

Common Device Classes

Talon SRX	com.ctre.phoenix.motorcontrol.can.WPI_TalonSRX	CTRE (Phoenix V5)
Victor SPX	com.ctre.phoenix.motorcontrol.can.WPI_VictorSPX	CTRE (Phoenix V5)
Talon FX	com.ctre.phoenix6.hardware.TalonFX	CTRE (Phoenix V6)
Pigeon 2	com.ctre.phoenix6.hardware.Pigeon2	CTRE (Phoenix V6)
CANCoder	com.ctre.phoenix6.hardware.CANCoder	CTRE (Phoenix V6)
Spark MAX	com.revrobotics.CANSparkMax	REV (REVLlib)
Spark Flex	com.revrobotics.CANSparkFlex	REV (REVLlib)
Through-Bore Encoder (RoboRIO, Absolute)	edu.wpi.first.wpilibj.DutyCycleEncoder	WPILib
Through-Bore Encoder (RoboRIO, Relative)	edu.wpi.first.wpilibj.Encoder	WPILib

Motor Basic Control (Talon SRX)

```
private WPI_TalonSRX motor;
@Override
public void robotInit() {
    motor = new WPI_TalonSRX (RobotMap.MOTOR_ID_IDENTIFIER);
    motor.configFactoryDefault(); // factory default
}
@Override
public void teleopInit() {
    motor.set(0.5);
}
@Override
public void teleopPeriodic() {
}
@Override
public void teleopExit() {
    motor.stopMotor()
}
```

Motor Basic Control (Spark MAX)

```
private CANSparkMax motor;
@Override
public void robotInit() {
    motor = new CANSparkMax (RobotMap.MOTOR_IDENTIFIER, CANSparkMaxLowLevelMotorType.kBrushless);
    motor.restoreFactoryDefaults(); // factory default
}
@Override
```



By **turback**

cheatography.com/turback/

Not published yet.

Last updated 18th December, 2024.

Page 1 of 4.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Motor Basic Control (Spark MAX) (cont)

```
> public void teleopInit() {
    motor.set(0.5);
}
@Override
public void teleopPeriodic() {
}
@Override
public void teleopExit() {
    motor.stopMotor()
}
```

Motor Basic Control (Talon FX)

```
private TalonFX motor;
private DutyCycleOut dutyCycleControl;
private NeutralOut neutralControl;
@Override
public void robotInit() {
    motor = new TalonFX(RobotMap.MOTOR_IDENTIFIER);
    motor.getConfigurator().apply(new TalonFXConfiguration()); // factory default
    dutyCycleControl = new DutyCycleOut(0);
    neutralControl = new NeutralOut();
}
@Override
public void teleopInit() {
    motor.setControl(DutyCycleControl.withOutput(0.5));
}
@Override
public void teleopPeriodic() {
}
@Override
public void teleopExit() {
    motor.setControl(neutralControl);
}
```

Encoder Access (Talon SRX, SRX Encoder)

```
private static final int PPR = 4096;
private static final double MOTOR_TO_MECHANISM_GEAR_RATIO = 8.0 / 1.0; // driver : driven
private WPI_TalonSRX motor;
@Override
public void robotInit() {
    motor = new WPI_TalonSRX(RobotMap.MOTOR_IDENTIFIER);
    motor.configureFactoryDefault(); // factory default
}
@Override
```



Encoder Access (Talon SRX, SRX Encoder) (cont)

```
> public void teleopPeriodic() {
    double counts = motor.getSelectedSensorPosition();
    double mechanismRotations = counts / PPR / MOTOR_TO_MECHANISM_GEAR_RATIO;
    double countsPer100ms = motor.getSelectedSensorVelocity();
    double mechanismRpm = countsPer100ms / PPR * 600.0 / MOTOR_TO_MECHANISM_GEAR_RATIO;
}
```

Encoder Access (Spark Max, NEO Integrated Encoder)

```
private static final double MOTOR_TO_MECHANISM_GEAR_RATIO = 8.0 / 1.0; // driver : driven
private CANSparkMax motor;
private RelativeEncoder encoder;
@Override
public void robotInit() {
    motor = new CANSparkMax( RobotMap.MOTOR_ID, CANSparkMax.LowLevelMotorType.kBrushless);
    motor.restoreFactoryDefaults(); // factory default
    encoder = motor.getEncoder();
}
@Override
public void teleopPeriodic() {
    double motorRotations = encoder.getPosition();
    double mechanismRotations = motorRotations / MOTOR_TO_MECHANISM_GEAR_RATIO;
    double motorRpm = encoder.getVelocity();
    double mechanismRpm = motorRpm / MOTOR_TO_MECHANISM_GEAR_RATIO;
}
```

Encoder Access (Spark Max, NEO Integrated Encoder)

```
private static final double MOTOR_TO_MECHANISM_GEAR_RATIO = 8.0 / 1.0; // driver : driven
private CANSparkMax motor;
private RelativeEncoder encoder;
@Override
public void robotInit() {
    motor = new CANSparkMax( RobotMap.MOTOR_ID, CANSparkMax.LowLevelMotorType.kBrushless);
    motor.restoreFactoryDefaults(); // factory default
    encoder = motor.getEncoder();
    encoder.setPositionFactor(1 / MOTOR_TO_MECHANISM_GEAR_RATIO);
    encoder.setVelocityFactor(1 / MOTOR_TO_MECHANISM_GEAR_RATIO);
}
@Override
public void teleopPeriodic() {
    double mechanismRotations = encoder.getPosition();
    double mechanismRpm = encoder.getVelocity();
}
```

Encoder Access (Talon FX, Integrated Encoder)

```
private static final double MOTOR_TO_MECHANISM_GEAR_RATIO = 8.0 / 1.0; // driver : driven
private TalonFX motor;
private Status Signal <Doable> positionSignal;
private Status Signal <Doable> velocitySignal;
@Override
public void robotInit() {
    motor = new TalonFX(RobotMap.MOTOR_ID);
    TalonFXConfiguration configuration = new TalonFXConfiguration();
    configuration.FeedbackKf = FeedbackConstants.kSensors;
    configuration.FeedbackKs = FeedbackConstants.kSources;
    configuration.FeedbackKt = FeedbackConstants.kTorque;
    configuration.FeedbackKv = FeedbackConstants.kVelocity;
    configuration.FeedbackKp = FeedbackConstants.kPosition;
    configuration.FeedbackKi = FeedbackConstants.kIntegral;
    configuration.FeedbackKd = FeedbackConstants.kDerivative;
    configuration.FeedbackKf = FeedbackConstants.kSensors;
    configuration.FeedbackKs = FeedbackConstants.kSources;
    configuration.FeedbackKt = FeedbackConstants.kTorque;
    configuration.FeedbackKv = FeedbackConstants.kVelocity;
    configuration.FeedbackKp = FeedbackConstants.kPosition;
    configuration.FeedbackKi = FeedbackConstants.kIntegral;
    configuration.FeedbackKd = FeedbackConstants.kDerivative;
    positionSignal = motor.getPosition();
    velocitySignal = motor.getVelocity();
}
@Override
public void teleopPeriodic() {
    BaseStatusSignal.refreshAll(positionSignal, velocitySignal); // only once per 20ms
    // loop, typically in periodic
    double mechanismRotations = positionSignal.getValue();
    double mechanismRotationsPerSecond = velocitySignal.getValue();
}
```

Encoder Access (CANCoder, Absolute)

```
private CANCoder encoder;
private Status Signal <Doable> absolutePositionSignal;
@Override
public void robotInit() {
    encoder = new CANCoder(RobotMap.CANCODER_ID);
    CANCoderConfiguration configuration = new CANCoderConfiguration();
    configuration.MagnetSensor.AbsoluteSensorRange = AbsoluteSensorRange.Full;
    configuration.MagnetSensor.Tolerance = 0.001;
    configuration.MagnetSensor.AbsolutePosition = 0;
    configuration.MagnetSensor.AbsolutePositionPerRotation = 360;
    configuration.MagnetSensor.AbsolutePositionPerRotation = 360;
    absolutePositionSignal = encoder.getPosition();
}
@Override
public void teleopPeriodic() {
    BaseStatusSignal.refreshAll(absolutePositionSignal); // only once per 20ms loop,
    // typically in periodic
    // typically absolute encoders are placed on the shaft after a gearbox, so gearbox ratio handling
    // isn't needed here.
    double mechanismRotations = absolutePositionSignal.getValue(); // [0, 1] -> [0,
    360]
}
```