SPARK MAX - Java Documentation

Generated by Doxygen 1.8.15

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 com.revrobotics.CANPIDController.AccelStrategy Enum Reference

Public Member Functions

- **AccelStrategy** (int value)

Static Public Member Functions

- static AccelStrategy **fromInt** (int value)

Public Attributes

- **kTrapezoidal** =(0)
- **kSCurve** =(1)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANPIDController.java

3.2 com.revrobotics.CANAnalog.AnalogMode Enum Reference

Public Member Functions

- **AnalogMode** (int value)

Static Public Member Functions

- static AnalogMode **fromId** (int id)

Public Attributes

- **kAbsolute** =(0)
- **kRelative** =(1)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANAnalog.java

3.3 com.revrobotics.CANPIDController.ArbFFUnits Enum Reference

Public Member Functions

- **ArbFFUnits** (int value)

Public Attributes

- **kVoltage** =(0)
- **kPercentOut** =(1)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANPIDController.java

3.4 com.revrobotics.CANAnalog Class Reference

Inherits com.revrobotics.CANSensor.

Classes

- enum AnalogMode

Public Member Functions

- CANAnalog (CANSparkMax device, AnalogMode mode)
- double getVoltage ()
- double getPosition ()
- double getVelocity ()
- CANError setPositionConversionFactor (double factor)
- CANError setVelocityConversionFactor (double factor)
- double getPositionConversionFactor ()
- double getVelocityConversionFactor ()
- CANError **setInverted** (boolean inverted)
- boolean **getInverted** ()

Protected Member Functions

- int **getID** ()

3.4.1 Constructor & Destructor Documentation

3.4.1.1 CANAnalog()

```
com.revrobotics.CANAnalog.CANAnalog (
            CANSparkMax device,
            AnalogMode mode )
```

Constructs a CANAnalog.

Parameters

device	The Spark Max to which the analog sensor is attached.
mode	The mode of the analog sensor, either absolute or relative

3.4.2 Member Function Documentation

3.4.2.1 getPosition()

```
double com.revrobotics.CANAnalog.getPosition ( )
```

Get the position of the motor. Returns value in the native unit of 'volt' by default, and can be changed by a scale factor using setPositionConversionFactor().

Returns

Position of the sensor in volts

3.4.2.2 getPositionConversionFactor()

```
double com.revrobotics.CANAnalog.getPositionConversionFactor ( )
```

Get the current conversion factor for the position of the analog sensor.

Returns

Analog position conversion factor

3.4.2.3 getVelocity()

```
double com.revrobotics.CANAnalog.getVelocity ( )
```

Get the velocity of the motor. Returns value in the native units of 'volts per second' by default, and can be changed by a scale factor using setVelocityConversionFactor().

Returns

Velocity of the sensor in volts per second

3.4.2.4 getVelocityConversionFactor()

```
double com.revrobotics.CANAnalog.getVelocityConversionFactor ( )
```

Get the current conversion factor for the velocity of the analog sensor.

Returns

Analog velocity conversion factor

3.4.2.5 getVoltage()

```
double com.revrobotics.CANAnalog.getVoltage ( )
```

Get the voltage of the analog sensor.

Returns

Voltage of the sensor.

3.4.2.6 setPositionConversionFactor()

```
CANError com.revrobotics.CANAnalog.setPositionConversionFactor (
            double factor )
```

Set the conversion factor for the position of the analog sensor. By default, revolutions per volt is 1. Changing the position conversion factor will also change the position units.

Parameters

factor	The conversion factor which will be multiplied by volts

Returns

CANError Set to CANError.kOK if successful

3.4.2.7 setVelocityConversionFactor()

```
CANError com.revrobotics.CANAnalog.setVelocityConversionFactor (
            double factor )
```

Set the conversion factor for the veolocity of the analog sensor. By default, revolutions per volt second is 1. Changing the velocity conversion factor will also change the velocity units.

Parameters

factor	The conversion factor which will be multipled by volts per second

Returns

CANError Set to CANError.kOK is successful

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANAnalog.java

3.5 com.revrobotics.CANDigitalInput Class Reference

Classes

- enum LimitSwitch
- enum LimitSwitchPolarity

Public Member Functions

- CANDigitalInput (CANSparkMax device, LimitSwitch limitSwitch, LimitSwitchPolarity polarity)
- boolean get ()
- CANError enableLimitSwitch (boolean enable)
- boolean isLimitSwitchEnabled ()

3.5.1 Constructor & Destructor Documentation

3.5.1.1 CANDigitalInput()

```
com.revrobotics.CANDigitalInput.CANDigitalInput (
            CANSparkMax device,
            LimitSwitch limitSwitch,
            LimitSwitchPolarity polarity )
```

Constructs a CANDigitalInput.

Parameters

device	The Spark Max to which the limit switch is attached.
limitSwitch	Whether this is forward or reverse limit switch.
polarity	Whether the limit switch is normally open or normally closed.

3.5.2 Member Function Documentation

3.5.2.1 enableLimitSwitch()

```
CANError com.revrobotics.CANDigitalInput.enableLimitSwitch (
            boolean enable )
```

Enables or disables controller shutdown based on limit switch.

Parameters

enable	Enable/disable motor shutdown based on limit switch state. This does not effect the result of the get() command.

Returns

CANError Set to CANError::kOk if successful

3.5.2.2 get()

```
boolean com.revrobotics.CANDigitalInput.get ( )
```

Get the value from a digital input channel.

Retrieve the value of a single digital input channel from a motor controller. This method will return the state of the limit input based on the selected polarity, whether or not it is enabled.

Returns

The state of the limit switch based on the configured polarity

3.5.2.3 isLimitSwitchEnabled()

```
boolean com.revrobotics.CANDigitalInput.isLimitSwitchEnabled ( )
```

Returns

True if limit switch is enabled

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANDigitalInput.java

3.6 com.revrobotics.CANEncoder Class Reference

Inherits com.revrobotics.CANSensor.

Public Member Functions

- CANEncoder (CANSparkMax device, SensorType sensorType, int cpr)
- CANEncoder (CANSparkMax device)
- double getPosition ()
- double getVelocity ()
- CANError setPosition (double position)
- CANError setPositionConversionFactor (double factor)
- CANError setVelocityConversionFactor (double factor)
- double getPositionConversionFactor ()
- double getVelocityConversionFactor ()
- CANError setAverageDepth (int depth)
- int getAverageDepth ()
- CANError setMeasurementPeriod (int period_us)
- int getMeasurementPeriod ()
- int getCPR ()
- CANError **setInverted** (boolean inverted)
- boolean **getInverted** ()

Protected Member Functions

- int **getID** ()

3.6.1 Constructor & Destructor Documentation

3.6.1.1 CANEncoder() [1/2]

```
com.revrobotics.CANEncoder.CANEncoder (
            CANSparkMax device,
            SensorType sensorType,
            int cpr )
```

Constructs a CANPIDController.

Parameters

device	The Spark Max to which the encoder is attached.
sensorType	The encoder type for the motor: kHallEffect or kQuadrature
cpr	The counts per revolution of the encoder

3.6.1.2 CANEncoder() [2/2]

```
com.revrobotics.CANEncoder.CANEncoder (
            CANSparkMax device )
```

Constructs a CANPIDController.

Parameters

device	The Spark Max to which the encoder is attached.

3.6.2 Member Function Documentation

3.6.2.1 getAverageDepth()

```
int com.revrobotics.CANEncoder.getAverageDepth ( )
```

Get the averafe sampling depth for a quadrature encoder.

Returns

The average sampling depth

3.6.2.2 getCPR()

```
int com.revrobotics.CANEncoder.getCPR ( )
```

Get the counts per revolution of the quadrature encoder.

Returns

Counts per revolution

3.6.2.3 getMeasurementPeriod()

```
int com.revrobotics.CANEncoder.getMeasurementPeriod ( )
```

Get the number of samples for reading from a quadrature encoder.

Returns

Number of samples

3.6.2.4 getPosition()

```
double com.revrobotics.CANEncoder.getPosition ( )
```

Get the position of the motor. This returns the native units of 'rotations' by default, and can be changed by a scale factor using setPositionConversionFactor().

Returns

Number of rotations of the motor

3.6.2.5 getPositionConversionFactor()

```
double com.revrobotics.CANEncoder.getPositionConversionFactor ( )
```

Get the conversion factor for position of the encoder. Multiplied by the native output units to give you position

Returns

The conversion factor for position

3.6.2.6 getVelocity()

```
double com.revrobotics.CANEncoder.getVelocity ( )
```

Get the velocity of the motor. This returns the native units of 'RPM' by default, and can be changed by a scale factor using setVelocityConversionFactor().

Returns

Number the RPM of the motor

3.6.2.7 getVelocityConversionFactor()

```
double com.revrobotics.CANEncoder.getVelocityConversionFactor ( )
```

Get the conversion factor for velocity of the encoder. Multiplied by the native output units to give you velocity

Returns

The conversion factor for velocity

3.6.2.8 setAverageDepth()

```
CANError com.revrobotics.CANEncoder.setAverageDepth (
            int depth )
```

Set the average sampling depth for a quadrature encoder. This value sets the number of samples in the average for velocity readings. This can be any value from 1 to 64.

When the SparkMax controller is in Brushless mode, this will not change any behavior.

Parameters

depth	The average sampling depth between 1 and 64 (default)

Returns

CANError.kOK if successful

3.6.2.9 setMeasurementPeriod()

```
CANError com.revrobotics.CANEncoder.setMeasurementPeriod (
            int period_us )
```

Set the measurement period for velocity measurements of a quadrature encoder. When the SparkMax controller is in Brushless mode, this will not change any behavior.

The basic formula to calculate velocity is change in positon / change in time. This parameter sets the change in time for measurement.

Parameters

period_us	Measurement period in milliseconds. This number may be between 1 and 100 (default).

Returns

CANError.kOK if successful

3.6.2.10 setPosition()

```
CANError com.revrobotics.CANEncoder.setPosition (
            double position )
```

Set the position of the encoder. By default the units are 'rotations' and can be changed by a scale factor using setPositionConversionFactor().

Parameters

position	Number of rotations of the motor

Returns

CANError Set to CANError.kOK if successful

3.6.2.11 setPositionConversionFactor()

```
CANError com.revrobotics.CANEncoder.setPositionConversionFactor (
            double factor )
```

Set the conversion factor for position of the encoder. Multiplied by the native output units to give you position.

Parameters

factor	The conversion factor to multiply the native units by

Returns

CANError Set to CANError.kOK if successful

3.6.2.12 setVelocityConversionFactor()

```
CANError com.revrobotics.CANEncoder.setVelocityConversionFactor (
            double factor )
```

Set the conversion factor for velocity of the encoder. Multiplied by the native output units to give you velocity

Parameters

factor	The conversion factor to multiply the native units by

Returns

CANError Set to CANError.kOK if successful

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANEncoder.java

3.7 com.revrobotics.CANError Enum Reference

Public Member Functions

- **CANError** (int value)

Static Public Member Functions

- static CANError **fromInt** (int id)

Public Attributes

- **kOk** =(0)
- **kError** =(1)
- **kTimeout** =(2)
- **kNotImplmented** =(3)
- **kHALError** =(4)
- **kCantFindFirmware** =(5)
- **kFirmwareTooOld** =(6)
- **kFirmwareTooNew** =(7)
- **kParamInvalidID** =(8)
- **kParamMismatchType** =(9)
- **kParamAccessMode** =(10)
- **kParamInvalid** =(11)
- **kParamNotImplementedDeprecated** =(12)
- **kFollowConfigMismatch** =(13)
- **kInvalid** =(14)
- **kSetpointOutOfRange** =(15)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANError.java

3.8 com.revrobotics.CANPIDController Class Reference

Classes

- enum AccelStrategy
- enum ArbFFUnits

Public Member Functions

- CANPIDController (CANSparkMax device)
- CANError setReference (double value, ControlType ctrl)
- CANError setReference (double value, ControlType ctrl, int pidSlot)
- CANError setReference (double value, ControlType ctrl, int pidSlot, double arbFeedforward)
- CANError setReference (double value, ControlType ctrl, int pidSlot, double arbFeedforward, ArbFFUnits arb↩ FFUnits)
- CANError setP (double gain)
- CANError setP (double gain, int slotID)
- CANError setI (double gain)
- CANError setI (double gain, int slotID)
- CANError setD (double gain)
- CANError setD (double gain, int slotID)
- CANError setDFilter (double gain)
- CANError setDFilter (double gain, int slotID)
- CANError setFF (double gain)
- CANError setFF (double gain, int slotID)
- CANError setIZone (double IZone)
- CANError setIZone (double IZone, int slotID)
- CANError setOutputRange (double min, double max)
- CANError setOutputRange (double min, double max, int slotID)
- double getP ()
- double getP (int slotID)
- double getI ()
- double getI (int slotID)
- double getD ()
- double getD (int slotID)
- double getDFilter (int slotID)
- double getFF ()
- double getFF (int slotID)
- double getIZone ()
- double getIZone (int slotID)
- double getOutputMin ()
- double getOutputMin (int slotID)
- double getOutputMax ()
- double getOutputMax (int slotID)
- CANError setSmartMotionMaxVelocity (double maxVel, int slotID)
- CANError setSmartMotionMaxAccel (double maxAccel, int slotID)
- CANError setSmartMotionMinOutputVelocity (double minVel, int slotID)
- CANError setSmartMotionAllowedClosedLoopError (double allowedErr, int slotID)
- CANError setSmartMotionAccelStrategy (AccelStrategy accelStrategy, int slotID)
- double getSmartMotionMaxVelocity (int slotID)
- double getSmartMotionMaxAccel (int slotID)
- double getSmartMotionMinOutputVelocity (int slotID)
- double getSmartMotionAllowedClosedLoopError (int slotID)

- AccelStrategy getSmartMotionAccelStrategy (int slotID)
- CANError setIMaxAccum (double iMaxAccum, int slotID)
- double getIMaxAccum (int slotID)
- CANError setIAccum (double iAccum)
- double getIAccum ()
- CANError setFeedbackDevice (final CANSensor sensor)

3.8.1 Constructor & Destructor Documentation

3.8.1.1 CANPIDController()

```
com.revrobotics.CANPIDController.CANPIDController (
            CANSparkMax device )
```

Constructs a CANPIDController.

Parameters

device	The Spark Max this object configures.

3.8.2 Member Function Documentation

3.8.2.1 getD() [1/2]

```
double com.revrobotics.CANPIDController.getD ( )
```

Get the Derivative Gain constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Returns

double D Gain value

3.8.2.2 getD() [2/2]

```
double com.revrobotics.CANPIDController.getD (
            int slotID )
```

Get the Derivative Gain constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

double D Gain value

3.8.2.3 getDFilter()

```
double com.revrobotics.CANPIDController.getDFilter (
            int slotID )
```

Get the Derivative Filter constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

double D Filter value

3.8.2.4 getFF() [1/2]

```
double com.revrobotics.CANPIDController.getFF ( )
```

Get the Feed-forward Gain constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Returns

double F Gain value

3.8.2.5 getFF() [2/2]

```
double com.revrobotics.CANPIDController.getFF (
            int slotID )
```

Get the Feed-forward Gain constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

double F Gain value

3.8.2.6 getI() [1/2]

```
double com.revrobotics.CANPIDController.getI ( )
```

Get the Integral Gain constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Returns

double I Gain value

3.8.2.7 getI() [2/2]

```
double com.revrobotics.CANPIDController.getI (
            int slotID )
```

Get the Integral Gain constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

double I Gain value

3.8.2.8 getIAccum()

```
double com.revrobotics.CANPIDController.getIAccum ( )
```

Get the I accumulator of the PID controller. This is useful when wishing to see what the I accumulator value is to help with PID tuning

Returns

The value of the I accumulator

3.8.2.9 getIMaxAccum()

```
double com.revrobotics.CANPIDController.getIMaxAccum (
            int slotID )
```

Get the maximum I accumulator of the PID controller. This value is used to constrain the I accumulator to help manage integral wind-up

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

The max value to contrain the I accumulator to

3.8.2.10 getIZone() [1/2]

```
double com.revrobotics.CANPIDController.getIZone ( )
```

Get the IZone constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Returns

double IZone value

3.8.2.11 getIZone() [2/2]

```
double com.revrobotics.CANPIDController.getIZone (
            int slotID )
```

Get the IZone constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

double IZone value

3.8.2.12 getOutputMax() [1/2]

```
double com.revrobotics.CANPIDController.getOutputMax ( )
```

Get the max output of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Returns

double max value

3.8.2.13 getOutputMax() [2/2]

```
double com.revrobotics.CANPIDController.getOutputMax (
            int slotID )
```

Get the max output of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

double max value

3.8.2.14 getOutputMin() [1/2]

```
double com.revrobotics.CANPIDController.getOutputMin ( )
```

Get the derivative filter constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

double D FilterGet the min output of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Returns

double min value

3.8.2.15 getOutputMin() [2/2]

```
double com.revrobotics.CANPIDController.getOutputMin (
            int slotID )
```

Get the min output of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

double min value

3.8.2.16 getP() [1/2]

```
double com.revrobotics.CANPIDController.getP ( )
```

Get the Proportional Gain constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Returns

double P Gain value

3.8.2.17 getP() [2/2]

```
double com.revrobotics.CANPIDController.getP (
            int slotID )
```

Get the Proportional Gain constant of the PIDF controller on the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

double P Gain value

3.8.2.18 getSmartMotionAccelStrategy()

```
AccelStrategy com.revrobotics.CANPIDController.getSmartMotionAccelStrategy (
            int slotID )
```

Get the acceleration strategy used to control acceleration on the motor.The current strategy is trapezoidal motion profiling.

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

The acceleration strategy to use for the automatically generated motion profile.

3.8.2.19 getSmartMotionAllowedClosedLoopError()

```
double com.revrobotics.CANPIDController.getSmartMotionAllowedClosedLoopError (
            int slotID )
```

Get the allowed closed loop error of SmartMotion mode. This value is how much deviation from your setpoint is tolerated and is useful in preventing oscillation around your setpoint.

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

The allowed deviation for your setpoint vs actual position in rotations

3.8.2.20 getSmartMotionMaxAccel()

```
double com.revrobotics.CANPIDController.getSmartMotionMaxAccel (
            int slotID )
```

Get the maximum acceleration of the SmartMotion mode. This is the accleration that the motor velocity will increase at until the max velocity is reached

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

The maxmimum acceleration for the motion profile in RPM per second

3.8.2.21 getSmartMotionMaxVelocity()

```
double com.revrobotics.CANPIDController.getSmartMotionMaxVelocity (
            int slotID )
```

Get the maximum velocity of the SmartMotion mode. This is the velocity that is reached in the middle of the profile and is what the motor should spend most of its time at

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

The maxmimum cruise velocity for the motion profile in RPM

3.8.2.22 getSmartMotionMinOutputVelocity()

```
double com.revrobotics.CANPIDController.getSmartMotionMinOutputVelocity (
            int slotID )
```

Get the mimimum velocity of the SmartMotion mode. Any requested velocities below this value will be set to 0.

Parameters

slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

The minimum velocity for the motion profile in RPM

3.8.2.23 setD() [1/2]

```
CANError com.revrobotics.CANPIDController.setD (
            double gain )
```

Set the Derivative Gain constant of the PIDF controller on the SPARK MAX. This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burnFlash() is called. The recommended method to configure this parameter is use to SPARK MAX GUI to tune and save parameters.

Parameters

gain	The derivative gain value, must be positive

Returns

CANError Set to REV_OK if successful

3.8.2.24 setD() [2/2]

```
CANError com.revrobotics.CANPIDController.setD (
            double gain,
            int slotID )
```

Set the Derivative Gain constant of the PIDF controller on the SPARK MAX. This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burnFlash() is called. The recommended method to configure this parameter is use to SPARK MAX GUI to tune and save parameters.

Parameters

gain	The derivative gain value, must be positive
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to REV_OK if successful

3.8.2.25 setDFilter() [1/2]

```
CANError com.revrobotics.CANPIDController.setDFilter (
            double gain )
```

Set the Derivative Filter constant of the PIDF controller on the SPARK MAX. This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burnFlash() is called.

Parameters

gain	The derivative filter value, must be a positive number between 0 and 1

Returns

CANError Set to REV_OK if successful

3.8.2.26 setDFilter() [2/2]

```
CANError com.revrobotics.CANPIDController.setDFilter (
            double gain,
            int slotID )
```

Set the Derivative Filter constant of the PIDF controller on the SPARK MAX. This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burnFlash() is called.

Parameters

gain	The derivative filter value, must be a positive number between 0 and 1
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to REV_OK if successful

3.8.2.27 setFeedbackDevice()

```
CANError com.revrobotics.CANPIDController.setFeedbackDevice (
            final CANSensor sensor )
```

Set the controller's feedback device.

The default feedback device in brushless mode is assumed to be the integrated encoder and the default feedback device in brushed mode is assumed to be a quadrature encoder. This is used to changed to another feedback device for the controller, such as an analog sensor.

If there is a limited range on the feedback sensor that should be observed by the PIDController, it can be set by calling SetFeedbackSensorRange() on the sensor object.

Parameters

sensor	The sensor to use as a feedback device

Returns

CANError set to kOK if successful

3.8.2.28 setFF() [1/2]

```
CANError com.revrobotics.CANPIDController.setFF (
            double gain )
```

Set the Feed-froward Gain constant of the PIDF controller on the SPARK MAX. This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burnFlash() is called. The recommended method to configure this parameter is use to SPARK MAX GUI to tune and save parameters.

Parameters

gain	The feed-forward gain value

Returns

CANError Set to REV_OK if successful

3.8.2.29 setFF() [2/2]

```
CANError com.revrobotics.CANPIDController.setFF (
            double gain,
            int slotID )
```

Set the Feed-froward Gain constant of the PIDF controller on the SPARK MAX. This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burnFlash() is called. The recommended method to configure this parameter is use to SPARK MAX GUI to tune and save parameters.

Parameters

gain	The feed-forward gain value
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to REV_OK if successful

3.8.2.30 setI() [1/2]

```
CANError com.revrobotics.CANPIDController.setI (
            double gain )
```

Set the Integral Gain constant of the PIDF controller on the SPARK MAX. This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burnFlash() is called. The recommended method to configure this parameter is use to SPARK MAX GUI to tune and save parameters.

Parameters

gain	The integral gain value, must be positive

Returns

CANError Set to REV_OK if successful

3.8.2.31 setI() [2/2]

```
CANError com.revrobotics.CANPIDController.setI (
            double gain,
            int slotID )
```

Set the Integral Gain constant of the PIDF controller on the SPARK MAX. This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burnFlash() is called. The recommended method to configure this parameter is use to SPARK MAX GUI to tune and save parameters.

Parameters

gain	The integral gain value, must be positive
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to REV_OK if successful

3.8.2.32 setIAccum()

```
CANError com.revrobotics.CANPIDController.setIAccum (
            double iAccum )
```

Set the I accumulator of the PID controller. This is useful when wishing to force a reset on the I accumulator of the PID controller. You can also preset values to see how it will respond to certain I characteristics

To use this function, the controller must be in a closed loop control mode by calling setReference()

Parameters

iAccum	The value to set the I accumulator to

Returns

CANError Set to kOK if successful

3.8.2.33 setIMaxAccum()

```
CANError com.revrobotics.CANPIDController.setIMaxAccum (
            double iMaxAccum,
            int slotID )
```

Configure the maximum I accumulator of the PID controller. This value is used to constrain the I accumulator to help manage integral wind-up

Parameters

iMaxAccum	The max value to contrain the I accumulator to
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to kOK if successful

3.8.2.34 setIZone() [1/2]

```
CANError com.revrobotics.CANPIDController.setIZone (
            double IZone )
```

Set the IZone range of the PIDF controller on the SPARK MAX. This value specifies the range the |error| must be within for the integral constant to take effect.

This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burn↩ Flash() is called. The recommended method to configure this parameter is to use the SPARK MAX GUI to tune and save parameters.

Parameters

IZone	The IZone value, must be positive. Set to 0 to disable

Returns

CANError Set to REV_OK if successful

3.8.2.35 setIZone() [2/2]

```
CANError com.revrobotics.CANPIDController.setIZone (
            double IZone,
            int slotID )
```

Set the IZone range of the PIDF controller on the SPARK MAX. This value specifies the range the |error| must be within for the integral constant to take effect.

This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burn↩ Flash() is called. The recommended method to configure this parameter is to use the SPARK MAX GUI to tune and save parameters.

Parameters

IZone	The IZone value, must be positive. Set to 0 to disable
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to REV_OK if successful

3.8.2.36 setOutputRange() [1/2]

```
CANError com.revrobotics.CANPIDController.setOutputRange (
            double min,
            double max )
```

Set the min amd max output for the closed loop mode.

This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burn↩ Flash() is called. The recommended method to configure this parameter is to use the SPARK MAX GUI to tune and save parameters.

Parameters

min	Reverse power minimum to allow the controller to output
max	Forward power maximum to allow the controller to output

Returns

CANError Set to REV_OK if successful

3.8.2.37 setOutputRange() [2/2]

```
CANError com.revrobotics.CANPIDController.setOutputRange (
            double min,
            double max,
            int slotID )
```

Set the min amd max output for the closed loop mode.

This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burn↩ Flash() is called. The recommended method to configure this parameter is to use the SPARK MAX GUI to tune and save parameters.

Parameters

min	Reverse power minimum to allow the controller to output
max	Forward power maximum to allow the controller to output
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to REV_OK if successful

3.8.2.38 setP() [1/2]

```
CANError com.revrobotics.CANPIDController.setP (
            double gain )
```

Set the Proportional Gain constant of the PIDF controller on the SPARK MAX. This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burnFlash() is called. The recommended method to configure this parameter is use to SPARK MAX GUI to tune and save parameters.

Parameters

gain	The proportional gain value, must be positive

Returns

CANError Set to REV_OK if successful

3.8.2.39 setP() [2/2]

```
CANError com.revrobotics.CANPIDController.setP (
            double gain,
            int slotID )
```

Set the Proportional Gain constant of the PIDF controller on the SPARK MAX. This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burnFlash() is called. The recommended method to configure this parameter is use to SPARK MAX GUI to tune and save parameters.

Parameters

gain	The proportional gain value, must be positive
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to REV_OK if successful

3.8.2.40 setReference() [1/4]

```
CANError com.revrobotics.CANPIDController.setReference (
            double value,
            ControlType ctrl )
```

Set the controller reference value based on the selected control mode.

Parameters

value	The value to set depending on the control mode. For basic duty cycle control this should be a value between -1 and 1 Otherwise: Voltage Control: Voltage (volts) Velocity Control: Velocity (RPM) Position Control: Position (Rotations) Current Control: Current (Amps). Native units can be changed using the setPositionConversionFactor() or setVelocityConversionFactor() methods of the CANEncoder class
ctrl	Is the control type

Returns

CANError Set to REV_OK if successful

3.8.2.41 setReference() [2/4]

```
CANError com.revrobotics.CANPIDController.setReference (
            double value,
            ControlType ctrl,
            int pidSlot )
```

Set the controller reference value based on the selected control mode. This will override the pre-programmed control mode but not change what is programmed to the controller.

Parameters

value	The value to set depending on the control mode. For basic duty cycle control this should be a value between -1 and 1 Otherwise: Voltage Control: Voltage (volts) Velocity Control: Velocity (RPM) Position Control: Position (Rotations) Current Control: Current (Amps). Native units can be changed using the setPositionConversionFactor() or setVelocityConversionFactor() methods of the CANEncoder class
ctrl	Is the control type to override with
pidSlot	for this command

Returns

CANError Set to REV_OK if successful

3.8.2.42 setReference() [3/4]

```
CANError com.revrobotics.CANPIDController.setReference (
            double value,
```

```
            ControlType ctrl,
            int pidSlot,
            double arbFeedforward )
```

Set the controller reference value based on the selected control mode. This will override the pre-programmed control mode but not change what is programmed to the controller.

Parameters

value	The value to set depending on the control mode. For basic duty cycle control this should be a value between -1 and 1 Otherwise: Voltage Control: Voltage (volts) Velocity Control: Velocity (RPM) Position Control: Position (Rotations) Current Control: Current (Amps). Native units can be changed using the setPositionConversionFactor() or setVelocityConversionFactor() methods of the CANEncoder class
ctrl	Is the control type to override with
pidSlot	for this command
arbFeedforward	A value from which is represented in voltage applied to the motor after the result of the specified control mode. The units for the parameter is Volts. This value is set after the control mode, but before any current limits or ramp rates.

Returns

CANError Set to REV_OK if successful

3.8.2.43 setReference() [4/4]

```
CANError com.revrobotics.CANPIDController.setReference (
            double value,
            ControlType ctrl,
            int pidSlot,
            double arbFeedforward,
            ArbFFUnits arbFFUnits )
```

Set the controller reference value based on the selected control mode. This will override the pre-programmed control mode but not change what is programmed to the controller.

Parameters

value	The value to set depending on the control mode. For basic duty cycle control this should be a value between -1 and 1 Otherwise: Voltage Control: Voltage (volts) Velocity Control: Velocity (RPM) Position Control: Position (Rotations) Current Control: Current (Amps). Native units can be changed using the setPositionConversionFactor() or setVelocityConversionFactor() methods of the CANEncoder class
ctrl	Is the control type to override with
pidSlot	for this command
arbFeedforward	A value from which is represented in voltage applied to the motor after the result of the specified control mode. The units for the parameter is Volts. This value is set after the control mode, but before any current limits or ramp rates.
arbFFUnits	The units the arbitrary feed forward term is in

Returns

CANError Set to REV_OK if successful

3.8.2.44 setSmartMotionAccelStrategy()

```
CANError com.revrobotics.CANPIDController.setSmartMotionAccelStrategy (
            AccelStrategy accelStrategy,
            int slotID )
```

Coming soon. Configure the acceleration strategy used to control acceleration on the motor. The current strategy is trapezoidal motion profiling.

Parameters

accelStrategy	The acceleration strategy to use for the automatically generated motion profile
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to kOK if successful

3.8.2.45 setSmartMotionAllowedClosedLoopError()

```
CANError com.revrobotics.CANPIDController.setSmartMotionAllowedClosedLoopError (
            double allowedErr,
            int slotID )
```

Configure the allowed closed loop error of SmartMotion mode. This value is how much deviation from your setpoint is tolerated and is useful in preventing oscillation around your setpoint.

Parameters

allowedErr	The allowed deviation for your setpoint vs actual position in rotations
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to kOK if successful

3.8.2.46 setSmartMotionMaxAccel()

```
CANError com.revrobotics.CANPIDController.setSmartMotionMaxAccel (
            double maxAccel,
            int slotID )
```

Configure the maximum acceleration of the SmartMotion mode. This is the accleration that the motor velocity will increase at until the max velocity is reached

Parameters

maxAccel	The maxmimum acceleration for the motion profile in RPM per second
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to kOK if successful

3.8.2.47 setSmartMotionMaxVelocity()

```
CANError com.revrobotics.CANPIDController.setSmartMotionMaxVelocity (
            double maxVel,
            int slotID )
```

Configure the maximum velocity of the SmartMotion mode. This is the velocity that is reached in the middle of the profile and is what the motor should spend most of its time at

Parameters

maxVel	The maxmimum cruise velocity for the motion profile in RPM
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to kOK if successful

3.8.2.48 setSmartMotionMinOutputVelocity()

```
CANError com.revrobotics.CANPIDController.setSmartMotionMinOutputVelocity (
            double minVel,
            int slotID )
```

Configure the mimimum velocity of the SmartMotion mode. Any requested velocities below this value will be set to 0.

Parameters

minVel	The minimum velocity for the motion profile in RPM
slotID	Is the gain schedule slot, the value is a number between 0 and 3. Each slot has its own set of gain values and can be changed in each control frame using SetReference().

Returns

CANError Set to kOK if successful

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANPIDController.java

3.9 com.revrobotics.CANSparkMax Class Reference

Inherits com.revrobotics.CANSparkMaxLowLevel, and AutoCloseable.

Classes

- class **ExternalFollower**
- enum FaultID
- enum IdleMode
- enum InputMode
- enum SoftLimitDirection

Public Member Functions

- CANSparkMax (int deviceID, MotorType type)
- void close ()
- void set (double speed)
- double get ()
- void setInverted (boolean isInverted)
- boolean getInverted ()
- void disable ()
- void **stopMotor** ()
- void **pidWrite** (double output)
- CANEncoder getEncoder ()
- CANEncoder getEncoder (SensorType sensorType, int cpr)
- CANAnalog getAnalog (AnalogMode mode)
- CANPIDController getPIDController ()
- CANDigitalInput getForwardLimitSwitch (CANDigitalInput.LimitSwitchPolarity polarity)
- CANDigitalInput getReverseLimitSwitch (CANDigitalInput.LimitSwitchPolarity polarity)
- CANError setSmartCurrentLimit (int limit)
- CANError setSmartCurrentLimit (int stallLimit, int freeLimit)
- CANError setSmartCurrentLimit (int stallLimit, int freeLimit, int limitRPM)
- CANError setSecondaryCurrentLimit (double limit)
- CANError setSecondaryCurrentLimit (double limit, int chopCycles)
- CANError setIdleMode (IdleMode mode)

- IdleMode getIdleMode ()
- CANError enableVoltageCompensation (double nominalVoltage)
- CANError disableVoltageCompensation ()
- double getVoltageCompensationNominalVoltage ()
- CANError setOpenLoopRampRate (double rate)
- CANError setClosedLoopRampRate (double rate)
- double getOpenLoopRampRate ()
- double getClosedLoopRampRate ()
- CANError follow (final CANSparkMax leader)
- CANError follow (final CANSparkMax leader, boolean invert)
- CANError follow (ExternalFollower leader, int deviceID)
- CANError follow (ExternalFollower leader, int deviceID, boolean invert)
- boolean isFollower ()
- short getFaults ()
- short getStickyFaults ()
- boolean getFault (FaultID faultID)
- boolean getStickyFault (FaultID faultID)
- double getBusVoltage ()
- double getAppliedOutput ()
- double getOutputCurrent ()
- double getMotorTemperature ()
- CANError clearFaults ()
- CANError burnFlash ()
- CANError setCANTimeout (int milliseconds)
- CANError enableSoftLimit (SoftLimitDirection direction, boolean enable)
- CANError setSoftLimit (SoftLimitDirection direction, float limit)
- double getSoftLimit (SoftLimitDirection direction)
- boolean isSoftLimitEnabled (SoftLimitDirection direction)
- CANError getLastError ()

Protected Member Functions

- int getFeedbackDeviceID ()

Additional Inherited Members

3.9.1 Constructor & Destructor Documentation

3.9.1.1 CANSparkMax()

```
com.revrobotics.CANSparkMax.CANSparkMax (
            int deviceID,
            MotorType type )
```

Create a new SPARK MAX Controller

Parameters

deviceID	The device ID.
type	The motor type connected to the controller. Brushless motors must be connected to their matching color and the hall sensor plugged in. Brushed motors must be connected to the Red and Black terminals only.

3.9.2 Member Function Documentation

3.9.2.1 burnFlash()

```
CANError com.revrobotics.CANSparkMax.burnFlash ( )
```

Writes all settings to flash.

Returns

CANError Set to CANError.kOK if successful

3.9.2.2 clearFaults()

```
CANError com.revrobotics.CANSparkMax.clearFaults ( )
```

Clears all sticky faults.

Returns

CANError Set to CANError.kOK if successful

3.9.2.3 close()

```
void com.revrobotics.CANSparkMax.close ( )
```

Closes the SPARK MAX Controller

3.9.2.4 disable()

```
void com.revrobotics.CANSparkMax.disable ( )
```

Common interface for disabling a motor.

3.9.2.5 disableVoltageCompensation()

```
CANError com.revrobotics.CANSparkMax.disableVoltageCompensation ( )
```

Disables the voltage compensation setting for all modes on the SPARK MAX.

Returns

CANError Set to CANError.kOK if successful

3.9.2.6 enableSoftLimit()

```
CANError com.revrobotics.CANSparkMax.enableSoftLimit (
            SoftLimitDirection direction,
            boolean enable )
```

Enable soft limits

Parameters

direction	the direction of motion to restrict
enable	set true to enable soft limits

Returns

CANError Set to CANError.kOK if successful

3.9.2.7 enableVoltageCompensation()

```
CANError com.revrobotics.CANSparkMax.enableVoltageCompensation (
            double nominalVoltage )
```

Sets the voltage compensation setting for all modes on the SPARK MAX and enables voltage compensation.

Parameters

nominalVoltage	Nominal voltage to compensate output to

Returns

CANError Set to CANError.kOK if successful

3.9.2.8 follow() [1/4]

```
CANError com.revrobotics.CANSparkMax.follow (
            final CANSparkMax leader )
```

Causes this controller's output to mirror the provided leader.

Only voltage output is mirrored. Settings changed on the leader do not affect the follower.

The motor will spin in the same direction as the leader. This can be changed by passing a true constant after the leader parameter.

Following anything other than a CAN SPARK MAX is not officially supported.

Parameters

leader	The motor controller to follow.

Returns

CANError Set to CANError.kOK if successful

3.9.2.9 follow() [2/4]

```
CANError com.revrobotics.CANSparkMax.follow (
            final CANSparkMax leader,
            boolean invert )
```

Causes this controller's output to mirror the provided leader.

Only voltage output is mirrored. Settings changed on the leader do not affect the follower.

Following anything other than a CAN SPARK MAX is not officially supported.

Parameters

leader	The motor controller to follow.
invert	Set the follower to output opposite of the leader

Returns

CANError Set to CANError.kOK if successful

3.9.2.10 follow() [3/4]

```
CANError com.revrobotics.CANSparkMax.follow (
            ExternalFollower leader,
            int deviceID )
```

Causes this controller's output to mirror the provided leader.

Only voltage output is mirrored. Settings changed on the leader do not affect the follower.

The motor will spin in the same direction as the leader. This can be changed by passing a true constant after the deviceID parameter.

Following anything other than a CAN SPARK MAX is not officially supported.

Parameters

leader	The type of motor controller to follow (Talon SRX, Spark Max, etc.).
deviceID	The CAN ID of the device to follow.

Returns

CANError Set to CANError.kOK if successful

3.9.2.11 follow() [4/4]

```
CANError com.revrobotics.CANSparkMax.follow (
            ExternalFollower leader,
            int deviceID,
            boolean invert )
```

Causes this controller's output to mirror the provided leader.

Only voltage output is mirrored. Settings changed on the leader do not affect the follower.

Following anything other than a CAN SPARK MAX is not officially supported.

Parameters

leader	The type of motor controller to follow (Talon SRX, Spark Max, etc.).
deviceID	The CAN ID of the device to follow.
invert	Set the follower to output opposite of the leader

Returns

CANError Set to CANError.kOK if successful

3.9.2.12 get()

```
double com.revrobotics.CANSparkMax.get ( )
```

Common interface for getting the current set speed of a speed controller.

Returns

The current set speed. Value is between -1.0 and 1.0.

3.9.2.13 getAnalog()

```
CANAnalog com.revrobotics.CANSparkMax.getAnalog (
            AnalogMode mode )
```

Parameters

mode	The mode of the analog sensor, either absolute or relative

Returns

An object for interfacing with a connected analog sensor.

3.9.2.14 getAppliedOutput()

```
double com.revrobotics.CANSparkMax.getAppliedOutput ( )
```

Returns

The motor controller's applied output duty cycle.

3.9.2.15 getBusVoltage()

```
double com.revrobotics.CANSparkMax.getBusVoltage ( )
```

Returns

The voltage fed into the motor controller.

3.9.2.16 getClosedLoopRampRate()

```
double com.revrobotics.CANSparkMax.getClosedLoopRampRate ( )
```

Get the configured closed loop ramp rate

This is the maximum rate at which the motor controller's output is allowed to change.

Returns

ramp rate time in seconds to go from 0 to full throttle.

3.9.2.17 getEncoder() [1/2]

```
CANEncoder com.revrobotics.CANSparkMax.getEncoder ( )
```

Assumes that the encoder the is integrated encoder, ie kHallEffect with 0 counts per revolution.

Returns

An object for interfacing with the integrated encoder.

3.9.2.18 getEncoder() [2/2]

```
CANEncoder com.revrobotics.CANSparkMax.getEncoder (
            SensorType sensorType,
            int cpr )
```

Parameters

sensorType	The encoder type for the motor: kHallEffect or kQuadrature
cpr	The counts per revolution of the encoder

Returns

An object for interfacing with an encoder

3.9.2.19 getFault()

```
boolean com.revrobotics.CANSparkMax.getFault (
            FaultID faultID )
```

Get the value of a specific fault

Parameters

faultID	The ID of the fault to retrive

Returns

True if the fault with the given ID occurred.

3.9.2.20 getFaults()

```
short com.revrobotics.CANSparkMax.getFaults ( )
```

Returns

All fault bits as a short

3.9.2.21 getFeedbackDeviceID()

```
int com.revrobotics.CANSparkMax.getFeedbackDeviceID ( )  [protected]
```

Gets the feedback device ID that was set on SparkMax itself.

Returns

Feedback device ID on the SparkMax

3.9.2.22 getForwardLimitSwitch()

```
CANDigitalInput com.revrobotics.CANSparkMax.getForwardLimitSwitch (
            CANDigitalInput.LimitSwitchPolarity polarity )
```

Returns

An object for interfacing with the integrated forward limit switch.

Parameters

polarity	Whether the limit switch is normally open or normally closed.

3.9.2.23 getIdleMode()

```
IdleMode com.revrobotics.CANSparkMax.getIdleMode ( )
```

Gets the idle mode setting for the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Returns

IdleMode Idle mode setting

3.9.2.24 getInverted()

```
boolean com.revrobotics.CANSparkMax.getInverted ( )
```

Common interface for returning the inversion state of a speed controller.

This call has no effect if the controller is a follower.

Returns

isInverted The state of inversion, true is inverted.

3.9.2.25 getLastError()

```
CANError com.revrobotics.CANSparkMax.getLastError ( )
```

All device errors are tracked on a per thread basis for all devices in that thread. This is meant to be called immediately following another call that has the possibility of returning an error to validate if an error has occurred.

Returns

the last error that was generated.

3.9.2.26 getMotorTemperature()

```
double com.revrobotics.CANSparkMax.getMotorTemperature ( )
```

Returns

The motor temperature in Celsius.

3.9.2.27 getOpenLoopRampRate()

```
double com.revrobotics.CANSparkMax.getOpenLoopRampRate ( )
```

Get the configured open loop ramp rate

This is the maximum rate at which the motor controller's output is allowed to change.

Returns

ramp rate time in seconds to go from 0 to full throttle.

3.9.2.28 getOutputCurrent()

```
double com.revrobotics.CANSparkMax.getOutputCurrent ( )
```

Returns

The motor controller's output current in Amps.

3.9.2.29 getPIDController()

```
CANPIDController com.revrobotics.CANSparkMax.getPIDController ( )
```

Returns

An object for interfacing with the integrated PID controller.

3.9.2.30 getReverseLimitSwitch()

```
CANDigitalInput com.revrobotics.CANSparkMax.getReverseLimitSwitch (
            CANDigitalInput.LimitSwitchPolarity polarity )
```

Returns

An object for interfacing with the integrated reverse limit switch.

Parameters

polarity	Whether the limit switch is normally open or normally closed.

3.9.2.31 getSoftLimit()

```
double com.revrobotics.CANSparkMax.getSoftLimit (
            SoftLimitDirection direction )
```

Get the soft limit setting in the controller

Parameters

direction	the direction of motion to restrict

Returns

position soft limit setting of the controller

3.9.2.32 getStickyFault()

```
boolean com.revrobotics.CANSparkMax.getStickyFault (
            FaultID faultID )
```

Get the value of a specific sticky fault

Parameters

faultID	The ID of the sticky fault to retrive

Returns

True if the sticky fault with the given ID occurred.

3.9.2.33 getStickyFaults()

```
short com.revrobotics.CANSparkMax.getStickyFaults ( )
```

Returns

All sticky fault bits as a short

3.9.2.34 getVoltageCompensationNominalVoltage()

```
double com.revrobotics.CANSparkMax.getVoltageCompensationNominalVoltage ( )
```

Get the configured voltage compensation nominal voltage value

Returns

The nominal voltage for voltage compensation mode.

3.9.2.35 isFollower()

```
boolean com.revrobotics.CANSparkMax.isFollower ( )
```

Returns whether the controller is following another controller

Returns

True if this device is following another controller false otherwise

3.9.2.36 isSoftLimitEnabled()

```
boolean com.revrobotics.CANSparkMax.isSoftLimitEnabled (
            SoftLimitDirection direction )
```

Parameters

direction	The direction of the motion to restrict

Returns

true if the soft limit is enabled.

3.9.2.37 set()

```
void com.revrobotics.CANSparkMax.set (
            double speed )
```

Common interface for setting the speed of a speed controller.

Parameters

speed	The speed to set. Value should be between -1.0 and 1.0.

3.9.2.38 setCANTimeout()

```
CANError com.revrobotics.CANSparkMax.setCANTimeout (
            int milliseconds )
```

Sets timeout for sending CAN messages with SetParameter∗ and GetParameter∗ calls. These calls will block for up to this amoutn of time before returning a timeout erro. A timeout of 0 will make the SetParameter∗ calls non-blocking, and instead will check the response in a separate thread. With this configuration, any error messages will appear on the drivestration but will not be returned by the GetLastError() call.

Parameters

milliseconds	The timeout in milliseconds.

Returns

CANError Set to CANError.kOK if successful

3.9.2.39 setClosedLoopRampRate()

```
CANError com.revrobotics.CANSparkMax.setClosedLoopRampRate (
            double rate )
```

Sets the ramp rate for closed loop control modes.

This is the maximum rate at which the motor controller's output is allowed to change.

Parameters

rate	Time in seconds to go from 0 to full throttle.

Returns

CANError Set to CANError.kOK if successful

3.9.2.40 setIdleMode()

```
CANError com.revrobotics.CANSparkMax.setIdleMode (
            IdleMode mode )
```

Sets the idle mode setting for the SPARK MAX.

Parameters

mode	Idle mode (coast or brake).

Returns

CANError Set to CANError.kOK if successful

3.9.2.41 setInverted()

```
void com.revrobotics.CANSparkMax.setInverted (
            boolean isInverted )
```

Common interface for inverting direction of a speed controller.

This call has no effect if the controller is a follower.

Parameters

isInverted	The state of inversion, true is inverted.

3.9.2.42 setOpenLoopRampRate()

```
CANError com.revrobotics.CANSparkMax.setOpenLoopRampRate (
            double rate )
```

Sets the ramp rate for open loop control modes.

This is the maximum rate at which the motor controller's output is allowed to change.

Parameters

rate	Time in seconds to go from 0 to full throttle.

Returns

CANError Set to CANError.kOK if successful

3.9.2.43 setSecondaryCurrentLimit() [1/2]

```
CANError com.revrobotics.CANSparkMax.setSecondaryCurrentLimit (
            double limit )
```

Sets the secondary current limit in Amps.

The motor controller will disable the output of the controller briefly if the current limit is exceeded to reduce the current. This limit is a simplified 'on/off' controller. This limit is enabled by default but is set higher than the default Smart Current Limit.

The time the controller is off after the current limit is reached is determined by the parameter limitCycles, which is the number of PWM cycles (20kHz). The recommended value is the default of 0 which is the minimum time and is part of a PWM cycle from when the over current is detected. This allows the controller to regulate the current close to the limit value.

The total time is set by the equation

```
t = (50us - t0) + 50us * limitCycles t = total off time after over current
t0 = time from the start of the PWM cycle until over current is detected
```

Parameters

limit	The current limit in Amps.

Returns

CANError Set to CANError.kOK if successful

3.9.2.44 setSecondaryCurrentLimit() [2/2]

```
CANError com.revrobotics.CANSparkMax.setSecondaryCurrentLimit (
            double limit,
            int chopCycles )
```

Sets the secondary current limit in Amps.

The motor controller will disable the output of the controller briefly if the current limit is exceeded to reduce the current. This limit is a simplified 'on/off' controller. This limit is enabled by default but is set higher than the default Smart Current Limit.

The time the controller is off after the current limit is reached is determined by the parameter limitCycles, which is the number of PWM cycles (20kHz). The recommended value is the default of 0 which is the minimum time and is part of a PWM cycle from when the over current is detected. This allows the controller to regulate the current close to the limit value.

The total time is set by the equation

```
t = (50us - t0) + 50us * limitCycles t = total off time after over current
t0 = time from the start of the PWM cycle until over current is detected
```

Parameters

limit	The current limit in Amps.
chopCycles	The number of additional PWM cycles to turn the driver off after overcurrent is detected.

Returns

CANError Set to CANError.kOK if successful

3.9.2.45 setSmartCurrentLimit() [1/3]

```
CANError com.revrobotics.CANSparkMax.setSmartCurrentLimit (
            int limit )
```

Sets the current limit in Amps.

The motor controller will reduce the controller voltage output to avoid surpassing this limit. This limit is enabled by default and used for brushless only. This limit is highly recommended when using the NEO brushless motor.

The NEO Brushless Motor has a low internal resistance, which can mean large current spikes that could be enough to cause damage to the motor and controller. This current limit provides a smarter strategy to deal with high current draws and keep the motor and controller operating in a safe region.

Parameters

limit	The current limit in Amps.

Returns

CANError Set to CANError.kOK if successful

3.9.2.46 setSmartCurrentLimit() [2/3]

```
CANError com.revrobotics.CANSparkMax.setSmartCurrentLimit (
            int stallLimit,
            int freeLimit )
```

Sets the current limit in Amps.

The motor controller will reduce the controller voltage output to avoid surpassing this limit. This limit is enabled by default and used for brushless only. This limit is highly recommended when using the NEO brushless motor.

The NEO Brushless Motor has a low internal resistance, which can mean large current spikes that could be enough to cause damage to the motor and controller. This current limit provides a smarter strategy to deal with high current draws and keep the motor and controller operating in a safe region.

The controller can also limit the current based on the RPM of the motor in a linear fashion to help with controllability in closed loop control. For a response that is linear the entire RPM range leave limit RPM at 0.

Parameters

stallLimit	The current limit in Amps at 0 RPM.
freeLimit	The current limit at free speed (5700RPM for NEO).

Returns

CANError Set to CANError.kOK if successful

3.9.2.47 setSmartCurrentLimit() [3/3]

```
CANError com.revrobotics.CANSparkMax.setSmartCurrentLimit (
            int stallLimit,
```

```
            int freeLimit,
            int limitRPM )
```

Sets the current limit in Amps.

The motor controller will reduce the controller voltage output to avoid surpassing this limit. This limit is enabled by default and used for brushless only. This limit is highly recommended when using the NEO brushless motor.

The NEO Brushless Motor has a low internal resistance, which can mean large current spikes that could be enough to cause damage to the motor and controller. This current limit provides a smarter strategy to deal with high current draws and keep the motor and controller operating in a safe region.

The controller can also limit the current based on the RPM of the motor in a linear fashion to help with controllability in closed loop control. For a response that is linear the entire RPM range leave limit RPM at 0.

Parameters

stallLimit	The current limit in Amps at 0 RPM.
freeLimit	The current limit at free speed (5700RPM for NEO).
limitRPM	RPM less than this value will be set to the stallLimit, RPM values greater than limitRPM will scale linearly to freeLimit

Returns

CANError Set to CANError.kOK if successful

3.9.2.48 setSoftLimit()

```
CANError com.revrobotics.CANSparkMax.setSoftLimit (
            SoftLimitDirection direction,
            float limit )
```

Set the soft limit based on position. The default unit is rotations, but will match the unit scaling set by the user.

Note that this value is not scaled internally so care must be taken to make sure these units match the desired conversion

Parameters

direction	the direction of motion to restrict
limit	position soft limit of the controller

Returns

CANError Set to CANError.kOK if successful

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANSparkMax.java

3.10 com.revrobotics.jni.CANSparkMaxJNI Class Reference

Inherits com.revrobotics.jni.RevJNIWrapper.

Static Public Member Functions

- static native long **c_SparkMax_Create** (int deviceId, int motortype)
- static native void **c_SparkMax_Destroy** (long handle)
- static native int **c_SparkMax_GetFirmwareVersion** (long handle)
- static native int **c_SparkMax_GetDeviceId** (long handle)
- static native int **c_SparkMax_SetMotorType** (long handle, int type)
- static native int **c_SparkMax_GetMotorType** (long handle)
- static native int **c_SparkMax_SetPeriodicFramePeriod** (long handle, int frameId, int periodMs)
- static native void **c_SparkMax_SetControlFramePeriod** (long handle, int periodMs)
- static native int **c_SparkMax_GetControlFramePeriod** (long handle)
- static native int **c_SparkMax_SetEncoderPosition** (long handle, float position)
- static native int **c_SparkMax_RestoreFactoryDefaults** (long handle, boolean persist)
- static native int **c_SparkMax_SetFollow** (long handle, int followerArbId, int followerCfg)
- static native float **c_SparkMax_SafeFloat** (float f)
- static native void **c_SparkMax_EnableExternalControl** (boolean enable)
- static native void **c_SparkMax_SetEnable** (boolean enable)
- static native int **c_SparkMax_SetpointCommand** (long handle, float value, int ctrlType, int pidSlot, float arbFeedforward, int arbFFUnits)
- static native int **c_SparkMax_SetInverted** (long handle, boolean inverted)
- static native boolean **c_SparkMax_GetInverted** (long handle)
- static native int **c_SparkMax_SetSmartCurrentLimit** (long handle, int stallLimit, int freeLimit, int limitRPM)
- static native int **c_SparkMax_GetSmartCurrentStallLimit** (long handle)
- static native int **c_SparkMax_GetSmartCurrentFreeLimit** (long handle)
- static native int **c_SparkMax_GetSmartCurrentLimitRPM** (long handle)
- static native int **c_SparkMax_SetSecondaryCurrentLimit** (long handle, float limit, int chopCycles)
- static native float **c_SparkMax_GetSecondaryCurrentLimit** (long handle)
- static native int **c_SparkMax_GetSecondaryCurrentLimitCycles** (long handle)
- static native int **c_SparkMax_SetIdleMode** (long handle, int idlemode)
- static native int **c_SparkMax_GetIdleMode** (long handle)
- static native int **c_SparkMax_EnableVoltageCompensation** (long handle, float nominalVoltage)
- static native float **c_SparkMax_GetVoltageCompensationNominalVoltage** (long handle)
- static native int **c_SparkMax_DisableVoltageCompensation** (long handle)
- static native int **c_SparkMax_SetOpenLoopRampRate** (long handle, float rate)
- static native float **c_SparkMax_GetOpenLoopRampRate** (long handle)
- static native int **c_SparkMax_SetClosedLoopRampRate** (long handle, float rate)
- static native float **c_SparkMax_GetClosedLoopRampRate** (long handle)
- static native boolean **c_SparkMax_IsFollower** (long handle)
- static native int **c_SparkMax_GetFaults** (long handle)
- static native int **c_SparkMax_GetStickyFaults** (long handle)
- static native boolean **c_SparkMax_GetFault** (long handle, int faultId)
- static native boolean **c_SparkMax_GetStickyFault** (long handle, int faultId)
- static native float **c_SparkMax_GetBusVoltage** (long handle)
- static native float **c_SparkMax_GetAppliedOutput** (long handle)
- static native float **c_SparkMax_GetOutputCurrent** (long handle)
- static native float **c_SparkMax_GetMotorTemperature** (long handle)
- static native int **c_SparkMax_ClearFaults** (long handle)
- static native int **c_SparkMax_BurnFlash** (long handle)
- static native int **c_SparkMax_SetCANTimeout** (long handle, int timeoutMs)

- static native int **c_SparkMax_EnableSoftLimit** (long handle, int dir, boolean enable)
- static native boolean **c_SparkMax_IsSoftLimitEnabled** (long handle, int dir)
- static native int **c_SparkMax_SetSoftLimit** (long handle, int dir, float limit)
- static native float **c_SparkMax_GetSoftLimit** (long handle, int dir)
- static native int **c_SparkMax_SetSensorType** (long handle, int sensorType)
- static native int **c_SparkMax_SetLimitPolarity** (long handle, int sw, int polarity)
- static native int **c_SparkMax_GetLimitPolarity** (long handle, int sw)
- static native boolean **c_SparkMax_GetLimitSwitch** (long handle, int sw)
- static native int **c_SparkMax_EnableLimitSwitch** (long handle, int sw, boolean enable)
- static native boolean **c_SparkMax_IsLimitEnabled** (long handle, int sw)
- static native float **c_SparkMax_GetAnalogPosition** (long handle)
- static native float **c_SparkMax_GetAnalogVelocity** (long handle)
- static native float **c_SparkMax_GetAnalogVoltage** (long handle)
- static native int **c_SparkMax_SetAnalogPositionConversionFactor** (long handle, float conversion)
- static native int **c_SparkMax_SetAnalogVelocityConversionFactor** (long handle, float conversion)
- static native float **c_SparkMax_GetAnalogPositionConversionFactor** (long handle)
- static native float **c_SparkMax_GetAnalogVelocityConversionFactor** (long handle)
- static native int **c_SparkMax_SetAnalogInverted** (long handle, boolean inverted)
- static native boolean **c_SparkMax_GetAnalogInverted** (long handle)
- static native int **c_SparkMax_SetAnalogAverageDepth** (long handle, int depth)
- static native int **c_SparkMax_GetAnalogAverageDepth** (long handle)
- static native int **c_SparkMax_SetAnalogMeasurementPeriod** (long handle, int samples)
- static native int **c_SparkMax_GetAnalogMeasurementPeriod** (long handle)
- static native int **c_SparkMax_SetAnalogMode** (long handle, int mode)
- static native int **c_SparkMax_GetAnalogMode** (long handle)
- static native float **c_SparkMax_GetEncoderPosition** (long handle)
- static native float **c_SparkMax_GetEncoderVelocity** (long handle)
- static native int **c_SparkMax_SetPositionConversionFactor** (long handle, float conversion)
- static native int **c_SparkMax_SetVelocityConversionFactor** (long handle, float conversion)
- static native float **c_SparkMax_GetPositionConversionFactor** (long handle)
- static native float **c_SparkMax_GetVelocityConversionFactor** (long handle)
- static native int **c_SparkMax_SetAverageDepth** (long handle, int depth)
- static native int **c_SparkMax_GetAverageDepth** (long handle)
- static native int **c_SparkMax_SetMeasurementPeriod** (long handle, int samples)
- static native int **c_SparkMax_GetMeasurementPeriod** (long handle)
- static native int **c_SparkMax_SetCPR** (long handle, int cpr)
- static native int **c_SparkMax_GetCPR** (long handle)
- static native int **c_SparkMax_SetEncoderInverted** (long handle, boolean inverted)
- static native boolean **c_SparkMax_GetEncoderInverted** (long handle)
- static native int **c_SparkMax_SetP** (long handle, int slotID, float gain)
- static native int **c_SparkMax_SetI** (long handle, int slotID, float gain)
- static native int **c_SparkMax_SetD** (long handle, int slotID, float gain)
- static native int **c_SparkMax_SetDFilter** (long handle, int slotID, float gain)
- static native int **c_SparkMax_SetFF** (long handle, int slotID, float gain)
- static native int **c_SparkMax_SetIZone** (long handle, int slotID, float IZone)
- static native int **c_SparkMax_SetOutputRange** (long handle, int slotID, float min, float max)
- static native float **c_SparkMax_GetP** (long handle, int slotID)
- static native float **c_SparkMax_GetI** (long handle, int slotID)
- static native float **c_SparkMax_GetD** (long handle, int slotID)
- static native float **c_SparkMax_GetDFilter** (long handle, int slotID)
- static native float **c_SparkMax_GetFF** (long handle, int slotID)
- static native float **c_SparkMax_GetIZone** (long handle, int slotID)
- static native float **c_SparkMax_GetOutputMin** (long handle, int slotID)
- static native float **c_SparkMax_GetOutputMax** (long handle, int slotID)
- static native int **c_SparkMax_SetSmartMotionMaxVelocity** (long handle, int slotID, float maxVel)

- static native int **c_SparkMax_SetSmartMotionMaxAccel** (long handle, int slotID, float maxAccel)
- static native int **c_SparkMax_SetSmartMotionMinOutputVelocity** (long handle, int slotID, float minVel)
- static native int **c_SparkMax_SetSmartMotionAccelStrategy** (long handle, int slotID, int accelStrategy)
- static native int **c_SparkMax_SetSmartMotionAllowedClosedLoopError** (long handle, int slotID, float allowedError)
- static native float **c_SparkMax_GetSmartMotionMaxVelocity** (long handle, int slotID)
- static native float **c_SparkMax_GetSmartMotionMaxAccel** (long handle, int slotID)
- static native float **c_SparkMax_GetSmartMotionMinOutputVelocity** (long handle, int slotID)
- static native int **c_SparkMax_GetSmartMotionAccelStrategy** (long handle, int slotID)
- static native float **c_SparkMax_GetSmartMotionAllowedClosedLoopError** (long handle, int slotID)
- static native int **c_SparkMax_SetIMaxAccum** (long handle, int slotID, float iMaxAccum)
- static native float **c_SparkMax_GetIMaxAccum** (long handle, int slotID)
- static native int **c_SparkMax_SetIAccum** (long handle, float iAccum)
- static native float **c_SparkMax_GetIAccum** (long handle)
- static native int **c_SparkMax_SetFeedbackDevice** (long handle, int sensorID)
- static native int **c_SparkMax_SetFeedbackDeviceRange** (long handle, float min, float max)
- static native int **c_SparkMax_GetFeedbackDeviceID** (long handle)
- static native int **c_SparkMax_GetAPIMajorRevision** ()
- static native int **c_SparkMax_GetAPIMinorRevision** ()
- static native int **c_SparkMax_GetAPIBuildRevision** ()
- static native int **c_SparkMax_GetAPIVersion** ()
- static native int **c_SparkMax_GetLastError** (long handle)

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/jni/CANSparkMaxJNI.java

3.11 com.revrobotics.CANSparkMaxLowLevel Class Reference

Inherits SpeedController.

Inherited by com.revrobotics.CANSparkMax.

Classes

- class **FollowConfig**
- enum MotorType
- enum PeriodicFrame
- class PeriodicStatus0
- class PeriodicStatus1
- class PeriodicStatus2

Public Member Functions

- CANSparkMaxLowLevel (int deviceID, MotorType type)
- int getFirmwareVersion ()
- void setControlFramePeriodMs (int periodMs)
- String getFirmwareString ()
- byte [ ] getSerialNumber ()
- int getDeviceId ()
- MotorType getInitialMotorType ()
- CANError setMotorType (MotorType type)
- MotorType getMotorType ()
- CANError setPeriodicFramePeriod (PeriodicFrame frameID, int periodMs)
- float **getSafeFloat** (float f)
- CANError restoreFactoryDefaults ()
- CANError restoreFactoryDefaults (boolean persist)

Static Public Member Functions

- static void enableExternalUSBControl (boolean enable)

Static Public Attributes

- static final int **kAPIMajorVersion** = CANSparkMaxJNI.c_SparkMax_GetAPIMajorRevision()
- static final int **kAPIMinorVersion** = CANSparkMaxJNI.c_SparkMax_GetAPIMinorRevision()
- static final int **kAPIBuildVersion** = CANSparkMaxJNI.c_SparkMax_GetAPIBuildRevision()
- static final int **kAPIVersion** = CANSparkMaxJNI.c_SparkMax_GetAPIVersion()

Protected Member Functions

- CANError **setEncPosition** (double value)
- CANError **setIAccum** (double value)

Protected Attributes

- long **m_sparkMax**
- final MotorType **m_motorType**

3.11.1 Constructor & Destructor Documentation

3.11.1.1 CANSparkMaxLowLevel()

```
com.revrobotics.CANSparkMaxLowLevel.CANSparkMaxLowLevel (
            int deviceID,
            MotorType type )
```

Create a new SPARK MAX Controller

Parameters

deviceID	The device ID.
type	The motor type connected to the controller. Brushless motors must be connected to their matching color and the hall sensor plugged in. Brushed motors must be connected to the Red and Black terminals only.

3.11.2 Member Function Documentation

3.11.2.1 enableExternalUSBControl()

```
static void com.revrobotics.CANSparkMaxLowLevel.enableExternalUSBControl (
            boolean enable )  [static]
```

Allow external controllers to recieve control commands over USB. For example, a configuration where the heartbeat (and enable/disable) is sent by the main controller, but control frames are sent by other CAN devices over USB.

This is global for all controllers on the same bus.

This does not disable sending control frames from this device. To prevent conflicts, do not enable this feature and also send Set() for SetReference() from the controllers you wish to control.

Parameters

enable	Enable or disable external control

3.11.2.2 getDeviceId()

```
int com.revrobotics.CANSparkMaxLowLevel.getDeviceId ( )
```

Get the configured Device ID of the SPARK MAX.

Returns

int device ID

3.11.2.3 getFirmwareString()

```
String com.revrobotics.CANSparkMaxLowLevel.getFirmwareString ( )
```

Get the firmware version of the SPARK MAX as a string.

Returns

std::string Human readable firmware version string

3.11.2.4 getFirmwareVersion()

```
int com.revrobotics.CANSparkMaxLowLevel.getFirmwareVersion ( )
```

Get the firmware version of the SPARK MAX.

Returns

uint32_t Firmware version integer. Value is represented as 4 bytes, Major.Minor.Build H.Build L

3.11.2.5 getInitialMotorType()

```
MotorType com.revrobotics.CANSparkMaxLowLevel.getInitialMotorType ( )
```

Get the motor type setting from when the SparkMax was created.

This does not use the Get Parameter API which means it does not read what motor type is stored on the SparkMax itself. Instead, it reads the stored motor type from when the SparkMax object was first created.

Returns

MotorType Motor type setting

3.11.2.6 getMotorType()

```
MotorType com.revrobotics.CANSparkMaxLowLevel.getMotorType ( )
```

Get the motor type setting for the SPARK MAX.

This uses the Get Parameter API and should be used infrequently. This function uses a non-blocking call and will return a cached value if the parameter is not returned by the timeout. The timeout can be changed by calling SetCANTimeout(int milliseconds)

Returns

MotorType Motor type setting

3.11.2.7 getSerialNumber()

```
byte [] com.revrobotics.CANSparkMaxLowLevel.getSerialNumber ( )
```

Get the unique serial number of the SPARK MAX. Not currently available.

Returns

byte[] Vector of bytes representig the unique serial number

3.11.2.8 restoreFactoryDefaults() [1/2]

```
CANError com.revrobotics.CANSparkMaxLowLevel.restoreFactoryDefaults ( )
```

Restore motor controller parameters to factory default until the next controller reboot

Returns

CANError Set to CANError::kOk if successful

3.11.2.9 restoreFactoryDefaults() [2/2]

```
CANError com.revrobotics.CANSparkMaxLowLevel.restoreFactoryDefaults (
            boolean persist )
```

Restore motor controller parameters to factory default

Parameters

persist	If true, burn the flash with the factory default parameters

Returns

CANError Set to CANError::kOk if successful

3.11.2.10 setControlFramePeriodMs()

```
void com.revrobotics.CANSparkMaxLowLevel.setControlFramePeriodMs (
            int periodMs )
```

Set the control frame send period for the native CAN Send thread.

Parameters

periodMs	The send period in milliseconds between 1ms and 100ms

3.11.2.11 setMotorType()

```
CANError com.revrobotics.CANSparkMaxLowLevel.setMotorType (
            MotorType type )
```

Set the motor type connected to the SPARK MAX.

This uses the Set Parameter API and should be used infrequently. The parameter does not presist unless burn↩ Flash() is called. The recommended method to configure this parameter is to use the SPARK MAX GUI to tune and save parameters.

Parameters

type	The type of motor connected to the controller. Brushless motors must be connected to their matching color and the hall sensor plugged in. Brushed motors must be connected to the Red and Black terminals only.

Returns

CANError Set to CANError::kOk if successful

3.11.2.12 setPeriodicFramePeriod()

```
CANError com.revrobotics.CANSparkMaxLowLevel.setPeriodicFramePeriod (
            PeriodicFrame frameID,
            int periodMs )
```

Set the rate of transmission for periodic frames from the SPARK MAX

Each motor controller sends back three status frames with different data at set rates. Use this function to change the default rates.

Defaults: Status0 - 10ms Status1 - 20ms Status2 - 50ms

This value is not stored in the FLASH after calling burnFlash() and is reset on powerup.

Refer to the SPARK MAX reference manual on details for how and when to configure this parameter.

Parameters

frameID	The frame ID can be one of PeriodicFrame type
periodMs	The rate the controller sends the frame to the controller.

Returns

CANError Set to CANError.kOK if successful

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANSparkMaxLowLevel.java

3.12 com.revrobotics.jni.CANSWDLJNI Class Reference

Inherits com.revrobotics.jni.RevJNIWrapper.

Static Public Member Functions

- static native void AddDevice (int numDevicesToAdd, int... deviceIDs)
- static native int RunSWDL (String fileName)

3.12.1 Member Function Documentation

3.12.1.1 AddDevice()

```
static native void com.revrobotics.jni.CANSWDLJNI.AddDevice (
            int numDevicesToAdd,
            int...  deviceIDs )  [static]
```

Add a device to be updated

Parameters

numDevicesToAdd	The total number of devices you wish to update. This must match the number of motor controller IDs you pass into the next parameter
deviceIDs	The device IDs you wish to update as a list of any size. Length must match the numDevicesToAdd parameter

3.12.1.2 RunSWDL()

```
static native int com.revrobotics.jni.CANSWDLJNI.RunSWDL (
             String fileName )  [static]
```

Begin the software update for the sparks. You can call this function repeatedly to get the status of the update

Parameters

fileName	The filename of the bin file to send to the Spark MAXs

Returns

The percent complete of the download

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/jni/CANSWDLJNI.java

3.13 com.revrobotics.ControlType Enum Reference

Public Member Functions

- **ControlType** (int value)

Public Attributes

- **kDutyCycle** =(0)
- **kVelocity** =(1)
- **kVoltage** =(2)
- **kPosition** =(3)
- **kSmartMotion** =(4)
- **kCurrent** =(5)
- **kSmartVelocity** =(6)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/ControlType.java

3.14 com.revrobotics.CANSparkMax.FaultID Enum Reference

Public Member Functions

- **FaultID** (int value)

Static Public Member Functions

- static FaultID **fromId** (int id)

Public Attributes

- **kBrownout** =(0)
- **kOvercurrent** =(1)
- **kIWDTReset** =(2)
- **kMotorFault** =(3)
- **kSensorFault** =(4)
- **kStall** =(5)
- **kEEPROMCRC** =(6)
- **kCANTX** =(7)
- **kCANRX** =(8)
- **kHasReset** =(9)
- **kDRVFault** =(10)
- **kOtherFault** =(11)
- **kSoftLimitFwd** =(12)
- **kSoftLimitRev** =(13)
- **kHardLimitFwd** =(14)
- **kHardLimitRev** =(15)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANSparkMax.java

3.15 com.revrobotics.CANSparkMax.IdleMode Enum Reference

Public Member Functions

- **IdleMode** (int value)

Static Public Member Functions

- static IdleMode **fromId** (int id)

Public Attributes

- **kCoast** =(0)
- **kBrake** =(1)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANSparkMax.java

3.16 com.revrobotics.CANSparkMax.InputMode Enum Reference

Public Member Functions

- **InputMode** (int value)

Static Public Member Functions

- static InputMode **fromId** (int id)

Public Attributes

- **kPWM** =(0)
- **kCAN** =(1)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANSparkMax.java

3.17 com.revrobotics.CANDigitalInput.LimitSwitch Enum Reference

Public Member Functions

- **LimitSwitch** (int value)

Public Attributes

- **kForward** =(0)
- **kReverse** =(1)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANDigitalInput.java

3.18 com.revrobotics.CANDigitalInput.LimitSwitchPolarity Enum Reference

Public Member Functions

- **LimitSwitchPolarity** (int value)

Public Attributes

- **kNormallyOpen** =(0)
- **kNormallyClosed** =(1)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANDigitalInput.java

3.19 com.revrobotics.CANSparkMaxLowLevel.MotorType Enum Reference

Public Member Functions

- **MotorType** (int value)

Static Public Member Functions

- static MotorType **fromId** (int id)

Public Attributes

- **kBrushed** =(0)
- **kBrushless** =(1)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANSparkMaxLowLevel.java

3.20 com.revrobotics.CANSparkMaxLowLevel.PeriodicFrame Enum Reference

Public Member Functions

- **PeriodicFrame** (int value)

Static Public Member Functions

- static PeriodicFrame **fromId** (int id)

Public Attributes

- **kStatus0** =(0)
- **kStatus1** =(1)
- **kStatus2** =(2)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANSparkMaxLowLevel.java

3.21 com.revrobotics.CANSparkMaxLowLevel.PeriodicStatus0 Class Reference

Public Attributes

- double **appliedOutput**
- short **faults**
- short **stickyFaults**
- byte **lock**
- MotorType **motorType**
- boolean **isFollower**
- boolean **isInverted**
- boolean **roboRIO**

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANSparkMaxLowLevel.java

3.22 com.revrobotics.CANSparkMaxLowLevel.PeriodicStatus1 Class Reference

Public Attributes

- double **sensorVelocity**
- byte **motorTemperature**
- double **busVoltage**
- double **outputCurrent**

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANSparkMaxLowLevel.java

3.23 com.revrobotics.CANSparkMaxLowLevel.PeriodicStatus2 Class Reference

Public Attributes

- double **sensorPosition**
- double **iAccum**

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANSparkMaxLowLevel.java

3.24 com.revrobotics.jni.RevJNIWrapper Class Reference

Inherited by com.revrobotics.jni.CANSparkMaxJNI, and com.revrobotics.jni.CANSWDLJNI.

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/jni/RevJNIWrapper.java

3.25 com.revrobotics.SensorType Enum Reference

Public Member Functions

- **SensorType** (int value)

Static Public Member Functions

- static SensorType **fromId** (int id)

Public Attributes

- **kNoSensor** =(0)
- **kHallSensor** =(1)
- **kEncoder** =(2)
- **kSensorless** =(3)
- **kAnalog** =(4)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/SensorType.java

3.26 com.revrobotics.CANSparkMax.SoftLimitDirection Enum Reference

Public Member Functions

- **SoftLimitDirection** (int value)

Static Public Member Functions

- static SoftLimitDirection **fromID** (int id)

Public Attributes

- **kForward** =(0)
- **kReverse** =(1)
- final int **value**

The documentation for this enum was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/CANSparkMax.java

3.27 com.revrobotics.SparkMax Class Reference

Inherits PWMSpeedController.

The documentation for this class was generated from the following file:

- C:/Users/Will/Src/SPARK-MAX-roboRIO/src/main/java/com/revrobotics/SparkMax.java

Index