problem Description</b>	<b>2</b>
1.1	Rudder Controller . . . . .	2
1.2	Wiper Controller . . . . .	2
<b>2</b>	<b>Procedure</b>	<b>2</b>
2.1	Assignment: Ch. 4 Servo Motor . . . . .	2
2.2	Assignment: Ch.5 Potentiometers . . . . .	3
2.3	Traffic Light Controller . . . . .	4
2.4	Jeopardy Style Game . . . . .	4
<b>3</b>	<b>Expected Results</b>	<b>5</b>
<b>4</b>	<b>Experiment and Design Revisions</b>	<b>5</b>
4.1	Traffic Light Controller . . . . .	5
4.2	Jeopardy Game . . . . .	5
<b>5</b>	<b>Observations</b>	<b>6</b>
<b>6</b>	<b>Discussion</b>	<b>7</b>
<b>7</b>	<b>Exercises</b>	<b>7</b>
<b>8</b>	<b>Programs</b>	<b>10</b>
8.1	Traffic Light Controller . . . . .	10
8.2	Jeopardy Style Game . . . . .	11

# 1 Objectives and Problem Description

The primary objective of this lab was learning to interface with two new types of peripherals: a servo motor and potentiometers. Several new functions in PBASIC are introduced for these types of actuators and transducers.

## 1.1 Rudder Controller

Control the angular position of a tiny ship's rudder (i.e. the horn of your servo) based on the input from a steering wheel (i.e. your potentiometer). The full range of motion of the ship's rudder should be  $45^\circ < \theta < 135^\circ$  and it should be controlled linearly by the full range of the potentiometer.

## 1.2 Wiper Controller

Instead of controlling angular position, as was done for the rudder controller, use the potentiometer to control the angular velocity of the servo. The servo represents a single windshield wiper, and should rotate continuously between  $15^\circ$  and  $160^\circ$ . The full range of the potentiometer should be used to linearly control the wiper speed from 0 degrees/sec up to the maximum speed.

# 2 Procedure

## 2.1 Assignment: Ch. 4 Servo Motor

To prepare for this lab, we were instructed to read chapters 4 and 5 in the Parallax What's a Microcontroller book and complete the example activities.

The parallax servo motor has three connections: power, ground, and signal. The signal (white) connection is used to control the angular position of the horn, from  $0^\circ$  to  $180^\circ$ . The servo expects a square wave control signal, where the period is approximately 20 ms and the width of the high pulse carries angular displacement information.

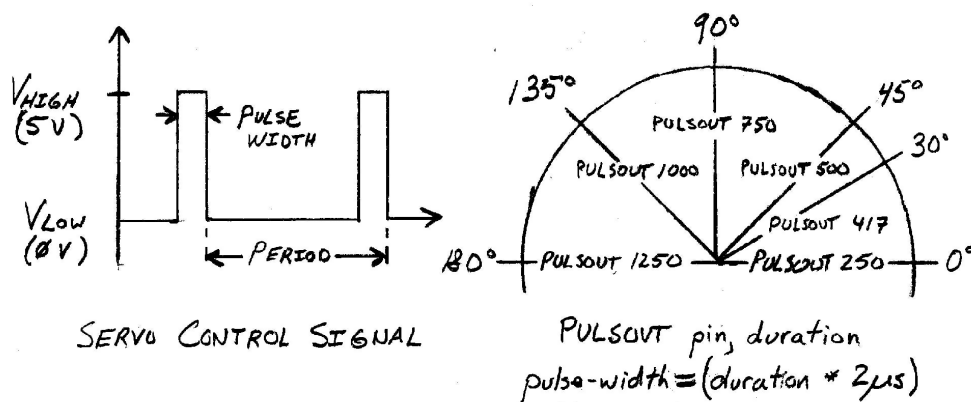


Figure 1: Servo Control Signal Timing

The formula relating pulse width and angular displacement is

$$t_{p-width}(\theta) = \begin{cases} \frac{250\mu s}{45^\circ} \theta + 250\mu s & 10^\circ < \theta < 170^\circ \\ 0\mu s & \text{servo off} \end{cases} \quad (1)$$

The function for doing pulse width modulation in PBASIC is `PULSOUT pin, duration`, where `pin` is the I/O port number and `duration` is the length of time high. Of course, since this Stamp and PBASIC, the board isn't quite fast enough so `duration` is in units of  $2\mu s$  and you have to complete the pulse width modulation yourself by `PAUSE`ing for 20 ms, or however long the period should be approximately.

To avoid dealing with PBASIC's strange limitations, we can rewrite equation 1 as

$$\text{duration} = \frac{25}{9} * \theta + 125, \quad 10^\circ < \theta < 170^\circ \quad (2)$$

Chapter 4 also provided two more expressions in PBASIC that are helpful with servo motors:

```
i VAR Word
FOR i = 0 TO 150
...
LOOP
}
```

and `duration = duration MIN 350 MAX 1150`.

## 2.2 Assignment: Ch.5 Potentiometers

Not surprisingly, the Basic Stamp is too primitive to have an analog-to-digital converter, so you can only measure analog inputs with a function called `RCTIME`, and then only if you construct an RC decay circuit—and again only if the input signal is stable for a long enough charging time and can source enough current.

The function prototype is `RCTIME pin, state, duration`, where `pin` is the I/O pin number (0-15), `state` is the expected charged status of the pin (0 or 1), and `duration` is a variable to store the decay time—in units of  $2\mu s$ . The decay time is taken when the voltage on `pin` falls below 1.4V, if the original charge `state` was 1.

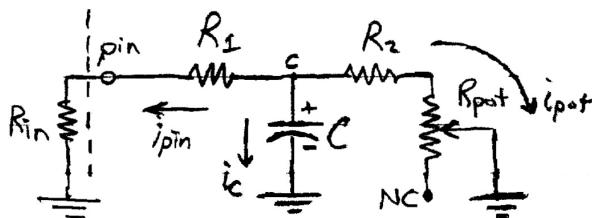


Figure 2: Input Circuit for Reading a Potentiometer with `RCTIME`

An input source that is well suited to `RCTIME` is a rotary potentiometer. A circuit combining a potentiometer and capacitor is shown in figure 2.

If we assume infinite input resistance once `RCTIME` switches the pin direction to input, then  $i_{pin} = 0$  and  $V_{in} = V_c$ . Taking Kirchhoff's Current Law at node `c` allows us to solve for input voltage as a function of time and initial condition across `C`.

$$i_{pin} + i_c + i_{pot} = 0$$

$$0 + C \frac{dV_{in}}{dt} + \frac{V_{in}}{R_2 + R_{pot}} = 0$$

$$\begin{aligned}
\frac{-V_{in}}{R_2 + R_{pot}} &= C \frac{dV_{in}}{dt} \\
\frac{-1}{C(R_2 + R_{pot})} * dt &= \frac{1}{V_{in}} * dV_{in} \\
\frac{-t}{C(R_2 + R_{pot})} + A &= \ln(V_{in}) \\
e^{\ln(V_{in})} &= e^{\frac{-t}{C(R_2 + R_{pot})} + A} \\
V_{in} &= e^A * e^{-t/C(R_2 + R_{pot})} \\
V_{in}(t) &= V_0 e^{-t/\tau} \quad \tau = (R_2 + R_{pot})C
\end{aligned} \tag{3}$$

Using equation 3, the decay time required to discharge from  $V_{HIGH}$  to  $V_{LOW}$  is

$$\begin{aligned}
V_{LOW} &= V_{HIGH} e^{-t_{discharge}/\tau} \\
t_{discharge} &= \tau * \ln\left(\frac{V_{HIGH}}{V_{LOW}}\right)
\end{aligned}$$

For  $V_{HIGH} = 5V$ ,  $V_{LOW} = 1.4V$ ,  $C = 3,300\mu F$ , this evaluates to

$$t_{discharge} = \left[ \frac{2.1}{10^3} * \frac{2\mu s}{\Omega} \right] * (R_2 + R_{pot}) \tag{4}$$

A  $10k\Omega$  potentiometer with  $R_2 = 220\Omega$ , leads to input domain  $220\Omega < R_2 + R_{pot} < 10.22k\Omega$  and output range  $0.462 * 2\mu s < t_{discharge} < 21.462 * 2\mu s$ . When we consider that the PBA-SIC Stamp board only has an integer resolution of  $2\mu s$ , then the expected result from a call to `RCTIME`, `pin`, `1`, `duration` is in the set

$$\text{duration} = \{1, 2, 3, \dots, 21, 22\} \tag{5}$$

when the  $10k\Omega$  potentiometer is used with a  $3,300\mu F$  capacitor.

## 2.3 Traffic Light Controller

The lab said to go through the first 3 chapters of the book and do the activities, so I did this.

I built the circuit shown in figure 3 on the Basic Stamp breadboard. It has the 6 LEDs in active low configuration with  $1k\Omega$  current limiting resistors. The pushbutton switch was also configured active low for consistency. It also has a  $1k\Omega$  current limiting resistor.

The control flow chart for the PBasic program is shown in figure 4.

## 2.4 Jeopardy Style Game

I designed the circuit for the Jeopardy game similarly to the traffic light controller. Each LED or pushbutton switch has a  $1k\Omega$  current limiting resistor and is configured active low. The circuit diagram for the Jeopardy game is shown in figure 5.

The jeopardy game was more complicated than the traffic light controller, but only from a software perspective—not hardware.

I used the random function to generate a random delay time before the game begins, passing in the previous game's reaction time as a seed value so that the same pseudorandom sequence is not repeated every game.

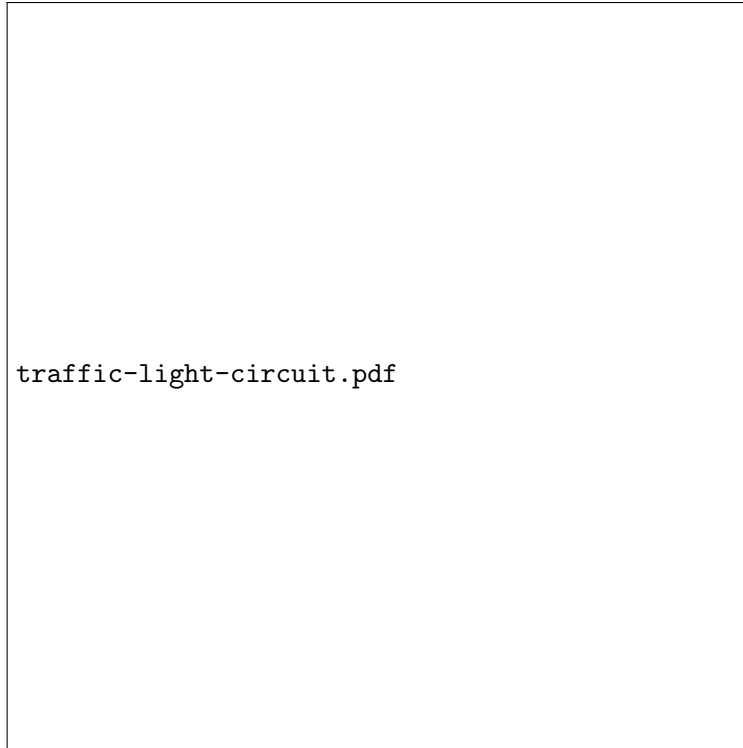


Figure 3: Traffic Light Controller Circuit Diagram

Once the game begins the microcontroller executes a tight loop that does checks the pushbuttons for a victor and then increments a `react_cycles` variable. When the first player buzzes in, the appropriate LED lights up and their reaction time is calculated from the number of cycles and the closest integer to the loop iteration time. This value is printed to the debug terminal.

The control flow chart for the PBasic program is shown in figure 6.

### 3 Expected Results

I expected both circuits to behave according to the problem statements.

## 4 Experiment and Design Revisions

### 4.1 Traffic Light Controller

This circuit and code worked like a charm. No revisions had to be made, but I did restyle the code to look more like C with boolean algebra.

### 4.2 Jeopardy Game

I really did not want to have two sets of Debug statements, depending on the winner. However, using only one Debug statement and storing the winner in a bit variable proved more difficult than I anticipated.

It was difficult to generate the boolean algebra statements for turning on the appropriate LEDs, mostly because PBasic sucks and doesn't even have a complete set of boolean operators.

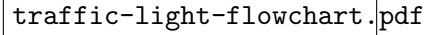
A rectangular box containing the text "traffic-light-flowchart.pdf". This box represents the flowchart for the Traffic Light Controller.

Figure 4: Traffic Light Controller Flowchart

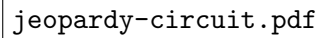
A large rectangular box containing the text "jeopardy-circuit.pdf". This box represents the circuit diagram for the Jeopardy board.

Figure 5: Jeopardy Circuit Diagram

So reducing to one set of Debut statements and only a single goto location proved more difficult, but I think it was worth the change for code brevity.

## 5 Observations

Both circuits worked as expected, and photos of the working boards are shown in figure 7 and figure 8. A screenshot of the Debug terminal for the Jeopardy board is shown in figure ??.



Figure 6: Jeopardy-Like Game Flowchart

## 6 Discussion

This lab really revealed a major flaw of the Stamp Basic board: the fact that running an interpreter prevents you from knowing exact timing. If we were writing code directly for PIC, we could know the clock rate and write some assembly inside the reaction time loop—and thereby know the exact number of clock cycles and the clock rate.

I have a low opinion of the Stamp Basic board right now, at least compared to other micro-controllers. This week I complain about not having control of clock cycles or there being a clock variable that is accessible. Next week's dissatisfaction preview: no ADC on board.

## 7 Exercises

There were no exercises given.



Figure 7: Traffic Light Controller Circuit

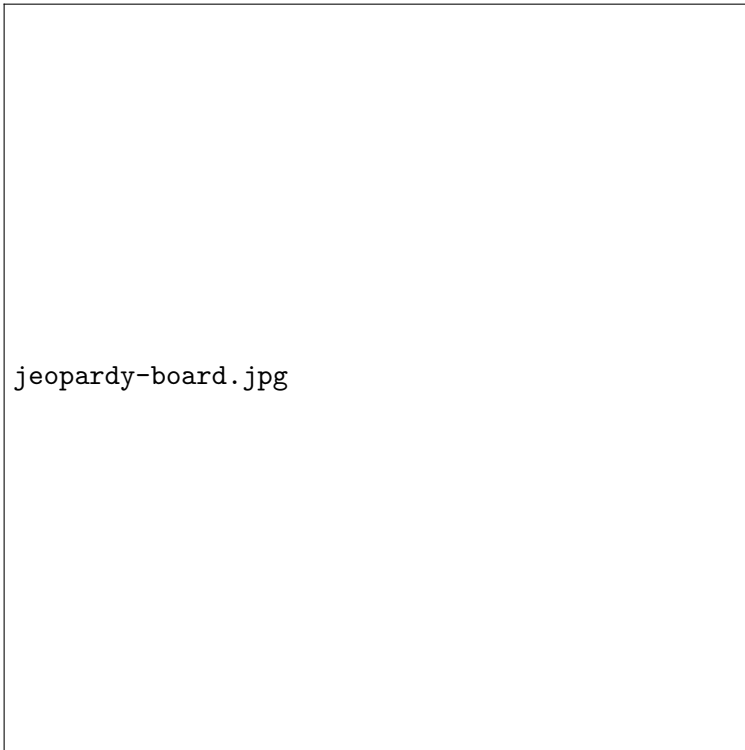


Figure 8: Jeopardy Game Circuit





Figure 9: Screenshot of Jeopardy Debug Terminal with Reaction Times

## 8 Programs

### 8.1 Traffic Light Controller

## 8.2 Jeopardy Style Game