com._604robotics.utils

Class EncoderPIDSource

java.lang.Object
  edu.wpi.first.wpilibj.SensorBase
    edu.wpi.first.wpilibj.Encoder
      com._604robotics.utils.EncoderOffset
        com._604robotics.utils.EncoderPIDSource

All Implemented Interfaces:

CounterBase, IDevice, ISensor, PIDSource

```
public class EncoderPIDSource
extends EncoderOffset
```

Encoder extender that return the value of Encoder.get() when pidGet is called. Drop-in replacement: all constructors from the Encoder class are implemented here.

Author:

Michael Smith

Nested Class Summary

Nested classes/interfaces inherited from class edu.wpi.first.wpilibj.Encoder

Encoder.PIDSourceParameter

Nested classes/interfaces inherited from interface edu.wpi.first.wpilibj.CounterBase

CounterBase.EncodingType

Field Summary

Fields inherited from class edu.wpi.first.wpilibj.Encoder

m_aSource, m_bSource, m_indexSource

Fields inherited from class edu.wpi.first.wpilibj.SensorBase

kAnalogChannels, kAnalogModules, kDigitalChannels, kPwmChannels, kRelayChannels, kSolenoidChannels, kSolenoidModules, kSystemClockTicksPerMicrosecond

Constructor Summary

Constructors

Constructor and Description
EncoderPIDSource(**DigitalSource** aSource, **DigitalSource** bSource) Encoder constructor.
EncoderPIDSource(**DigitalSource** aSource, **DigitalSource** bSource, boolean reverseDirection) Encoder constructor.
EncoderPIDSource(**DigitalSource** aSource, **DigitalSource** bSource, boolean reverseDirection, **CounterBase.EncodingType** encodingType) Encoder constructor.
EncoderPIDSource(**DigitalSource** aSource, **DigitalSource** bSource, **DigitalSource** indexSource) Encoder constructor.
EncoderPIDSource(**DigitalSource** aSource, **DigitalSource** bSource, **DigitalSource** indexSource, boolean reverseDirection) Encoder constructor.
EncoderPIDSource(int aChannel, int bChannel) Encoder constructor.
EncoderPIDSource(int aChannel, int bChannel, boolean reverseDirection)

Encoder constructor.

EncoderPIDSource(int aChannel, int bChannel, boolean reverseDirection, **CounterBase.EncodingType** encodingType)
Encoder constructor.

EncoderPIDSource(int aChannel, int bChannel, int indexChannel)
Encoder constructor.

EncoderPIDSource(int aChannel, int bChannel, int indexChannel, boolean reverseDirection)
Encoder constructor.

EncoderPIDSource(int aSlot, int aChannel, int bSlot, int bChannel)
Encoder constructor.

EncoderPIDSource(int aSlot, int aChannel, int bSlot, int bChannel, boolean reverseDirection)
Encoder constructor.

EncoderPIDSource(int aSlot, int aChannel, int bSlot, int bChannel, boolean reverseDirection, **CounterBase.EncodingType** encodingType)
Encoder constructor.

EncoderPIDSource(int aSlot, int aChannel, int bSlot, int bChannel, int indexSlot, int indexChannel)
Encoder constructor.

EncoderPIDSource(int aSlot, int aChannel, int bSlot, int bChannel, int indexSlot, int indexChannel, boolean reverseDirection)
Encoder constructor.

Method Summary

Methods

Modifier and Type	Method and Description
double	**pidGet**() Hooks into the PIDSource interface.

Methods inherited from class com._604robotics.utils.EncoderOffset

getRaw, reset, setOffset

Methods inherited from class edu.wpi.first.wpilibj.Encoder

free, get, getDirection, getDistance, getPeriod, getRate, getStopped, setDistancePerPulse, setMaxPeriod, setMinRate, setPIDSourceParameter, setReverseDirection, start, stop

Methods inherited from class edu.wpi.first.wpilibj.SensorBase

checkAnalogChannel, checkAnalogModule, checkDigitalChannel, checkDigitalModule, checkPWMChannel, checkPWMModule, checkRelayChannel, checkRelayModule, checkSolenoidChannel, checkSolenoidModule, getDefaultAnalogModule, getDefaultDigitalModule, getDefaultSolenoidModule, setDefaultAnalogModule, setDefaultDigitalModule, setDefaultSolenoidModule

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EncoderPIDSource

```
public EncoderPIDSource(int aSlot,
                        int aChannel,
                        int bSlot,
                        int bChannel,
                        boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified.

Parameters:

`aSlot` - The a channel digital input module.

`aChannel` - The a channel digital input channel.

`bSlot` - The b channel digital input module.

`bChannel` - The b channel digital input channel.

`reverseDirection` - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

EncoderPIDSource

```
public EncoderPIDSource(int aSlot,
                int aChannel,
                int bSlot,
                int bChannel)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified.

Parameters:

`aSlot` - The a channel digital input module.

`aChannel` - The a channel digital input channel.

`bSlot` - The b channel digital input module.

`bChannel` - The b channel digital input channel.

EncoderPIDSource

```
public EncoderPIDSource(int aSlot,
                int aChannel,
                int bSlot,
                int bChannel,
                boolean reverseDirection,
                CounterBase.EncodingType encodingType)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified.

Parameters:

`aSlot` - The a channel digital input module.

`aChannel` - The a channel digital input channel.

`bSlot` - The b channel digital input module.

`bChannel` - The b channel digital input channel.

`reverseDirection` - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

`encodingType` - either k1X, k2X, or k4X to indicate 1X, 2X or 4X decoding. If 4X is selected, then an encoder FPGA object is used and the returned counts will be 4x the encoder spec'd value since all rising and falling edges are counted. If 1X or 2X are selected then a counter object will be used and the returned value will either exactly match the spec'd count or be double (2x) the spec'd count.

EncoderPIDSource

```
public EncoderPIDSource(int aSlot,
                int aChannel,
                int bSlot,
                int bChannel,
                int indexSlot,
                int indexChannel,
                boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified. Using the index pulse forces 4x encoding.

Parameters:

`aSlot` - The a channel digital input module.

`aChannel` - The a channel digital input channel.

`bSlot` - The b channel digital input module.

`bChannel` - The b channel digital input channel.

`indexSlot` - The index channel digital input module.

`indexChannel` - The index channel digital input channel.

`reverseDirection` - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

EncoderPIDSource

```
public EncoderPIDSource(int aSlot,
                int aChannel,
                int bSlot,
                int bChannel,
                int indexSlot,
                int indexChannel)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified. Using the index pulse forces 4x encoding.

Parameters:

`aSlot` - The a channel digital input module.

aSlot - The a channel digital input module.

aChannel - The a channel digital input channel.

bSlot - The b channel digital input module.

bChannel - The b channel digital input channel.

indexSlot - The index channel digital input module.

indexChannel - The index channel digital input channel.

EncoderPIDSource

```
public EncoderPIDSource(int aChannel,
                int bChannel,
                boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module.

Parameters:

aChannel - The a channel digital input channel.

bChannel - The b channel digital input channel.

reverseDirection - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

EncoderPIDSource

```
public EncoderPIDSource(int aChannel,
                int bChannel)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module.

Parameters:

aChannel - The a channel digital input channel.

bChannel - The b channel digital input channel.

EncoderPIDSource

```
public EncoderPIDSource(int aChannel,
                int bChannel,
                boolean reverseDirection,
                CounterBase.EncodingType encodingType)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module.

Parameters:

aChannel - The a channel digital input channel.

bChannel - The b channel digital input channel.

reverseDirection - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

encodingType - either k1X, k2X, or k4X to indicate 1X, 2X or 4X decoding. If 4X is selected, then an encoder FPGA object is used and the returned counts will be 4x the encoder spec'd value since all rising and falling edges are counted. If 1X or 2X are selected then a counter object will be used and the returned value will either exactly match the spec'd count or be double (2x) the spec'd count.

EncoderPIDSource

```
public EncoderPIDSource(int aChannel,
                int bChannel,
                int indexChannel,
                boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module. Using an index pulse forces 4x encoding

Parameters:

aChannel - The a channel digital input channel.

bChannel - The b channel digital input channel.

indexChannel - The index channel digital input channel.

reverseDirection - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

EncoderPIDSource

```
public EncoderPIDSource(int aChannel,
```

```
                int bChannel,
                int indexChannel)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module. Using an index pulse forces 4x encoding

Parameters:

`aChannel` - The a channel digital input channel.

`bChannel` - The b channel digital input channel.

`indexChannel` - The index channel digital input channel.

EncoderPIDSource

```
public EncoderPIDSource(DigitalSource aSource,
                DigitalSource bSource,
                boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

Parameters:

`aSource` - The source that should be used for the a channel.

`bSource` - the source that should be used for the b channel.

`reverseDirection` - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

EncoderPIDSource

```
public EncoderPIDSource(DigitalSource aSource,
                DigitalSource bSource)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

Parameters:

`aSource` - The source that should be used for the a channel.

`bSource` - the source that should be used for the b channel.

EncoderPIDSource

```
public EncoderPIDSource(DigitalSource aSource,
                DigitalSource bSource,
                boolean reverseDirection,
                CounterBase.EncodingType encodingType)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

Parameters:

`aSource` - The source that should be used for the a channel.

`bSource` - the source that should be used for the b channel.

`reverseDirection` - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

`encodingType` - either k1X, k2X, or k4X to indicate 1X, 2X or 4X decoding. If 4X is selected, then an encoder FPGA object is used and the returned counts will be 4x the encoder spec'd value since all rising and falling edges are counted. If 1X or 2X are selected then a counter object will be used and the returned value will either exactly match the spec'd count or be double (2x) the spec'd count.

EncoderPIDSource

```
public EncoderPIDSource(DigitalSource aSource,
                DigitalSource bSource,
                DigitalSource indexSource,
                boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

Parameters:

`aSource` - The source that should be used for the a channel.

`bSource` - the source that should be used for the b channel.

`indexSource` - the source that should be used for the index channel.

`reverseDirection` - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

EncoderPIDSource

```
public EncoderPIDSource(DigitalSource aSource,
                        DigitalSource bSource,
                        DigitalSource indexSource)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

Parameters:

`aSource` - The source that should be used for the a channel.

`bSource` - the source that should be used for the b channel.

`indexSource` - the source that should be used for the index channel.

Method Detail

pidGet

```
public double pidGet()
```

Hooks into the PIDSource interface. This method overrides the one implemented by the underlying Encoder class, simply returning the value of this.get();

Specified by:

`pidGet` in interface `PIDSource`

Overrides:

`pidGet` in class `Encoder`

Returns:

The value to pass back to the PIDSource; in this case, that of this.get();