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)
CAN Coder	com.ctre.phoenix6.hardware.CANCoder	CTRE (Phoenix V6)
Spark MAX	com.revrobotics.CANSparkMax	REV (REVLib)
Spark Flex	com.revrobotics.CANSparkFlex	REV (REVLib)
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 robotI nit() {
        motor = new WPI_Ta lon SRX (Ro bot Map.MO TOR _ID ENT IFIER);
        mot or.c on fig Fac tor yDe fau lt(); // factory default
}
@Override
public void teleop Init() {
        mot or.s et (0.5);
}
@Override
public void teleop Per iodic() {
}
@Override
public void teleop Exit() {
        mot or.s to pMo tor()
}
```

Motor Basic Control (Spark MAX)

```
private CANSparkMax motor;
@Override
public void robotI nit() {
        motor = new CANSpa rkM ax( Rob otM ap.M OT OR_ IDE NTI FIER, CANSpa rkL owL eve l.M oto rTy pe.k -
Br ush less);
        mot or.r es tor eFa cto ryD efa ults(); // factory default
}
@Override
```

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 DutyCy cleOut dutyCy cle Con trol;
private NeutralOut neutra lCo ntrol;
@Override
public void robotI nit() {
        motor = new TalonF X(R obo tMa p.M OTO R_I DEN TIF IER);
        mot or.g et Con fig ura tor ().a pp ly(new TalonF XCo nfi gur ati on()); // factory default
        dut yCy cle Control = new DutyCy cle Out(0);
        neu tra lCo ntrol = new Neutra lOut();
}
@Override
public void teleop Init() {
        mot or.s et Con tro l(d uty Cyc leC ont rol.wi thO utp ut( 0.5));
}
@Override
public void teleop Per iodic() {
}
@Override
public void teleop Exit() {
        mot or.s et Con tro l(n eut ral Con trol);
}
```

Encoder Access (Talon SRX, SRX Encoder)

```
private static final int PPR = 4096;
private static final double MOTOR_ TO_ MEC HAN ISM _GE AR_ RATIO = 8.0 / 1.0; // driver : driven
private WPI_Ta lonSRX motor;
@Override
public void robotI nit() {
        motor = new WPI_Ta lon SRX (Ro bot Map.MO TOR _ID ENT IFIER);
        mot or.c on fig Fac tor yDe fau lt(); // 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 CANSpa rkMax motor;
private Relati veE ncoder encoder;
@Override
public void robotI nit() {
        motor = new CANSpa rkM ax( Rob otM ap.M OT OR_ IDE NTI FIER, CANSpa rkL owL eve l.M oto rTy pe.k -
Br ush less);
        mot or.r es tor eFa cto ryD efa ults(); // factory default
        encoder = motor.g et Enc oder();
}
@Override
public void teleop Per iodic() {
        double motorR ota tions = encode r.g etP osi tion();
        double mechan ism Rot ations = motorR ota tions / MOTOR_ TO_ MEC HAN ISM _GE AR_ RATIO;
        double motorRpm = encode r.g etV elo city();
        double mechan ismRpm = motorRpm / MOTOR_ TO_ MEC HAN ISM _GE AR_ RATIO;
}
```

Encoder Access (Spark Max, NEO Integrated Encoder)

```
private static final double MOTOR_TO_MECHANISM_GEAR_RATIO = 8.0 / 1.0; // driver : driven
private CANSpa rkMax motor;
private Relati veE ncoder encoder;
@Override
public void robotI nit() {
        motor = new CANSpa rkM ax( Rob otM ap.M OT OR_ IDE NTI FIER, CANSpa rkL owL eve l.M oto rTy pe.k -
Br ush less);
        mot or.r es tor eFa cto ryD efa ults(); // factory default
        encoder = motor.g et Enc oder();
        enc ode r.s etP osi tio nCo nve rsi onF actor(1 / MOTOR_ TO_ MEC HAN ISM _GE AR_ RATIO);
        enc ode r.s etV elo cit yCo nve rsi onF actor(1 / MOTOR_ TO_ MEC HAN ISM _GE AR_ RATIO);
}
@Override
public void teleop Per iodic() {
        double mechan ism Rot ations = encode r.g etP osi tion();
        double mechan ismRpm = encode r.g etV elo city();
}
```

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 Sig nal <Do ubl e> positi onS ignal;
private Status Sig nal <Do ubl e> veloci tyS ignal;
@Override
public void robotI nit() {
        motor = new TalonF X(R obo tMa p.M OTO R_I DEN TIF IER);
        Tal onF XCo nfi gur ation config uration = new TalonF XCo nfi gur ati on();
        con fig ura tio n.F eed bac k.F eed bac kSe nso rSource = Feedba ckS ens orS our ceV alu e.R oto -
rSe nsor;
        con fig ura tio n.F eed bac k.S ens orT oMe cha nis mRatio = MOTOR_ TO_ MEC HAN ISM _GE AR_ RATIO;
        mot or.g et Con fig ura tor ().a pp ly( con fig ura tion);
        pos iti onS ignal = motor.g et Pos iti on();
        vel oci tyS ignal = motor.g et Vel oci ty();
}
@Override
public void teleop Per iodic() {
        Bas eSt atu sSi gna l.r efr esh All (po sit ion Signal, veloci tyS ignal); // only once per 20ms
loop, typically in periodic
        double mechan ism Rot ations = positi onS Ign al.g et Val ueA sDo uble();
        double mechan ism Rot ati ons Per Second = veloci tyS ign al.g et Val ueA sDo uble();
}
```

Encoder Access (CANCoder, Absolute)

```
private CANCoder encoder;
private Status Sig nal <Do ubl e> absPos iti onS ignal;
@Override
public void robotI nit() {
        encoder = new CANCod er( Rob otM ap.E NC ODER);
        CAN cod erC onf igu ration config uration = new CANcod erC onf igu rat ion();
        con fig ura tio n.M agn etS ens or.A bs olu teS ens orRange = Absolu teS ens orR ang eVa lue.Un -
sig ned _0To1;
        enc ode r.g etC onf igu rat or( ).a ppl y(c onf igu rat ion);
        abs Pos iti onS ignal = motor.g et Abs olu teP osi tion();
}
@Override
public void teleop Per iodic() {
        Bas eSt atu sSi gna l.r efr esh All (ab sPo sit ion Sig nal); // 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 mechan ism Rot ati onAbs = absPos iti onS ign al.g et Val ueA sDo uble(); // [0, 1] -> [0,
360]
}
```