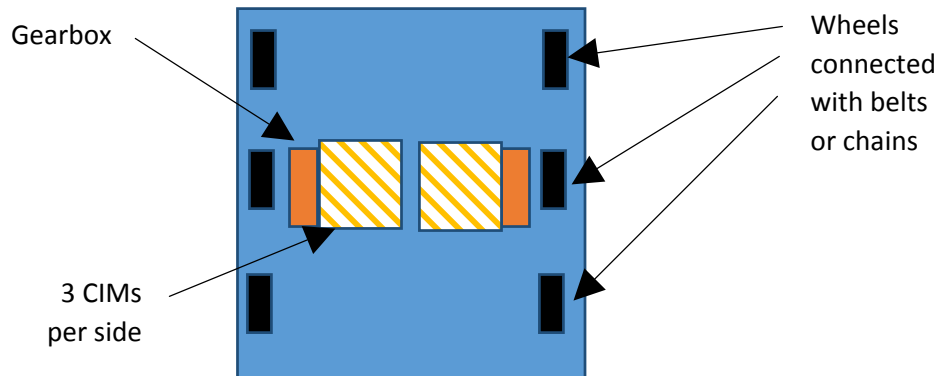


Example Talon SRX Feedforward and Feedback Motion Profile on a Six CIM Drive Robot

These code snippets and explanations are for an arcade drive robot that uses six CIM motors. The three motors on each side are integrated into a single gearbox with an integrated quadrature encoder. All three motors on a side move in the same direction to turn the wheels in a particular direction. The six motors are controlled with six Talon SRXs.



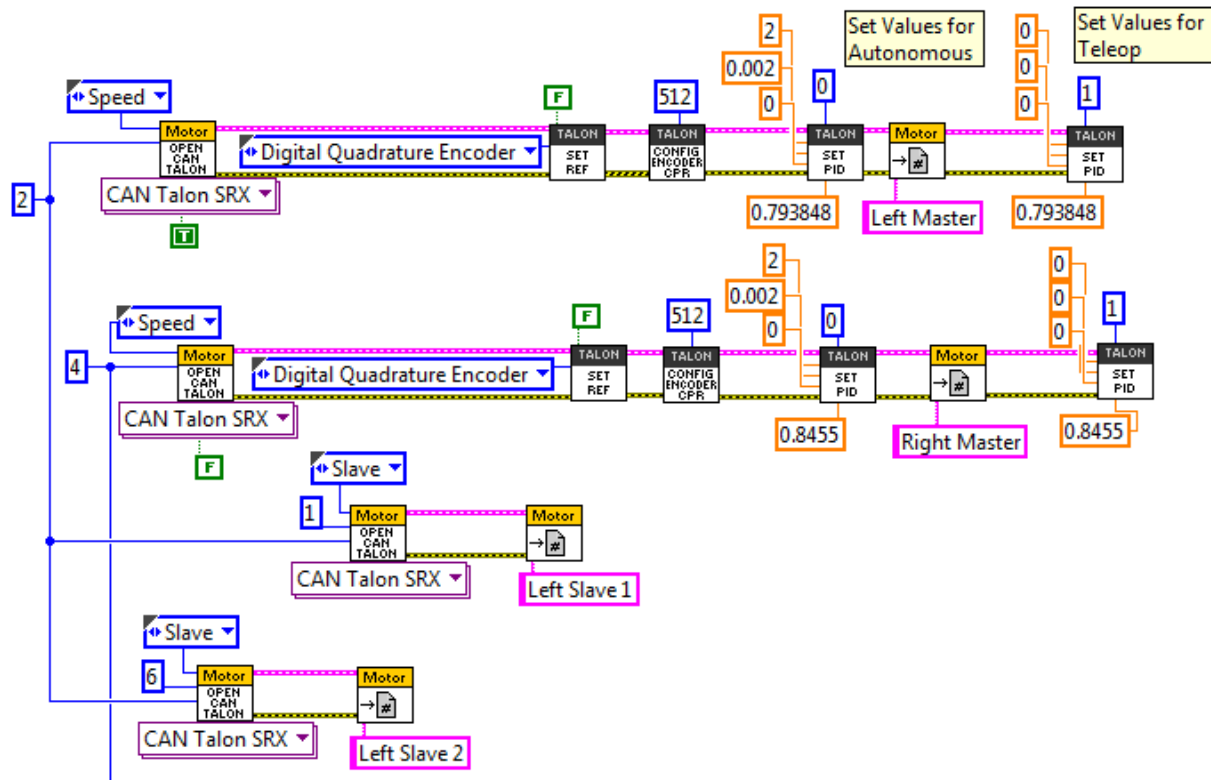
We want to set up our code to achieve certain consistencies for inputs and values and certain capabilities:

1. When positive values are sent to either side of the robot, the wheels move the robot forward.
2. When the wheels move the robot forward, the reported position and speed are positive.
3. Be able to use both feedforward and feedback control to achieve a motion profile during autonomous.

First, you need to check if your joystick or controller puts out positive or negative values when you move it in the “forward” direction. If it puts out negative values, invert that axis. In the same code location where deadbands are applied is a good place to do this.

Next, set up the six Talons. For this, you will need several pieces of information, and gathering some of this information requires running the robot. So, you will build some of this code, run the robot to measure or determine other parameters and then fill those in the code. For each side, there will be one master talon and two slaves. The master Talons need to have the quadrature encoders connected to them via the data ports.

Below is example code for Begin. For this example robot, the master Talon for the left side of the robot has CAN id #2 and the right side master Talon is CAN id #4. The first determination is which side of the robot to invert. In the example robot, when the Talons on the left side are running in the “green” direction, then the wheels are running backwards, and so the Invert Boolean is set for that master. Note that the Left slaves do not have the Invert Boolean set. Both Talons are set up for Speed Control Mode. This will allow us to send them target speeds for the feedforward and feedback control to achieve.



The Set Ref is used to indicate that digital quadrature encoders are the input sensors for both sides. It also indicates how the encoder counts relate to the direction of the Talons. In the case of this example robot, encoder counts increase when the Talons run in the “green” direction for both sides of the robot, so Reverse Feedback Sensor is false. Note that this is a relation between the sensor direction and the Talon direction, and nothing to do with wheel direction. This is important for the feedback control to work properly. This means that when we query the left side Talon for Sensor Speed and Sensor Position, we will get negative values when the wheels go forward. This is opposite of what we want for understanding data output, but it will be inverted later for human reading.

Next is some information from the encoder spec sheet. This example robot has encoders with 128 cycles per revolution. Since the Talon counts each transition on both channels (4X encoder) the Counts per Rotation is 512. This is a very important setting. By providing this number, all target control data you send the Talon will be in Revolutions for position and RPM for speed. If encoder revolutions are not 1:1 with wheel revolutions, then further adjustment of this constant will be necessary to convert to wheel revolutions.

Two sets of control gain settings are next. The Talon SRX has two slots for separate sets of control gains. These gains, once set, can be used on-demand. The first set of values (slot 0) we will use for autonomous, and the second set (slot 1) will be for Teleop.

First let’s discuss the F gain for feedforward. This is determined by running the robot at a known throttle speed and determining the speed measured by the Talon SRX via the encoder. This can be done either through the web dashboard or by logging data to a file on the RIO. Below is a sample screen capture from the web interface. Since I have already configured the Talon SRX with the number of encoder counts per revolution, you can see position and velocity in both native units and rotations/RPM.

```

Selected Device:0:Quad Encoder
Pos (rot): -33.483 Velocity (RPM): -394.33
Pos:-68574 Velocity:-1346

```

```

Quad Encoder (4x)
Pos (rot): -33.601 Velocity (RPM): -394.33
Pos:-68815 Velocity:-1346
Pins: A=1 B=0 Idx=1
Idx rise edges:0

```

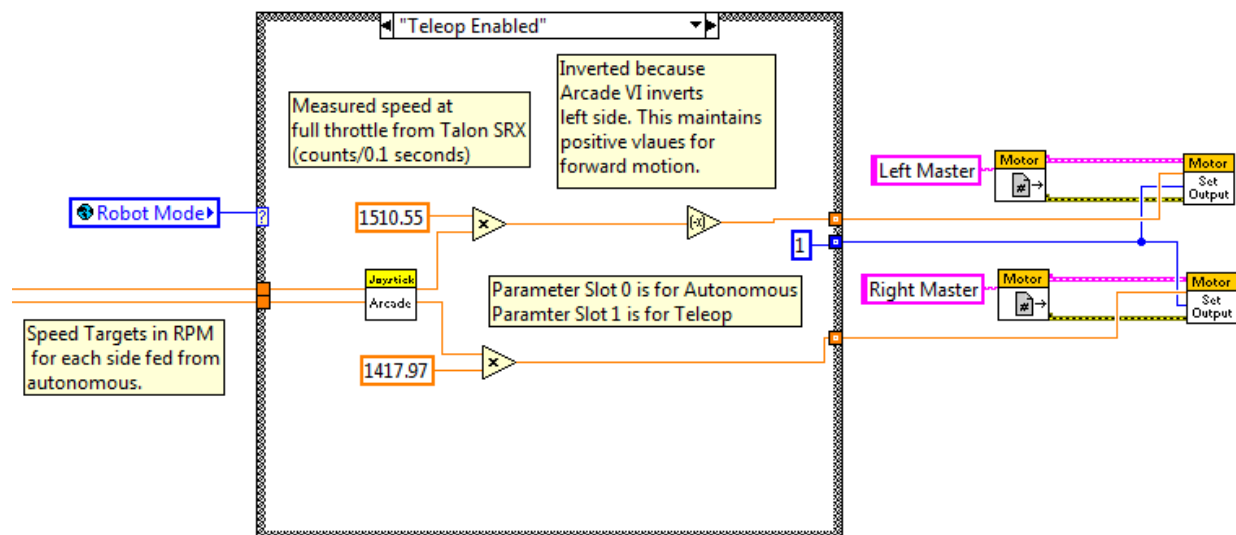
The F gain value should be in units of throttle divided by speed. Talon native units for throttle are +- 1023. Talon native units for speed are in counts per 0.1 seconds. For this example robot, data was logged to a file and then correlated. For the left side of the robot at full throttle, the Talon SRX reported a speed of 1210. Therefore the left side F gain is $1023/1210 = 0.8455$. The right side had a speed of 1250 for a throttle setting of 0.97. The right side F gain is $992.31/1250 = 0.793848$. The F gain allows us to send a target speed (in RPM) to the motor output VI and the Talon uses the F gain to convert to a throttle setting. It is a unit conversion.

The PID gains come later, but are easier than usual PID tuning with the feedforward gain properly set. Unlike the F gain, they are tuned, not calculated, and tuning them is outside of this example write-up.

Define the second set of gains for teleop. This works with the same F gains, but no PID. Since there is no closed loop control with only feedforward, the teleop settings in slot 1 do nothing more than change how we send throttle settings to motor output.

Finally, set up the Slave talons. Only the left side is shown in the code snippet. The right side is done identically (nothing is inverted or reversed).

Next is setting up the motor outputs (see code below). The first snippet is for teleop.



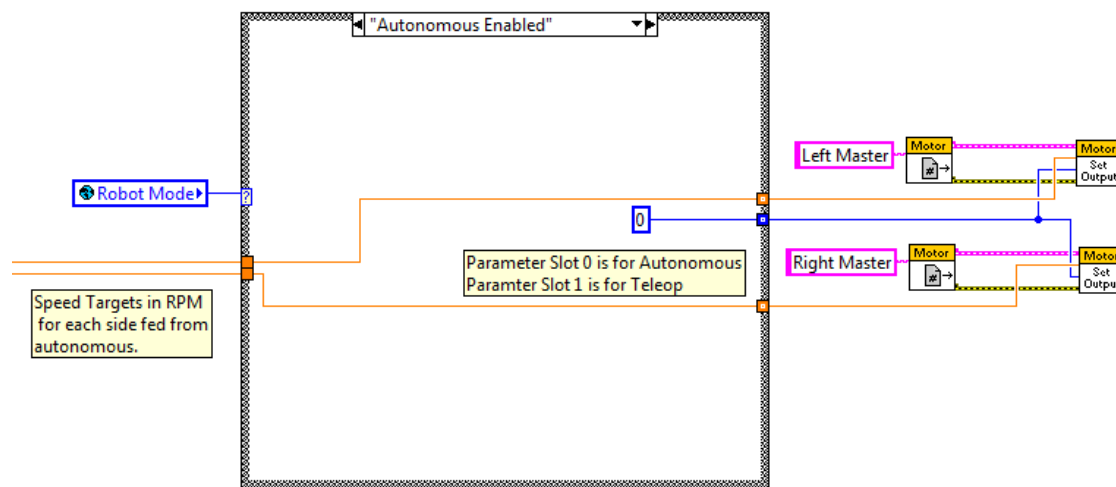
The key points here are that the standard Arcade output inverts the left motor (instead of inverting it in begin for a standard setup). Therefore we invert it here so that positive values to motor outputs yield forward rotation of the wheels. Recall that joystick inputs are ± 1 and so they need to be converted to RPM. For the right side, full throttle measured speed was 1210 counts per 0.1 seconds. Therefore:

$$(12100 \text{ counts per second}) / (512 \text{ counts per Revolution}) * (60 \text{ seconds per minute}) = 1417.97$$

For the left side, the speed of 1250 was at a throttle of 0.97 so this becomes 1289 at full throttle and the unit conversion is the same as for the right side yielding 1510.55

Finally, we are telling the motor outputs to use parameter slot 1 for teleop (feedforward only).

The next snippet is for autonomous.



Our autonomous routine feeds left and right speed targets in RPM already, so no unit conversion is needed. We just shift to parameter slot 0 so that the motor controllers use our combined feedforward and feedback control gains.

Below is our simple autonomous routine for the example robot. The timed loop reads speed targets from each row of a CSV file. These are sent to each side of the robot via the previous code snippet. Note that this VI includes charts so we can see the target and actual values of position and speed to aid in tuning the PID gains.

