```
#pragma config(Sensor, dgtl2,  rightFront,     sensorQuadEncoder)
#pragma config(Sensor, dgtl4,  rightFront,     sensorQuadEncoder)
#pragma config(Sensor, dgtl6,  leftFront,      sensorQuadEncoder)
#pragma config(Sensor, dgtl8,  leftFront,      sensorQuadEncoder)
#pragma config(Sensor, dgtl10, solenoid1,      sensorDigitalOut)
#pragma config(Sensor, dgtl11, solenoid,       sensorDigitalOut)
#pragma config(Motor,  port2,           rightFront,    tmotorVex393_MC29, openLoop, reversed)
#pragma config(Motor,  port3,           leftFront,     tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port4,           intakeR,       tmotorVex393TurboSpeed_MC29, openLoop)
#pragma config(Motor,  port5,           liftL,         tmotorVex393_MC29, openLoop, reversed)
#pragma config(Motor,  port6,           liftR,         tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port7,           rightBack,     tmotorVex393_MC29, openLoop, reversed)
#pragma config(Motor,  port8,           leftBack,      tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port9,           claw,          tmotorVex393_MC29, openLoop, reversed)
#pragma config(Motor,  port10,          intakeL,       tmotorVex393TurboSpeed_HBridge,
openLoop, reversed)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

#pragma platform(VEX)

//Competition Control and Duration Settings


#include "Vex_Competition_Includes.c"   //Main competition background code...do not modify!

const short leftButton = 1;
const short centerButton = 2;
const short rightButton = 4;

//Wait for Press---------------------------------------------------
void waitForPress()
{
	while(nLCDButtons == 0){}
	wait1Msec(5);
}
//-----------------------------------------------------------------

//Wait for Release-------------------------------------------------
void waitForRelease()
{
	while(nLCDButtons != 0){}
	wait1Msec(5);
}
```

```
//----------

//////////////////////////////
// Pre-Autonomous Functions //
//////////////////////////////

void pre_auton()
{


		//Declare count variable to keep track of our choice
		int count = 0;

		//------------- Beginning of User Interface Code ---------------
		//Clear LCD
		clearLCDLine(0);
		clearLCDLine(1);
		//Loop while center button is not pressed
		while(nLCDButtons != centerButton)
		{
			//Switch case that allows the user to choose from 4 different options
			switch(count){
			case 0:
				//Display first choice
				displayLCDCenteredString(0, "22 point right");
				displayLCDCenteredString(1, "<                Enter            >");
				waitForPress();
				//Increment or decrement "count" based on button press
				if(nLCDButtons == leftButton)
				{
					waitForRelease();
					count = 3;
				}
				else if(nLCDButtons == rightButton)
				{
					waitForRelease();
					count++;
				}
				break;
			case 1:
				//Display second choice
				displayLCDCenteredString(0, "22 point left ");
				displayLCDCenteredString(1, "<                Enter            >");
```

```
            waitForPress();
            //Increment or decrement "count" based on button press
            if(nLCDButtons == leftButton)
            {
                  waitForRelease();
                  count = --;
            }
            else if(nLCDButtons == rightButton)
            {
                  waitForRelease();
                  count++;
            }
            break;
      case 2:
            //Display third choice
            displayLCDCenteredString(0, "7 point right ");
            displayLCDCenteredString(1, "<                Enter           >");
            waitForPress();
            //Increment or decrement "count" based on button press
            if(nLCDButtons == leftButton)
            {
                  waitForRelease();
                  count--;
            }
            else if(nLCDButtons == rightButton)
            {
                  waitForRelease();
                  count++;
            }
            break;
      case 3:
            //Display fourth choice
            displayLCDCenteredString(0, "7 point left ");
            displayLCDCenteredString(1, "<                Enter           >");
            waitForPress();
            //Increment or decrement "count" based on button press
            if(nLCDButtons == leftButton)
            {
                  waitForRelease();
                  count--;
            }
            else if(nLCDButtons == rightButton)
            {
```

```
                                        waitForRelease();
                                        count = 0;
                                }
                                break;
                        default:
                                count = 0;
                                break;
                        }
                }
        }
}


task autonomous()

{
int count = 0;
        clearLCDLine(0);
        clearLCDLine(1);
        //Switch Case that actually runs the user choice
        switch(count){
        case 0:
                //If count = 0, run the code correspoinding with choice 1
                displayLCDCenteredString(0, "22 point right blue");
                displayLCDCenteredString(1, "was selected!");
                wait1Msec(50);                                          // Robot waits for
2000 milliseconds

                // Move forward at full power for 3 seconds
                {
                        clearLCDLine(0);
                        clearLCDLine(1);
                        // Robot waits for 500 milliseconds before executing program
                        resetMotorEncoder(rightFront);
                        resetMotorEncoder(leftFront);           // Set the encoder so that it starts
counting at 0
                        {
                                wait1Msec(50);
// Robot waits for 500 milliseconds before executing program
                                SensorValue[rightFront] = 0;
                                SensorValue[rightFront] = 0;    // Set the encoder so that it starts
counting at 0
                        }
```

```
                        // Creates an infinite loop, since "true" always evaluates to true

                        if(SensorValue[rightFront] == SensorValue[leftFront]) // If rightEncoder
has counted the same amount as leftEncoder:
                        {
                                // Move Forward
                                motor[rightFront] = -127;
                                motor[rightBack] = -127;                        // Right Motor is run
at power level 80
                                motor[leftFront] = -127;
                                motor[leftBack]  = -127;                                // Left Motor is
run at power level 80
                        }
                        else if(SensorValue[rightFront] > SensorValue[leftFront])    // If
rightEncoder has counted more encoder counts
                        {
                                // Turn slightly right
                                motor[rightFront] = -123;               // Right Motor is run at
power level 125
                                motor[rightBack]  = -127;
                                motor[leftFront] = -123;                // Right Motor is run at
power level 125
                                motor[leftBack]  = -127;
                        }
                        else    // Only runs if leftEncoder has counted more encoder counts
                        {
                                // Turn slightly left
                                motor[rightFront] = -127;               // Right Motor is run at
power level 127
                                motor[rightBack]  = -123;
                                motor[leftFront] = -127;                // Right Motor is run at
power level 127
                                motor[leftBack]  = -123;
                        }
                        // Left Motor is run at power level 125

                        {
                                motor[liftR] = -75;
                                motor[liftL] = -75;    //move cone out of way of mobile goal intake
                                motor[claw] = 100;
                                wait1Msec(2850);

                                motor[rightFront] = 0;
```

```
motor[rightBack] = 0;
motor[leftFront] = 0;
motor[leftBack] = 0;
motor[intakeL] = 127;  //pick up mobile goal
motor[intakeR] = 127;
motor[claw] = 100;
wait1Msec(1500);

motor[intakeL] = 0;
motor[intakeR] = 0;   // release driver load
motor[claw] = 100;
wait1Msec(300);

motor[rightFront] = 127;
motor[rightBack] = 127;
motor[leftFront] = 127;
motor[leftBack] = 127;
motor[liftR] = 127;  // Move toward the 20-point zone
motor[liftL] = 127;
motor[intakeL] = 50;
motor[intakeR] = 50;
motor[claw] = 100;
wait1Msec(2500);

motor[claw] = -127;
motor[liftR] = 0;
motor[liftL] = 0;  // release cone
wait1Msec(300);

motor[leftFront] = 127;
motor[leftBack] = 127;
motor[rightFront] = -20;  // turn to position for mobile goal drop off
motor[rightBack] = -20;
motor[liftR] = 0;
motor[liftL] = 0;

wait1Msec(650);

motor[rightFront] = 127;
motor[rightBack] = 127;
motor[leftFront] = 127;    // line up robot for final spin
motor[leftBack] = 127;
motor[intakeL] = 127;
```

```
                    motor[intakeR] = 127;
                    wait1Msec(1200);


                    motor[leftFront] = 127;
                    motor[leftBack] = 127;
                    motor[rightFront] = -127; //spin the robot into the correct angle for
mobile goal drop off
                    motor[rightBack] = -127;
                    motor[liftL] = 0;
                    motor[liftR] = 0;
                    wait1Msec(1050);

                    motor[rightFront] = -127;
                    motor[leftFront] = -127;    //drive back in to the 20 point zone and
reposition the lift
                    motor[rightBack] = -127;
                    motor[leftBack] = -127;

                    motor[liftL] = -127;
                    motor[liftR] = -127;
                    motor[claw] = -127;
                    wait1Msec(1000);
                    motor[rightFront] = -127;
                    motor[leftFront] = -127;    //drive back in to the 20 point zone and
reposition the lift
                    motor[rightBack] = -127;
                    motor[leftBack] = -127;
                    motor[intakeL] = -100;    //drop off mobile goal
                    motor[intakeR] = -100;
                    wait1Msec(600);

                    motor[intakeL] = 30;    //drop off mobile goal
                    motor[intakeR] = 30;

                    wait1Msec(650);
                    motor[intakeL] = -30;    //drop off mobile goal
                    motor[intakeR] = -30;

                    wait1Msec(450);
```

```
                motor[intakeL] = 127;
                motor[intakeR] = 127; //drive out of the zone
                motor[rightFront] = 127;
                motor[leftFront] = 127;
                motor[rightBack] = 127;
                motor[leftBack] = 127;
                wait1Msec(500);

                //drive out of the zone
                motor[rightFront] = 127;
                motor[leftFront] = 127;
                motor[rightBack] = 127;
                motor[leftBack] = 127;
                wait1Msec(500);
                motor[intakeL] = -100;
                motor[intakeR] = -100; //drive out of the zone

                wait1Msec(500);
                motor[rightFront] = 0;
                motor[leftFront] = 0;   //stop motors
                motor[rightBack] = 0;
                motor[leftBack] = 0;   //stop motors
                motor[intakeL] = 0;
                motor[intakeR] = 0;
                wait1Msec(100000);

            }
        }
    // Robot runs previous code for 3000 milliseconds before moving on
        break;
    case 1:
        //If count = 1, run the code correspoinding with choice 2
        displayLCDCenteredString(0, "22 point left blue");
        displayLCDCenteredString(1, "was selected!");
        wait1Msec(50);                                          // Robot waits for
2000 milliseconds

        // Move reverse at full power for 3 seconds
        {
            clearLCDLine(0);
            clearLCDLine(1);
            // Robot waits for 500 milliseconds before executing program
```

```
                resetMotorEncoder(rightFront);
                resetMotorEncoder(leftFront);           // Set the encoder so that it starts
counting at 0
                {
                        wait1Msec(50);
// Robot waits for 500 milliseconds before executing program
                        SensorValue[rightFront] = 0;
                        SensorValue[rightFront] = 0;    // Set the encoder so that it starts
counting at 0
                }
                // Creates an infinite loop, since "true" always evaluates to true

                if(SensorValue[rightFront] == SensorValue[leftFront]) // If rightEncoder
has counted the same amount as leftEncoder:
                {
                        // Move Forward
                        motor[rightFront] = -127;
                        motor[rightBack] = -127;                        // Right Motor is run
at power level 80
                        motor[leftFront] = -127;
                        motor[leftBack]  = -127;                                // Left Motor is
run at power level 80
                }
                else if(SensorValue[rightFront] > SensorValue[leftFront])    // If
rightEncoder has counted more encoder counts
                {
                        // Turn slightly right
                        motor[rightFront] = -123;               // Right Motor is run at
power level 125
                        motor[rightBack]  = -127;
                        motor[leftFront] = -123;                // Right Motor is run at
power level 125
                        motor[leftBack]  = -127;
                }
                else    // Only runs if leftEncoder has counted more encoder counts
                {
                        // Turn slightly left
                        motor[rightFront] = -127;               // Right Motor is run at
power level 127
                        motor[rightBack]  = -123;
                        motor[leftFront] = -127;                // Right Motor is run at
power level 127
                        motor[leftBack]  = -123;
```

```
}
// Left Motor is run at power level 125

{
	motor[liftR] = -75;
	motor[liftL] = -75;    //move cone out of way of mobile goal intake
	motor[claw] = 100;
	wait1Msec(2850);

	motor[rightFront] = 0;
	motor[rightBack] = 0;
	motor[leftFront] = 0;
	motor[leftBack] = 0;
	motor[intakeL] = 127;  //pick up mobile goal
	motor[intakeR] = 127;
	motor[claw] = 100;
	wait1Msec(1500);

	motor[intakeL] = 0;
	motor[intakeR] = 0;   // release driver load
	motor[claw] = 100;
	wait1Msec(300);

	motor[rightFront] = 127;
	motor[rightBack] = 127;
	motor[leftFront] = 127;
	motor[leftBack] = 127;
	motor[liftR] = 127;  // Move toward the 20-point zone
	motor[liftL] = 127;
	motor[intakeL] = 50;
	motor[intakeR] = 50;
	motor[claw] = 100;
	wait1Msec(2500);

	motor[claw] = -127;
	motor[liftR] = 0;
	motor[liftL] = 0;  // release cone
	wait1Msec(300);

	motor[leftFront] = -20;
	motor[leftBack] = -20;
	motor[rightFront] = 127;  // turn to position for mobile goal drop off
	motor[rightBack] = 127;
```

```
motor[liftR] = 0;
motor[liftL] = 0;

wait1Msec(650);

motor[rightFront] = 127;
motor[rightBack] = 127;
motor[leftFront] = 127;    // line up robot for final spin
motor[leftBack] = 127;
motor[intakeL] = 127;
motor[intakeR] = 127;
wait1Msec(1200);


motor[leftFront] = -127;
motor[leftBack] = -127;
motor[rightFront] = 127; //spin the robot into the correct angle for
mobile goal drop off
motor[rightBack] = 127;
motor[liftL] = 0;
motor[liftR] = 0;
wait1Msec(1050);

motor[rightFront] = -127;
motor[leftFront] = -127;    //drive back in to the 20 point zone and
reposition the lift
motor[rightBack] = -127;
motor[leftBack] = -127;

motor[liftL] = -127;
motor[liftR] = -127;
motor[claw] = -127;
wait1Msec(1000);
motor[rightFront] = -127;
motor[leftFront] = -127;    //drive back in to the 20 point zone and
reposition the lift
motor[rightBack] = -127;
motor[leftBack] = -127;
motor[intakeL] = -100;    //drop off mobile goal
motor[intakeR] = -100;
wait1Msec(600);

motor[intakeL] = 30;    //drop off mobile goal
```

```
                motor[intakeR] = 30;

                wait1Msec(650);
                motor[intakeL] = -30;    //drop off mobile goal
                motor[intakeR] = -30;

                wait1Msec(450);



                motor[intakeL] = 127;
                motor[intakeR] = 127; //drive out of the zone
                motor[rightFront] = 127;
                motor[leftFront] = 127;
                motor[rightBack] = 127;
                motor[leftBack] = 127;
                wait1Msec(500);

                //drive out of the zone
                motor[rightFront] = 127;
                motor[leftFront] = 127;
                motor[rightBack] = 127;
                motor[leftBack] = 127;
                wait1Msec(500);
                motor[intakeL] = -100;
                motor[intakeR] = -100; //drive out of the zone

                wait1Msec(500);
                motor[rightFront] = 0;
                motor[leftFront] = 0;   //stop motors
                motor[rightBack] = 0;
                motor[leftBack] = 0;   //stop motors
                motor[intakeL] = 0;
                motor[intakeR] = 0;
                wait1Msec(100000);

            }
        }
        break;
case 2:
        //If count = 2, run the code correspoinding with choice 3
        displayLCDCenteredString(0, "7 point right blue");
```

```
		displayLCDCenteredString(1, "was selected!");
		wait1Msec(50);                                          // Robot waits for
2000 milliseconds

		//Turn right for 3 seconds
		{
			clearLCDLine(0);
			clearLCDLine(1);
			// Robot waits for 500 milliseconds before executing program
			resetMotorEncoder(rightFront);
			resetMotorEncoder(leftFront);		// Set the encoder so that it starts
counting at 0
			{
				wait1Msec(50);
// Robot waits for 500 milliseconds before executing program
				SensorValue[rightFront] = 0;
				SensorValue[rightFront] = 0;	// Set the encoder so that it starts
counting at 0
			}
			// Creates an infinite loop, since "true" always evaluates to true

			if(SensorValue[rightFront] == SensorValue[leftFront]) // If rightEncoder
has counted the same amount as leftEncoder:
			{
				// Move Forward
				motor[rightFront] = -127;		// Right Motor is run at
power level 80
				motor[leftFront]  = -127;		// Left Motor is run at
power level 80
				motor[rightBack] = -127;		// Right Motor is run at
power level 80
				motor[leftBack]  = -127;
			}
			else if(SensorValue[rightFront] > SensorValue[leftFront])	// If
rightEncoder has counted more encoder counts
			{
				// Turn slightly right
				motor[rightFront] = -123;		// Right Motor is run at
power level 125
				motor[leftFront]  = -127;		// Left Motor is run at
power level 127
				motor[rightBack] = -123;		// Right Motor is run at
power level 125
```

```
                motor[leftBack]  = -127;
        }
        else    // Only runs if leftEncoder has counted more encoder counts
        {
                // Turn slightly left
                motor[rightFront] = -127;                   // Right Motor is run at
power level 127
                motor[leftFront]  = -123;
                motor[rightBack] = -127;                    // Right Motor is run at
power level 127
                motor[leftBack]  = -123;
        }
        // Left Motor is run at power level 125

        {
                motor[liftR] = -75;
                motor[liftL] = -75;    //move cone out of way of mobile goal intake
                motor[claw] = -100;
                wait1Msec(2850);

                motor[rightFront] = 0;
                motor[leftFront] = 0;
                motor[rightBack] = 0;
                motor[leftBack] = 0;
                motor[intakeL] = 127;  // Pick up mobile goal
                motor[intakeR] = 127;
                wait1Msec(1500);

                motor[intakeL] = 0;
                motor[intakeR] = 0;   // release driver load
                motor[claw] = -100;
                wait1Msec(200);

                motor[rightFront] = 127;
                motor[leftFront] = 127;
                motor[rightBack] = 127;
                motor[leftBack] = 127;
                motor[liftR] = 127;  // Move toward the 20-point zone
                motor[liftL] = 127;
                wait1Msec(2300);

                motor[rightFront] = -127;
                motor[leftFront] = 127;
```

```
                motor[rightBack] = -127;
                motor[leftBack] = 127;
                motor[liftR] = -127;  // Move toward the 20-point zone
                motor[liftL] = -127;
                motor[claw] = 100;
                wait1Msec(2000);
                motor[rightFront] = -127;
                motor[leftFront] = -127;
                motor[rightBack] = -127;
                motor[leftBack] = -127;
                wait1Msec(300);

                motor[intakeL] = -127;
                motor[intakeR] = -127;
                wait1Msec(500);
                motor[rightFront] = 127;
                motor[leftFront] = 127;
                motor[rightBack] = 127;
                motor[leftBack] = 127;
                wait1Msec(1000);
                motor[rightFront] = 0;
                motor[leftFront] = 0;
                motor[rightBack] = 0;
                motor[leftFront] = 0;
                wait1Msec(10000);

            }
        }
    case 3:
        //If count = 3, run the code correspoinding with choice 4
        displayLCDCenteredString(0, "7 point left blue");
        displayLCDCenteredString(1, "was selected");
        wait1Msec(50);                                          // Robot waits for
2000 milliseconds

        //Turn left for 3 seconds
        {
            clearLCDLine(0);
            clearLCDLine(1);
            // Robot waits for 500 milliseconds before executing program
            resetMotorEncoder(rightFront);
            resetMotorEncoder(leftFront);          // Set the encoder so that it starts
counting at 0
```

```
                {
                        wait1Msec(50);
// Robot waits for 500 milliseconds before executing program
                        SensorValue[rightFront] = 0;
                        SensorValue[rightFront] = 0;    // Set the encoder so that it starts
counting at 0
                }
                // Creates an infinite loop, since "true" always evaluates to true

                if(SensorValue[rightFront] == SensorValue[leftFront]) // If rightEncoder
has counted the same amount as leftEncoder:
                {
                        // Move Forward
                        motor[rightFront] = -127;               // Right Motor is run at
power level 80
                        motor[leftFront]  = -127;               // Left Motor is run at
power level 80
                        motor[rightBack] = -127;                // Right Motor is run at
power level 80
                        motor[leftBack]  = -127;
                }
                else if(SensorValue[rightFront] > SensorValue[leftFront])    // If
rightEncoder has counted more encoder counts
                {
                        // Turn slightly right
                        motor[rightFront] = -123;               // Right Motor is run at
power level 125
                        motor[leftFront]  = -127;               // Left Motor is run at
power level 127
                        motor[rightBack] = -123;                // Right Motor is run at
power level 125
                        motor[leftBack]  = -127;
                }
                else    // Only runs if leftEncoder has counted more encoder counts
                {
                        // Turn slightly left
                        motor[rightFront] = -127;               // Right Motor is run at
power level 127
                        motor[leftFront]  = -123;
                        motor[rightBack] = -127;                // Right Motor is run at
power level 127
                        motor[leftBack]  = -123;
                }
```

```
// Left Motor is run at power level 125

{
	motor[liftR] = -75;
	motor[liftL] = -75;    //move cone out of way of mobile goal intake
	motor[claw] = -100;
	wait1Msec(2850);

	motor[rightFront] = 0;
	motor[leftFront] = 0;
	motor[rightBack] = 0;
	motor[leftBack] = 0;
	motor[intakeL] = 127;  // Pick up mobile goal
	motor[intakeR] = 127;
	wait1Msec(1500);

	motor[intakeL] = 0;
	motor[intakeR] = 0;   // release driver load
	motor[claw] = -100;
	wait1Msec(200);

	motor[rightFront] = 127;
	motor[leftFront] = 127;
	motor[rightBack] = 127;
	motor[leftBack] = 127;
	motor[liftR] = 127;  // Move toward the 20-point zone
	motor[liftL] = 127;
	wait1Msec(2300);

	motor[rightFront] = 127;
	motor[leftFront] = -127;
	motor[rightBack] = 127;
	motor[leftBack] = -127;
	motor[liftR] = -127;  // Move toward the 20-point zone
	motor[liftL] = -127;
	motor[claw] = 100;
	wait1Msec(2000);
	motor[rightFront] = -127;
	motor[leftFront] = -127;
	motor[rightBack] = -127;
	motor[leftBack] = -127;
	wait1Msec(300);
```

```
                                motor[intakeL] = -127;
                                motor[intakeR] = -127;
                                wait1Msec(500);
                                motor[rightFront] = 127;
                                motor[leftFront] = 127;
                                motor[rightBack] = 127;
                                motor[leftBack] = 127;
                                wait1Msec(1000);
                                motor[rightFront] = 0;
                                motor[leftFront] = 0;
                                motor[rightBack] = 0;
                                motor[leftBack] = 0;
                                wait1Msec(10000);

                        }
                }                       // Robot runs previous code for 3000 milliseconds before
moving on
                break;
        default:
                displayLCDCenteredString(0, "No valid choice");
                displayLCDCenteredString(1, "was made!");
                break;
        }
        //------------- End of Robot Movement Code ----------------------
}


//////////////////////////
// Driver Control Task //
//////////////////////////

task usercontrol()
{

        int threshold = 5;

        while (true)
        {
                if(abs(vexRT[Ch3]) > threshold)        // If the left joystick is greater than or less
than the threshold:
                {
                        motor[leftFront]  = (vexRT[Ch3])/1;   // Left Joystick Y value / 2.
                        motor[leftBack]  = (vexRT[Ch3])/1;
```

```
		}
		else				// If the left joystick is within the threshold:
		{

			motor[leftFront]  = 0;
			motor[leftBack]  = 0;
			// Stop the left motor (cancel noise)
		}

		if(abs(vexRT[Ch2]) > threshold)		// If the right joystick is greater than or less
than the threshold:
		{
			motor[rightFront] = (vexRT[Ch2])/1;
			motor[rightBack] = (vexRT[Ch2])/1;
			// Right Joystick Y value / 2.
		}
		else				// If the right joystick is within the threshold:
		{
			motor[rightFront] = 0;
			motor[rightBack] = 0;
			// Stop the right motor (cancel noise) }



		}
		motor[liftL] = ((vexRT[Btn6U] * 127)) - ((vexRT[Btn6D] * 127));
		motor[liftR] = ((vexRT[Btn6U] * 127)) - ((vexRT[Btn6D] * 127));
		motor[claw] = ((vexRT[Btn5U] * 127)) - ((vexRT[Btn5D] * 127));
		motor[intakeR] = ((vexRT[Btn8D] * 127)) - ((vexRT[Btn8U] * 127));
		motor[intakeL] = ((vexRT[Btn8D] * 127)) - ((vexRT[Btn8U] * 127));

		if(vexRT[Btn7U] == 1)		// If button 7U (upper right shoulder button) is
pressed:
		{
			SensorValue[solenoid] = 1;
			SensorValue[solenoid1] = 1;
			wait1Msec(5000);				// ...activate the solenoid.
		}

		else(vexRT[Btn7D] == 1);		// If button 7D (upper right shoulder button) is
pressed:
		{
			(vexRT[Btn7D] == 1);
```

```
                    SensorValue[solenoid] = 0;
                    SensorValue[solenoid1] = 0;     // ..deactivate the solenoid.
                }


        }
}
```