Half step Rotary Encoder

1.0.1

Generated by Doxygen 1.8.11

Contents

1 3 speed Rotary Encoder Half Step library for Arduino **1**

2 Class Index **5**

2.1 Class List 5

3 File Index **7**

3.1 File List 7

4 Class Documentation **9**

4.1 RotaryHalfStep Class Reference 9

4.1.1 Detailed Description 9

4.1.2 Constructor & Destructor Documentation 9

4.1.2.1 RotaryHalfStep(uint8_t pin1, uint8_t pin2, bool pullUp=true, uint8_t sensitivity=100) 9

4.1.3 Member Function Documentation 10

4.1.3.1 getSensitivity() 10

4.1.3.2 read() 10

4.1.3.3 setSensitivity(uint8_t sensitivity) 10

5 File Documentation **11**

5.1 RotaryHalfStep.cpp File Reference 11

5.1.1 Detailed Description 11

5.2 RotaryHalfStep.h File Reference 12

5.2.1 Detailed Description 12

Index **13**

Chapter 1

3 speed Rotary Encoder Half Step library for Arduino

This is an optimized three speed Rotary Encoder library for Arduino which supports:

- Half step Rotary Encoder types.
- Detect three rotation speeds.
- Configurable rotation speed sensitivity.
- Polling and interrupt based.
- Single or multiple Rotary Encoders.
- Optional Rotary button.
- Pin state table in flash.

Half step / half step Rotary Encoders

The difference between a half step or half step Rotary Encoder type is how the data signals of the two pins are generated. It depends on the mechanical construction of the notches and contacts inside the Rotary Encoder.

Please refer to the `ErriezRotaryEncoderFullStep` library for full step Rotary Encoders. Experiment with the full step and half step libraries which works optimal for your Rotary Encoder.

Hardware

Connect the two rotary pins to the DIGITAL pins of an Arduino board.

A third rotary button pin is not used in the Rotary library, but can be used in the sketch.

Tested with Arduino IDE v1.8.5 on hardware:

- Arduino UNO
- Arduino Nano
- Arduino Micro
- Arduino Pro or Pro Mini
- Arduino Mega or Mega2560
- Arduino Leonardo
- WeMos D1 R2 & mini (ESP8266)

Interrupts

Both rotary pins must be connected to a DIGITAL pin with interrupt support, such as `INT0` or `INT1`. This is chip specific. Please refer to the documentation of your board or `attachInterrupt()`.

Arduino UNO hardware

The connection below can be used for polled and interrupts. An optional button pin can be connected to DIGITAL pin 4.

Rotary pin	Arduino UNO/NANO/Mega2560/Leonardo board
1	DIGITAL pin 2 (INT0)
2	DIGITAL pin 3 (INT1)
Button (optional)	DIGITAL pin 4
GND	GND

Arduino WeMos D1 R2 & mini (ESP8266) hardware

Note that some ESP8266 pins mixes ESP8622 GPIO pins with Arduino digital pins. Connect a Rotary Encoder to the following pins which can be used with polled and interrupt examples:

Rotary pin	ESP8622 pin	Text on board / WeMos D1 & R2
1	GPIO13	D7 (MOSI)
2	GPIO12	D6 (MISO)
Button (optional)	GPIO14	D5 (SCK)
LED (Not used)	GPIO2	D4
GND	GND	GND

Note: An external pull-up resistor is required when a pin does not have an internal pull-up.

```
1 {c++}
2 // Connect the rotary pins to the WeMos D1 R2 board:
3 #define ROTARY_PIN1         12
4 #define ROTARY_PIN2         13
5 #define ROTARY_BUTTON_PIN   14
```

Examples

The following examples are available:

- Rotary | Interrupt | `InterruptHalfStepBasic`
- Rotary | Interrupt | `InterruptHalfStepButton`
- Rotary | Interrupt | `InterruptHalfStepCounter`
- Rotary | Polled | `PolledHalfStepBasic`
- Rotary | Polled | `PolledHalfStepButton`
- Rotary | Polled | `PolledHalfStepCounter`
- Rotary | Polled | `PolledHalfStepMultiple`

Documentation

- Doxygen online HTML
- Doxygen PDF

Usage

Read rotary with polling

```
{c++}
#include <RotaryHalfStep.h>

// Connect rotary pins to the DIGITAL pins of the Arduino board
#define ROTARY_PIN1   2
#define ROTARY_PIN2   3

// Enable ONE of the three constructors below with different number of arguments:

// Initialize half step rotary encoder, default pull-up enabled, default
// sensitive=100
RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2);

// Or initialize half step rotary encoder, pull-up disabled, default sensitive=100
// RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2, false);

// Or initialize half step rotary encoder, pull-up enabled, sensitive 1..255
// A higher value is more sensitive
// RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2, true, 150);

void loop()
{
  int rotaryState = rotary.read();

  // rotaryState = -3: Counter clockwise turn, multiple notches fast
  // rotaryState = -2: Counter clockwise turn, multiple notches
  // rotaryState = -1: Counter clockwise turn, single notch
  // rotaryState = 0:  No change
  // rotaryState = 1:  Clockwise turn, single notch
  // rotaryState = 2:  Clockwise turn, multiple notches
  // rotaryState = 3:  Clockwise turn, multiple notches fast
}
```

Read rotary with interrupts

```
{c++}
#include <RotaryHalfStep.h>

// Connect rotary pins to Arduino DIGITAL pins with interrupt support:
//
// +-----------------------------------+--------------------------+
// |              Board                |  DIGITAL interrupt pins  |
// +-----------------------------------+--------------------------+
// | Uno, Nano, Mini, other 328-based  |  2, 3                    |
// | Mega, Mega2560, MegaADK           |  2, 3, 18, 19, 20, 21    |
// | Micro, Leonardo, other 32u4-based |  0, 1, 2, 3, 7           |
// +-----------------------------------+--------------------------+
//
#define ROTARY_PIN1   2
#define ROTARY_PIN2   3

// Enable ONE of the three constructors below with different number of arguments:

// Initialize half step rotary encoder, default pull-up enabled, default
// sensitive=100
RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2);

// Or initialize half step rotary encoder, pull-up disabled, default sensitive=100
// RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2, false);

// Or initialize half step rotary encoder, pull-up enabled, sensitive 1..255
// A higher value is more sensitive
// RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2, true, 150);

void setup()
{
```

```
32   // Initialize pin change interrupt on both rotary encoder pins
33   attachInterrupt(digitalPinToInterrupt(ROTARY_PIN1), rotaryInterrupt, CHANGE);
34   attachInterrupt(digitalPinToInterrupt(ROTARY_PIN2), rotaryInterrupt, CHANGE);
35 }
36
37 void rotaryInterrupt()
38 {
39   int rotaryState = rotary.read();
40
41   // rotaryState = -3: Counter clockwise turn, multiple notches fast
42   // rotaryState = -2: Counter clockwise turn, multiple notches
43   // rotaryState = -1: Counter clockwise turn, single notch
44   // rotaryState = 0:  No change
45   // rotaryState = 1:  Clockwise turn, single notch
46   // rotaryState = 2:  Clockwise turn, multiple notches
47   // rotaryState = 3:  Clockwise turn, multiple notches fast
48 }
```

Library dependencies

- None.

Library installation

Please refer to the Wiki page.

Other Arduino Libraries and Sketches from Erriez

- Erriez Libraries and Sketches

Chapter 2

Class Index

2.1 Class List

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

RotaryHalfStep
Half step Rotary Encoder class 9

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

RotaryHalfStep.cpp
Three speed half step Rotary Encoder library for Arduino 11
RotaryHalfStep.h
Three speed half step Rotary Encoder library for Arduino 12

Chapter 4

Class Documentation

4.1 RotaryHalfStep Class Reference

Half step Rotary Encoder class.

`#include <RotaryHalfStep.h>`

Public Member Functions

- RotaryHalfStep (uint8_t pin1, uint8_t pin2, bool pullUp=true, uint8_t sensitivity=100)
  Constructor half step Rotary Encoder.
- int read ()
  Read Rotary Encoder state.
- void setSensitivity (uint8_t sensitivity)
  Set sensitivity value.
- uint8_t getSensitivity ()
  Get sensitivity value.

4.1.1 Detailed Description

Half step Rotary Encoder class.

Definition at line 41 of file RotaryHalfStep.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 RotaryHalfStep::RotaryHalfStep (uint8_t *pin1*, uint8_t *pin2*, bool *pullUp* = `true`, uint8_t *sensitivity* = `100`)

Constructor half step Rotary Encoder.

Parameters

pin1	Rotary Encoder pin 1
pin2	Rotary Encoder pin 2
pullUp	true: Enable internal pull-up on Rotary Encoder pins [default argument]. false: Disable internal pull-up on Rotary Encoder pins.
sensitivity	Set sensitivity rotation speed value 0..255. A higher is more sensitive for rotation speed, a smaller value is less sensitive or will disable speed detection. Default is 100.

Definition at line 84 of file RotaryHalfStep.cpp.

4.1.3 Member Function Documentation

4.1.3.1 uint8_t RotaryHalfStep::getSensitivity ( )

Get sensitivity value.

Returns

Sensitivity value 0..255.

Definition at line 175 of file RotaryHalfStep.cpp.

4.1.3.2 int RotaryHalfStep::read ( )

Read Rotary Encoder state.

Call this function as fast as possible to prevent missing pin changes. This seems to work for most rotary encoders when calling this function within 10ms in an endless loop.

The sensitivity value is used to calculate three rotation speeds by measuring the speed of the Rotary Encoder pin changes. The rotation speed depends on the number of Rotary notches and knob size. The value should experimentally determined.

Returns

Rotary speed and direction -3: Counter clockwise turn, multiple notches fast -2: Counter clockwise turn, multiple notches -1: Counter clockwise turn, single notch 0: No change 1: Clockwise turn, single notch 2: Clockwise turn, multiple notches 3: Clockwise turn, multiple notches fast

Definition at line 115 of file RotaryHalfStep.cpp.

4.1.3.3 void RotaryHalfStep::setSensitivity (uint8_t *sensitivity*)

Set sensitivity value.

Parameters

sensitivity	Sensitivity value 0..255

Definition at line 164 of file RotaryHalfStep.cpp.

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

- RotaryHalfStep.h
- RotaryHalfStep.cpp

Chapter 5

File Documentation

5.1 RotaryHalfStep.cpp File Reference

Three speed half step Rotary Encoder library for Arduino.

```
#include <pgmspace.h>
#include "RotaryHalfStep.h"
```

Macros

- #define DIR_NONE 0x00
  No complete step yet.
- #define DIR_CW 0x10
  Clockwise step.
- #define DIR_CCW 0x20
  Counter-clockwise step.
- #define RHS_START 0x00
  Rotary half step start.
- #define RHS_CCW_BEGIN 0x01
  Rotary half step counter clock wise begin.
- #define RHS_CW_BEGIN 0x02
  Rotary half step clock wise begin.
- #define RHS_START_M 0x03
  Rotary half step start.
- #define RHS_CW_BEGIN_M 0x04
  Rotary half step clock wise begin.
- #define RHS_CCW_BEGIN_M 0x05
  Rotary half step counter clock wise begin.

5.1.1 Detailed Description

Three speed half step Rotary Encoder library for Arduino.

Source: https://github.com/Erriez/ErriezRotaryEncoderHalfStep Documentation↩
: https://erriez.github.io/ErriezRotaryEncoderHalfStep

5.2 RotaryHalfStep.h File Reference

Three speed half step Rotary Encoder library for Arduino.

```
#include <Arduino.h>
```

Classes

- class RotaryHalfStep

  Half step Rotary Encoder class.

5.2.1 Detailed Description

Three speed half step Rotary Encoder library for Arduino.

Source: https://github.com/Erriez/ErriezRotaryEncoderHalfStep Documentation: https://erriez.github.io/ErriezRotaryEncoderHalfStep

Index

getSensitivity
    RotaryHalfStep, 10

read
    RotaryHalfStep, 10
RotaryHalfStep, 9
    getSensitivity, 10
    read, 10
    RotaryHalfStep, 9
    setSensitivity, 10
RotaryHalfStep.cpp, 11
RotaryHalfStep.h, 12

setSensitivity
    RotaryHalfStep, 10