

Jaguar speed controllers

When used with CAN control, Jaguars are *smart* speed controllers. You can simply send a command to the Jaguar such as a position setpoint and it will use attached sensors to move to that position and maintain it. This relieves the robot controller from having to run control loops that read the sensors, determine the error, and supply correction values.

To use the Jaguar speed controller in one of the CAN modes, you set the speed controller mode to one of:

1. Voltage mode - you supply the positive or negative voltage that is within the bounds of the actual bus voltage (typically around 12V) that you would like the speed controller to output to the motor. The bus voltage will typically vary depending on how long the battery has been in use, so supplying absolute values will guarantee consistent output voltage provided that the request is less than or equal to the actual bus voltage.
2. Percentage mode - you supply a percentage of bus voltage expressed as a floating point between -1 and 1. The actual voltage supplied to the motor will vary depending on the charge of the battery and will actually change as the robot is operating.
3. Position mode - the CANJaguar object uses a specified sensor (either a quadrature encoder or potentiometer) and PID values to move the motor to a specified setpoint. All the closed loop control is done inside the speed controller itself so that your program doesn't need to hold the position in software.
4. Current mode - a current value is supplied in Amps which is effectively the requested motor torque (twisting effort). The Jaguar uses an internal current sensor and your PID values to keep the current to the requested value.
5. Speed mode - CANJaguar object uses a specified sensor (encoder or quadrature encoder) and PID values to move the motor at a requested speed in rotations per minutes (RPM).

When using speed or position mode a sensor is attached to the inputs on the Jaguar (not to the RoboRIO). This sensor is referred to as the **reference** device. This reference device is used to determine the error for closed loop feedback control. The choices of reference devices are potentiometers (for position mode), quadrature encoders (for position or speed mode), or single-input encoders (for speed mode). Even though there are separate inputs for potentiometers and encoders on the Jaguar, only one reference device can be in use at a time.

In addition, a reference device may be supplied for Voltage, Percentage, and Current modes that is not used for closed loop control, but can be read to monitor the performance of the speed controller.

Jaguar speed controllers

The Jaguar also has inputs for forward and reverse **limit switches**. The limit switches will turn the Jaguar off when the switch corresponding to the direction of motion is pressed (should be wired so that this opens the switch, also known as "Normally Open" or NO) while allowing the motor to operate in the opposite direction. This is used to limit the travel of a mechanism without writing any code. For example, when the switch at the top of travel for an arm is closed, the motor will stop moving in that direction. Which is forward and which is reverse depends on the motor polarity so it is a good idea to test to make sure that it is stopping the motor in the correct direction at the correct limit.

In all case, a new CANJaguar object is instantiated, the mode is selected, then you can begin using it in your program.

Creating Jaguar objects

Each physical Jaguar has a corresponding Jaguar object in either C++ or Java. Jaguar objects are created using the new operator in either language as follows:

```
m_jaguar = new CANJaguar(CANJaguarIDValue);
```

The value of *CANJaguarIDValue* is the CAN ID programmed in the web interface of the RoboRIO for the device being instantiated. **Be sure that the most recent firmware version is installed on the Jaguar.** After creating the Jaguar instance, the operating mode should be set and the `enableControl()` method called to set the operating mode as shown:

```
m_jaguar->SetPercentMode(CANJaguar::QuadEncoder, 360);  
m_jaguar->EnableControl();  
m_jaguar->Set(0.0f);
```

The control mode is not actually set until `EnableControl()` is called to give the program a chance to set other parameters specific to the operating mode. In this example the Jaguar is set to Percent Mode with a quadrature encoder as the reference sensor. Specifying the encoder in the `SetPercentMode()` method will allow the program to get feedback on the encoders current position as shown:

```
double initialPosition = m_jaguar->GetPosition();
```

Jaguar speed controllers

Using limits

There are two types of limits that can be used with the Jaguar:

1. Soft limits - reference device values that the Jaguar won't exceed in either direction such as a minimum or maximum number of encoder rotations.
2. Limit switches - switches that control the maximum movement of a mechanism

The soft limits can be enabled and disabled, the limit switch control is always operating in all Jaguar modes (including PWM). If limit switches are not used, jumpers should be installed to allow the Jaguar to operate properly. Refer to the Jaguar documentation for details on using the jumpers.

Using closed loop control in the Jaguar

Closed loop control means that the Jaguar looks at sensor inputs, computes an error value (difference between sensor value and setpoint), operates some actuators (motors) and loops. The Jaguars can perform this algorithm inside the device for Speed, Position and Current modes of operation. To use closed loop feedback you must do the following:

1. Set the requested mode using the SetMode() method
2. Supply P, I, and D constants in the correct set mode method
3. Supply a sensor appropriate for the operation
4. and call EnableControl() to start the controller running.

The first three operations are typically done with the SetMode() method - it is supplied with the sensor and P, I, and D constants.

Using voltage control

The output voltage can either be a percentage of the system bus voltage or an absolute voltage as described above. To set the Jaguar in Voltage mode use the following method:

```
m_jaguar->SetVoltageMode();
```

and to set it into Percent Voltage mode use this method:

```
m_jaguar->SetPercentMode();
```

Jaguar speed controllers

In either case a reference device may be specified as shown here:

```
m_jaguar->SetPercentMode(CANJaguar::QuadEncoder, 360);
```

In this case a quadrature encoder is selected as the reference device with a CPI (counts per revolution) value of 360. You can read the value of the encoder but it is not used for controlling the device - it is simply treated as an attached sensor.

Using position control

Position control can use either a potentiometer or quadrature encoder along with PID constants to move the motor to precise positions. When setting position mode, you need to specify the sensor (in this case a quadrature encoder), the number of counts per revolution, and the P, I, and D constants for the internal PID feedback loop.

```
m_jaguar->SetPositionMode(CANJaguar::QuadEncoder, 360, 10.0f, 0.4f, 0.2f);  
m_jaguar->EnableControl();  
double setpoint = m_jaguar->GetPosition() + 10.0f;  
m_jaguar->Set(setpoint);
```

This program puts the Jaguar into position mode and drives the motor to 10 encoder rotations forward. Remember, this is not necessarily motor shaft rotations since the encoder could be mounted on an intermediate gear of a transmission.

Using current control

Using speed control

Put the Jaguar into speed mode by calling the SetMode method as shown:

```
setSpeedMode(EncoderTag tag, int codesPerRev, double p, double i, double d)
```

The EncoderTag is one of: kEncoder, kQuadEncoder, or kPotentiometer. The codesPerRev argument is the number of counts per revolution that the encoder uses.

Ramping the speed

Jaguar speed controllers

Getting status from the Jaguar

Various status values can be read from the Jaguars as shown here:

```
m_jaguar->GetBusVoltage();  
m_jaguar->GetOutputVoltage();  
m_jaguar->GetOutputCurrent();  
m_jaguar->GetTemperature();  
  
m_jaguar->GetFaults();
```

These methods return various operating values from the Jaguar. The last method, GetFaults() returns a bit field with individual bits describing if there were faults of various types. For a full description of the faults and the other status methods refer to the header file or the JavaDocs for your language of choice.

When the Jaguar is operating, it is set to send status values back to the RoboRIO every 20ms without the program needing to request this service. This means that the status that you read from these methods may be as old as 20ms or more recent depending on when the last set of messages was received.