```
#pragma config(Sensor, in1,    expander,       sensorAnalog)
#pragma config(Sensor, in2,    gyro,           sensorGyro)
#pragma config(Sensor, dgtl1,  armQuad,        sensorQuadEncoder)
#pragma config(Sensor, dgtl3,  gateSwitch,     sensorDigitalIn)
#pragma config(Sensor, dgtl4,  armSwitch,      sensorDigitalIn)
#pragma config(Sensor, dgtl5,  gateQuad,       sensorQuadEncoder)
#pragma config(Sensor, dgtl7,  quad,           sensorQuadEncoder)
#pragma config(Motor,  port2,           yOne,          tmotorVex393HighSpeed_MC29, openLoop, reversed)
#pragma config(Motor,  port3,           yTwo,          tmotorVex393HighSpeed_MC29, openLoop)
#pragma config(Motor,  port4,           yThree,        tmotorVex393HighSpeed_MC29, openLoop, reversed)
#pragma config(Motor,  port5,           gate,          tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port6,           frontRight,    tmotorVex393HighSpeed_MC29, openLoop, reversed)
#pragma config(Motor,  port7,           backRight,     tmotorVex393HighSpeed_MC29, openLoop, reversed)
#pragma config(Motor,  port8,           backLeft,      tmotorVex393HighSpeed_MC29, openLoop)
#pragma config(Motor,  port9,           frontLeft,     tmotorVex393HighSpeed_MC29, openLoop)
#pragma config(Motor,  port10,          intake,        tmotorVex393HighSpeed_HBridge, openLoop)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

#pragma platform(VEX)

//Competition Control and Duration Settings
#pragma competitionControl(Competition)
#pragma autonomousDuration(20)
#pragma userControlDuration(120)

#include "Vex_Competition_Includes.c"

void pre_auton()
{
  bStopTasksBetweenModes = true;
}

task autonomous()
{
	AutonomousCodePlaceholderForTesting();
}

task usercontrol()
{
	//This tests the RPM of a motor
	while (true)
	{
		motor[port4] = 127;
		SensorValue[armQuad] = 0;
```

```
                wait1Msec(1000);
                float rpm = (SensorValue[armQuad] / 360.0) * 60.0;
                writeDebugStream("%d\n", rpm);
        }
}
```