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.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 ()

3.1.1 Constructor & Destructor Documentation

3.1.1.1 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.

3.1.2 Member Function Documentation

3.1.2.1 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

3.1.2.2 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

3.1.2.3 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

3.2 com.revrobotics.CANEncoder Class Reference

Public Member Functions

- CANEncoder (CANSparkMax device)
- double getPosition ()
- double getVelocity ()

3.2.1 Constructor & Destructor Documentation

3.2.1.1 CANEncoder()

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

Constructs a CANPIDController.

Parameters

device	The Spark Max to which the encoder is attached.

3.2.2 Member Function Documentation

3.2.2.1 getPosition()

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

Get the position of the motor. This returns the native units of 'rotations'.

Returns

Number of rotations of the motor

3.2.2.2 getVelocity()

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

Get the velocity of the motor. This returns the native units of 'RPM'.

Returns

Number the RPM of the motor

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

3.3 com.revrobotics.CANError Enum Reference

Public Attributes

- **kOK**
- **kError**
- **kTimeout**

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

3.4 com.revrobotics.CANPIDController Class Reference

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 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 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 getFF ()
- double getFF (int slotID)
- double getIZone ()
- double getIZone (int slotID)
- double getOutputMin ()
- double getOutputMin (int slotID)
- double getOutputMax ()
- double getOutputMax (int slotID)

3.4.1 Constructor & Destructor Documentation

3.4.1.1 CANPIDController()

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

Constructs a CANPIDController.

Parameters

device	The Spark Max this object configures.

3.4.2 Member Function Documentation

3.4.2.1 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

3.4.2.2 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

3.4.2.3 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

3.4.2.4 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

3.4.2.5 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

3.4.2.6 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

3.4.2.7 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

3.4.2.8 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

3.4.2.9 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

3.4.2.10 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

3.4.2.11 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

3.4.2.12 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

3.4.2.13 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

3.4.2.14 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

3.4.2.15 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

3.4.2.16 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

3.4.2.17 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

3.4.2.18 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

3.4.2.19 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

3.4.2.20 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

3.4.2.21 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

3.4.2.22 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

3.4.2.23 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

3.4.2.24 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

3.4.2.25 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

3.4.2.26 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

3.4.2.27 setReference() [1/3]

```
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)
ctrl	Is the control type

Returns

CANError Set to REV_OK if successful

3.4.2.28 setReference() [2/3]

```
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)
ctrl	Is the control type to override with
pidSlot	for this command

Returns

CANError Set to REV_OK if successful

3.4.2.29 setReference() [3/3]

```
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)
ctrl	Is the control type to override with
pidSlot	for this command
arbFeedforward	A value from -32.0 to 32.0 which is a 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

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

3.5 com.revrobotics.CANSparkMax Class Reference

Inherits com.revrobotics.CANSparkMaxLowLevel, and AutoCloseable.

Classes

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

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 ()
- 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 setRampRate (double rate)
- double getRampRate ()
- 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)

Additional Inherited Members

3.5.1 Constructor & Destructor Documentation

3.5.1.1 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.

3.5.2 Member Function Documentation

3.5.2.1 burnFlash()

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

Writes all settings to flash.

Returns

CANError Set to CANError.kOk if successful

3.5.2.2 clearFaults()

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

Clears all sticky faults.

Returns

CANError Set to CANError.kOk if successful

3.5.2.3 close()

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

Closes the SPARK MAX Controller

3.5.2.4 disable()

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

Common interface for disabling a motor.

3.5.2.5 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.

Parameters

leader	The motor controller to follow.

Returns

CANError Set to CANError.kOk if successful

3.5.2.6 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.

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

3.5.2.7 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.

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

3.5.2.8 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.

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

3.5.2.9 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.

3.5.2.10 getAppliedOutput()

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

Returns

The motor controller's applied output duty cycle.

3.5.2.11 getBusVoltage()

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

Returns

The voltage fed into the motor controller.

3.5.2.12 getEncoder()

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

Returns

An object for interfacing with the integrated encoder.

3.5.2.13 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.

3.5.2.14 getFaults()

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

Returns

All fault bits as a short

3.5.2.15 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.

3.5.2.16 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

3.5.2.17 getInverted()

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

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

Returns

isInverted The state of inversion, true is inverted.

3.5.2.18 getMotorTemperature()

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

Returns

The motor temperature in Celsius. **Note** This parameter is on the roadmap and will return a value between 0 and 255 inversly proportional to the temp sensor voltage.

3.5.2.19 getOutputCurrent()

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

Returns

The motor controller's output current in Amps.

3.5.2.20 getPIDController()

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

Returns

An object for interfacing with the integrated PID controller.

3.5.2.21 getRampRate()

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

Get the configured 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.

3.5.2.22 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.

3.5.2.23 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.

3.5.2.24 getStickyFaults()

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

Returns

All sticky fault bits as a short

3.5.2.25 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

3.5.2.26 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.

3.5.2.27 setCANTimeout()

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

Sets timeout for sending CAN messages.

Parameters

milliseconds	The timeout in milliseconds.

Returns

CANError Set to CANError.kOk if successful

3.5.2.28 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

3.5.2.29 setInverted()

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

Common interface for inverting direction of a speed controller.

Parameters

isInverted	The state of inversion, true is inverted.

3.5.2.30 setRampRate()

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

Sets the ramp rate for all 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

3.5.2.31 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

3.5.2.32 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

3.5.2.33 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

3.5.2.34 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

3.5.2.35 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

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

3.6 com.revrobotics.CANSparkMaxFrames Class Reference

Classes

- class **BurnFlashOut**
- interface DataFrame
- class **FirmwareIn**
- class **FollowerOut**
- class **GetParamIn**
- class **SetParamOut**
- class **SetpointOut**
- class **Status0In**
- class **Status1In**
- class **Status2In**
- class **StatusConfigOut**

Static Public Member Functions

- static int **packFloat32** (double val)
- static double **unpackFloat32** (int val)

Static Public Attributes

- static final int **CMD_API_SETPNT_SET** = 0x001
- static final int **CMD_API_DC_SET** = 0x002
- static final int **CMD_API_SPD_SET** = 0x012
- static final int **CMD_API_POS_SET** = 0x032
- static final int **CMD_API_VOLT_SET** = 0x042
- static final int **CMD_API_STAT0** = 0x060
- static final int **CMD_API_STAT1** = 0x061
- static final int **CMD_API_STAT2** = 0x062
- static final int **CMD_API_CLEAR_FAULTS** = 0x06E
- static final int **CMD_API_DRV_STAT** = 0x06A
- static final int **CMD_API_BURN_FLASH** = 0x072
- static final int **CMD_API_SET_FOLLOWER** = 0x073
- static final int **CMD_API_NACK** = 0x080
- static final int **CMD_API_ACK** = 0x081
- static final int **CMD_API_BROADCAST** = 0x090
- static final int **CMD_API_HEARTBEAT** = 0x092
- static final int **CMD_API_SYNC** = 0x093
- static final int **CMD_API_ID_QUERY** = 0x094
- static final int **CMD_API_ID_ASSIGN** = 0x095
- static final int **CMD_API_FIRMWARE** = 0x098
- static final int **CMD_API_PARAM_ACCESS** = 0x300

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

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

3.7 com.revrobotics.CANSparkMaxLowLevel Class Reference

Inherits SpeedController.

Inherited by com.revrobotics.CANSparkMax.

Classes

- enum ConfigParameter
- class **FollowConfig**
- enum MotorType
- enum ParameterStatus
- enum ParameterType
- enum PeriodicFrame
- class PeriodicStatus0
- class PeriodicStatus1
- class PeriodicStatus2

Public Member Functions

- CANSparkMaxLowLevel (int deviceID, MotorType type)
- int getFirmwareVersion ()
- String getFirmwareString ()
- byte [ ] getSerialNumber ()
- int getDeviceId ()
- CANError setMotorType (MotorType type)
- MotorType getMotorType ()
- CANError setPeriodicFramePeriod (PeriodicFrame frameID, int periodMs)
- void setControlFramePeriod (int periodMs)
- int getControlFramePeriod ()
- ParameterStatus **setParameter** (ConfigParameter parameterID, double value)
- ParameterStatus **setParameter** (ConfigParameter parameterID, int value)
- ParameterStatus **setParameter** (ConfigParameter parameterID, boolean value)
- Optional< Double > **getParameterDouble** (ConfigParameter parameterID)
- Optional< Integer > **getParameterInt** (ConfigParameter parameterID)
- Optional< Boolean > **getParameterBoolean** (ConfigParameter parameterID)
- ParameterStatus **setParameterCore** (ConfigParameter parameterID, ParameterType type, int value)
- Optional< Integer > **getParameterCore** (ConfigParameter parameterID, ParameterType expectedType)
- ParameterType **getParameterType** (ConfigParameter parameterID)

Static Public Attributes

- static final byte **kNumFirmwareRetries** = 10
- static final int **kDefaultControlFramePeriodMs** = 10
- static final int **kDefaultCANTimeoutMs** = 20
- static final int **kDefaultStatus0PeriodMs** = 10
- static final int **kDefaultStatus1PeriodMs** = 20
- static final int **kDefaultStatus2PeriodMs** = 50
- static final int **kMinFirmwareVersion** = 0x1000179

Protected Member Functions

- PeriodicStatus0 **getPeriodicStatus0** ()
- PeriodicStatus1 **getPeriodicStatus1** ()
- PeriodicStatus2 **getPeriodicStatus2** ()

Protected Attributes

- CAN **m_can**
- int **m_controlPeriodMs**
- int **m_canTimeoutMs**
- boolean **m_inverted**

3.7.1 Constructor & Destructor Documentation

3.7.1.1 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.

3.7.2 Member Function Documentation

3.7.2.1 getControlFramePeriod()

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

Return the rate of transmission for setpoint frame

Returns

Control frame perioid in milliseconds

3.7.2.2 getDeviceId()

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

Get the configured Device ID of the SPARK MAX.

Returns

int device ID

3.7.2.3 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

3.7.2.4 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

3.7.2.5 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

3.7.2.6 getSerialNumber()

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

Get the unique serial number of the SPARK MAX.

Returns

byte[] Vector of bytes representig the unique serial number

3.7.2.7 setControlFramePeriod()

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

Set the rate of transmission for setpoint frame. This will not take effect until the next call of a method that sets a new setpoint.

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

Parameters

periodMs	The rate to send control frames

3.7.2.8 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

3.7.2.9 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

3.8 com.revrobotics.CANSparkMaxLowLevel.ConfigParameter Enum Reference

Public Member Functions

- **ConfigParameter** (int value)

Static Public Member Functions

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

Public Attributes

- **kCanID** =(0)
- **kInputMode** =(1)
- **kMotorType** =(2)
- **kCommAdvance** =(3)
- **kSensorType** =(4)
- **kCtrlType** =(5)
- **kIdleMode** =(6)
- **kInputDeadband** =(7)
- **kFirmwareVer** =(8)
- **kHallOffset** =(9)
- **kPolePairs** =(10)
- **kCurrentChop** =(11)
- **kCurrentChopCycles** =(12)
- **kP_0** =(13)
- **kI_0** =(14)
- **kD_0** =(15)
- **kF_0** =(16)
- **kIZone_0** =(17)
- **kDFilter_0** =(18)
- **kOutputMin_0** =(19)
- **kOutputMax_0** =(20)
- **kP_1** =(21)
- **kI_1** =(22)
- **kD_1** =(23)
- **kF_1** =(24)
- **kIZone_1** =(25)
- **kDFilter_1** =(26)
- **kOutputMin_1** =(27)
- **kOutputMax_1** =(28)
- **kP_2** =(29)

- **kI_2** =(30)
- **kD_2** =(31)
- **kF_2** =(32)
- **kIZone_2** =(33)
- **kDFilter_2** =(34)
- **kOutputMin_2** =(35)
- **kOutputMax_2** =(36)
- **kP_3** =(37)
- **kI_3** =(38)
- **kD_3** =(39)
- **kF_3** =(40)
- **kIZone_3** =(41)
- **kDFilter_3** =(42)
- **kOutputMin_3** =(43)
- **kOutputMax_3** =(44)
- **kReserved** =(45)
- **kOutputRatio** =(46)
- **kSerialNumberLow** =(47)
- **kSerialNumberMid** =(48)
- **kSerialNumberHigh** =(49)
- **kLimitSwitchFwdPolarity** =(50)
- **kLimitSwitchRevPolarity** =(51)
- **kHardLimitFwdEn** =(52)
- **kHardLimitRevEn** =(53)
- **kSoftLimitFwdEn** =(54)
- **kSoftLimitRevEn** =(55)
- **kRampRate** =(56)
- **kFollowerID** =(57)
- **kFollowerConfig** =(58)
- **kSmartCurrentStallLimit** =(59)
- **kSmartCurrentFreeLimit** =(60)
- **kSmartCurrentConfig** =(61)
- **kSmartCurrentReserved** =(62)
- **kMotorKv** =(63)
- **kMotorR** =(64)
- **kMotorL** =(65)
- **kMotorRsvd1** =(66)
- **kMotorRsvd2** =(67)
- **kMotorRsvd3** =(68)
- **kEncoderCountsPerRev** =(69)
- **kEncoderAverageDepth** =(70)
- **kEncoderSampleDelta** =(71)
- 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

3.9 com.revrobotics.ControlType Enum Reference

Public Member Functions

- **ControlType** (int value)

Public Attributes

- **kDutyCycle** =(0)
- **kVelocity** =(1)
- **kVoltage** =(2)
- **kPosition** =(3)
- 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

3.10 com.revrobotics.CANSparkMaxFrames.DataFrame Interface Reference

Inherited by com.revrobotics.CANSparkMaxFrames.BurnFlashOut, com.revrobotics.CANSparkMaxFrames.↩ FirmwareIn, com.revrobotics.CANSparkMaxFrames.FollowerOut, com.revrobotics.CANSparkMaxFrames.Get↩ ParamIn, com.revrobotics.CANSparkMaxFrames.SetParamOut, com.revrobotics.CANSparkMaxFrames.Setpoint↩ Out, com.revrobotics.CANSparkMaxFrames.Status0In, com.revrobotics.CANSparkMaxFrames.Status1In, com.↩ revrobotics.CANSparkMaxFrames.Status2In, and com.revrobotics.CANSparkMaxFrames.StatusConfigOut.

Public Member Functions

- byte [ ] **Serialize** ()
- void **Deserialize** (byte[ ] buf)

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

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

3.11 com.revrobotics.CANSparkMax.FaultID Enum Reference

Public Member Functions

- **FaultID** (int value)

Public Attributes

- **kBrownout** =(0)
- **kOvercurrent** =(1)
- **kOvervoltage** =(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

3.12 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

3.13 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

3.14 com.revrobotics.CANDigitalInput.LimitSwitch Enum Reference

Public Attributes

- **kForward**
- **kReverse**

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

3.15 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

3.16 com.revrobotics.CANSparkMaxLowLevel.MotorType Enum Reference

Public Member Functions

- **MotorType** (int value)

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

3.17 com.revrobotics.CANSparkMaxLowLevel.ParameterStatus Enum Reference

Public Member Functions

- **ParameterStatus** (int value)

Public Attributes

- **kOK** =(0)
- **kInvalidID** =(1)
- **kMismatchType** =(2)
- **kAccessMode** =(3)
- **kInvalid** =(4)
- **kNotImplementedDeprecated** =(5)
- 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

3.18 com.revrobotics.CANSparkMaxLowLevel.ParameterType Enum Reference

Public Member Functions

- **ParameterType** (int value)

Public Attributes

- **kInt32** =(0)
- **kUint32** =(1)
- **kFloat32** =(2)
- **kBool** =(3)
- 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

3.19 com.revrobotics.CANSparkMaxLowLevel.PeriodicFrame Enum Reference

Public Member Functions

- **PeriodicFrame** (int value)

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

3.20 com.revrobotics.CANSparkMaxLowLevel.PeriodicStatus0 Class Reference

Public Attributes

- double **appliedOutput**
- short **faults**
- short **stickyFaults**
- byte **idleMode**
- MotorType **motorType**
- boolean **isFollower**

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

3.21 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

3.22 com.revrobotics.CANSparkMaxLowLevel.PeriodicStatus2 Class Reference

Public Attributes

- double **sensorPosition**

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.CANSparkMax.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)
- 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

Index