Relay and PWM Control Program

The purpose of this note is to explain the ROBOTC program used to control the relay and PWM on the extended testbed, and to provide background information on the overall functioning of the system.

Relays
Before understanding how the ROBOTC program works, it is necessary to understand the basic purpose of a relay. As described in this lesson’s helper link, a relay is placed in the middle of two electronic circuits. When the current in the first circuit rises above a certain value, the relay acts to either open or close the second circuit. Opening the second circuit would, of course, stop the flow of current inside that circuit. Conversely, closing the second circuit would allow the flow of current. The flow of current is necessary for the operation of any devices, such as motors, within the circuit.

In the extended testbed, the Vex microcontroller and its battery are in the first circuit, while the large motor and the car battery are in the second circuit. When the Vex microcontroller sends a certain voltage (5 volts in this case), the relay acts to close the second circuit, which provides electron flow and causes the large motor to act.

Response of Relay to Electronic Input
We just learned that the relay in our testbed allows power to flow to the large motor when it receives a specific electronic input from the microcontroller. But how does the relay “interpret” these electronic signals?

Take a look at the cable that connects the microcontroller to the relay. On the relay end, you will see that the white wire enters the outside position, the red wire enters the middle position, and the black wire enters the inside position. The microcontroller acts by transmitting 5 volt signals down the white and red wires, and the relay responds according to the location of these signals. For example, if the relay receives a 5 volt signal in its central position (which would be through the red wire) it responds by powering the motor to full speed in the reverse direction. If the relay receives a 5 volt signal in its outside position (which would be through the white wire) it responds by powering the motor to full speed in the forward direction. If the relay receives 5 volt signals in both its outside and central positions, it responds by stopping the motor. A complete list of input signals and relay responses is shown in the table below:

<table>
<thead>
<tr>
<th>Outside Relay Position</th>
<th>Central Relay Position</th>
<th>Action of Motor</th>
<th>Indicator Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 V</td>
<td>0 V</td>
<td>no movement</td>
<td>orange</td>
</tr>
<tr>
<td>5 V</td>
<td>0 V</td>
<td>full speed forward</td>
<td>green</td>
</tr>
<tr>
<td>0 V</td>
<td>5 V</td>
<td>full speed reverse</td>
<td>red</td>
</tr>
<tr>
<td>5 V</td>
<td>5 V</td>
<td>no movement</td>
<td>off</td>
</tr>
</tbody>
</table>

You may be wondering why we are ignoring the black wire, which connects to the inside position on the relay. As in all vex cables, the black wire serves as the “ground” connection, which means that it does not transmit any voltage and is not involved in electronic input.
Relay and PWM Control Program

**Micron controller Signal Transmission**

We now know that the relay reacts to small voltage signals originating from the microcontroller. Let’s take a closer look at how the microcontroller sends these signals.

Again, look at the cable that connects the microcontroller to the relay. This time, trace the wires back to the microcontroller. Notice that the white wire is placed in analog/digital port 3, while the red wire is placed in analog/digital port 4. Let’s imagine that the microcontroller is told to send an output signal from digital port 4. We can see that the signal would begin at the red wire where it is connected to the microcontroller, and then travel along until it reaches the central position on the relay, at which point the relay would respond by powering the motor at full speed in the reverse direction. The following table illustrates how the digital ports on the microcontroller influence the behavior of the relay:

<table>
<thead>
<tr>
<th>Microcontroller Digital Port</th>
<th>Outside Relay Position</th>
<th>Central Relay Position</th>
<th>Action of Motor</th>
<th>Indicator Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>0 V</td>
<td>0 V</td>
<td>no movement</td>
<td>orange</td>
</tr>
<tr>
<td>3</td>
<td>5 V</td>
<td>0 V</td>
<td>full speed forward</td>
<td>green</td>
</tr>
<tr>
<td>4</td>
<td>0 V</td>
<td>5 V</td>
<td>full speed reverse</td>
<td>red</td>
</tr>
<tr>
<td>3 &amp; 4</td>
<td>5 V</td>
<td>5 V</td>
<td>no movement</td>
<td>off</td>
</tr>
</tbody>
</table>

Interestingly, this knowledge helps us to understand the ROBOTC program written for relay control. We have seen that the relay acts after it receives a signal from digital ports 3 and 4 on the microcontroller. Therefore, we would expect that a program in ROBOTC should work by telling the microcontroller to send signals from digital ports 3 and 4. In fact, that is exactly what the program does. The variables relay1 and relay2 are declared equal to digital outputs 3 and 4 below.

```cpp
const tSensors relay1 = (tSensors) in3; //sensorDigitalOut
const tSensors relay2 = (tSensors) in4; //sensorDigitalOut
```

Then two if else statements below program the values of relay1 and relay2 to be 1, and therefore for the digital outputs for ports 3 and 4 to send a 5 V signal, respectively, if the two uppermost buttons on the back of the remote control (vexRT[Ch5] and vexRT[Ch6]) are pressed, and also for them to not send any signal if they are not.

```cpp
if(vexRT[Ch5] == 127)
{
    SensorValue(relay1) = 1;
}
else
{
    SensorValue(relay1) = 0;
}
```
Relay and PWM Control Program

```c
if(vexRT[Ch6] == 127)
{
    SensorValue(relay2) = 1;
}
else
{
    SensorValue(relay2) = 0;
}
```

Transmitter Buttons and ROBOTC Program

As we’ve seen, the ROBOTC program allows you to use the Vex transmitter to control the microcontroller’s digital output signals. For example, pressing the up arrow of channel 5 (right side on back of transmitter) causes a digital output to be sent from port 3, which travels along the white wire to the outside port of the relay, and finally causes the motor to spin full speed in the forward direction. A final table is presented below to organize all of the information in this lesson:

<table>
<thead>
<tr>
<th>Transmitter Channel</th>
<th>Microcontroller Digital Port</th>
<th>Outside Relay Position</th>
<th>Central Relay Position</th>
<th>Action of Motor</th>
<th>Indicator Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>none</td>
<td>0 V</td>
<td>0 V</td>
<td>nothing</td>
<td>orange</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5 V</td>
<td>0 V</td>
<td>full speed forward</td>
<td>green</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>0 V</td>
<td>5 V</td>
<td>full speed reverse</td>
<td>red</td>
</tr>
<tr>
<td>5 &amp; 6</td>
<td>3 &amp; 4</td>
<td>5 V</td>
<td>5 V</td>
<td>nothing</td>
<td>off</td>
</tr>
</tbody>
</table>

Once you understand the ROBOTC program, you have the ability to alter your control over the testbed. For example, if you wished to control the motors with the joysticks on the front of the transmitter rather than with the buttons on the back, you could substitute vexRT[Ch2] and vexRT[Ch3] for vexRT[Ch5] and vexRT[Ch6] in the conditions above.

Summary of Relay Control

The ROBOTC program says that the up buttons on the back of the transmitter (channels 5 and 6) will cause a 5 volt signal to be sent from digital ports 3 and 4 on the Vex microcontroller. These small signals will travel up the white and red wires of the cable until the reach the relay’s input port. The relay will then act according to location of the voltage input. The action of the relay will dictate the direction of the motor, as well as the color of the indicator light.

While the information presented above may be confusing, a firm understanding of its basic concepts will prove valuable when working with the extended testbed.
Relay and PWM Control Program

**PWM Control**

Fortunately, controlling PWMs with the microcontroller is much simpler than controlling relays. In fact, the ROBOTC code for controlling the PWMs is actually only one line long! Let’s take a look at the code and figure out what it means.

Go back to the ROBOTC program and find the first command within the while loop.

```c
motor[port2] = vexRT[Ch2];
```

This means the value for motor[port2] will equal the value of the vertical component of the right joystick on the remote control. You might ask, What’s the difference between this and the control I get over Vex motors using the remote control? The answer is, nothing! The Vex motors are PWMs as well, and operate just like the PWMs on the testbed. Try it and see.