ButtonDebounce

1.1

Generated by Doxygen 1.9.8

Chapter 1

ButtonDebounce

Library for push buttons (software debounce), Work in progress

A class for push buttons which is pretty debounce resistant. Idea after https://hackaday.com/2015/12/10/embed-with-elliot-debounce-your-noisy-buttons-part-ii/ form Elliot Williams.

For usage you have to call the updateButton-Method regularly, I recommened a task scheduler. Examples will follow soon.

Check for button presses with isPressed and isLongPressed. The return value will give you if the button is pressed, the parameter only chooses if you want to execute the corresponding function for the button.

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 ButtonDebounce Class Reference

Button Class using software debounce.

`#include <ButtonDebounce.h>`

Public Member Functions

- ButtonDebounce (unsigned char pin, bool pullUp=true, bool executeAtRelease=false, void(∗bFunction)()=nullptr)

  Construct a new Button Debounce object.
- bool isPressed (bool execute=false)

  Checks if the button is short pressed.
- bool isLongPressed (bool execute=false)

  Checks if the button is long pressed.
- bool stillPressed (bool execute)

  Checks if a Button is still pressed.
- bool anyPressed (bool execute)

  Checks if a button is newly pressed or still pressed.
- void updateButton ()

  Has to be called regularly! Update the current status of the button. This method reads the current status of the button and stores in history.
- uint8_t getButtonHistory ()

  For Debugging. Get current button history.

Setter Methods

Setter Methods for objects of the ButtonDebounce class.

All Setter Methods available. The functions always returns a true value. Pin cannot be changed after initialization. If a long press behavior is wanted, one can set the duration (Standard is 1000ms) and the function, which shall be executed automatically.

- bool **setPullUp** (bool pullUp)
- bool **setExecuteAtRelease** (bool executeAtRelease)
- bool **setFunction** (void(∗bFunction)())
- bool setLongPressDuration (unsigned long duration)
- bool setLongPressFunction (void(∗bFunction)())

4.1.1 Detailed Description

Button Class using software debounce.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ButtonDebounce()

```
ButtonDebounce::ButtonDebounce (
            unsigned char pin,
            bool pullUp = true,
            bool executeAtRelease = false,
            void(*)() bFunction = nullptr )
```

Construct a new Button Debounce object.

Parameters

pin	Physical pin at microcontroller
pullUp	Chooses if internal pullup resistor should be used. Does not check if pullup is present.
executeAtRelease	True: Function is executed at button release. False: Function is executed at button press.
bFunction	Pointer to function which shall be executed at button press

Constructor for ButtonDebounce object. A pin is required, all other arguemnts are optional. Additionally the longPressDuration is set to 1000ms, no longpress function is registered. This has to be done via the corresponding setter methods.

4.1.3 Member Function Documentation

4.1.3.1 anyPressed()

```
bool ButtonDebounce::anyPressed (
            bool execute = false )
```

Checks if a button is newly pressed or still pressed.

Parameters

execute	If the corresponding function should be executed

Returns

true if the button is pressed

false if the button is not pressed

4.1.3.2 getButtonHistory()

```
uint8_t ButtonDebounce::getButtonHistory ( )
```

For Debugging. Get current button history.

Returns

Current button history This method is solely meant for class debugging purposes. Can be used e.g. for checking if updating the button history works.

4.1.3.3 isLongPressed()

```
bool ButtonDebounce::isLongPressed (
            bool execute = false )
```

Checks if the button is long pressed.

Parameters

execute	If the corresponding function should be executed

Returns

if the button is long pressed

4.1.3.4 isPressed()

```
bool ButtonDebounce::isPressed (
            bool execute = false )
```

Checks if the button is short pressed.

Parameters

execute	If the corresponding function should be executed

Returns

if the button is pressed

4.1.3.5 setLongPressDuration()

```
bool ButtonDebounce::setLongPressDuration (
            unsigned long duration )
```

Set the duration that a long press is registered

4.1.3.6 setLongPressFunction()

```
bool ButtonDebounce::setLongPressFunction (
            void(*)() bFunction )
```

Set the function which can be automatically called when a long press is registered.

4.1.3.7 stillPressed()

```
bool ButtonDebounce::stillPressed (
            bool execute = false )
```

Checks if a Button is still pressed.

Parameters

execute	If the corresponding function should be executed

Returns

true if the button is pressed

false if the button is not pressed

4.1.3.8 updateButton()

```
void ButtonDebounce::updateButton ( )
```

Has to be called regularly! Update the current status of the button. This method reads the current status of the button and stores in history.

This has to be called regularly to register button presses! If not called regularly, button presses will be missed!

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

- C:/Users/Timo/Documents/Arduino/libraries/ButtonDebounce/ButtonDebounce.h
- C:/Users/Timo/Documents/Arduino/libraries/ButtonDebounce/ButtonDebounce.cpp

Chapter 5

File Documentation

5.1 C:/Users/Timo/Documents/Arduino/libraries/ButtonDebounce/↩ ButtonDebounce.h File Reference

Register Button Presses with Software Debounce.

```
#include "Arduino.h"
```

Classes

- class ButtonDebounce

  Button Class using software debounce.

5.1.1 Detailed Description

Register Button Presses with Software Debounce.

Author

Timo Raab

Version

1.1

Date

2023-12-23

Copyright

Buttons class for momentary buttons (not switches). The class allows for far better debouncing in buttons. Debouncing is completely done via software.

Note

Idea after https://hackaday.com/2015/12/10/embed-with-elliot-debounce-your-noisy-buttons from Elliot Williams

5.2 ButtonDebounce.h

Go to the documentation of this file.

```
00001
00020 #ifndef ButtonDebounce_h
00021 #define ButtonDebounce_h
00022
00023 #include "Arduino.h"
00024
00025
00030 class ButtonDebounce {
00031
00032     private:
00033         unsigned char _pin;             // Pin
00034         bool _pullUp;                   // Use of internal pull up resistor,
00035                                             // if not pull down is assumend,
00036                                             // standard: true
00037
00038         //Standard Operation
00039         bool _executeAtRelease;         // Choose, when the button press should be registered
00040                                         // Standard: at buttonPress (_executeAtRelease = false)
00041         void (*_bFunc)();               // function call at button press
00042
00043         //Long Press Operation, only available with setter-functions
00044         //Long Press is only available with execute at start
00045         unsigned long _longPressDuration;// Duration till long press triggers
00046         void (*_bFuncLong)();           // function call at long press activation
00047
00048         //Internal handling
00049         uint8_t _buttonHistory;         // saves history for debounce
00050         bool _isPressedTemp;            // for longPress needed
00051         unsigned long _pressTimeTemp;   // time when button is pressed for longPress
00052
00053         uint8_t readButton();           // read current button status
00054
00055
00056
00057     public:
00069         ButtonDebounce(unsigned char pin, bool pullUp = true, bool executeAtRelease = false, void
      (*bFunction)() = nullptr) ;
00070
00081         bool setPullUp(bool pullUp);
00082         bool setExecuteAtRelease(bool executeAtRelease);
00083         bool setFunction(void (*bFunction)());
00085         bool setLongPressDuration(unsigned long duration);
00087         bool setLongPressFunction(void (*bFunction)());
00089
00095         bool isPressed(bool execute = false);
00096
00097
00103         bool isLongPressed(bool execute = false);
00104
00112         bool stillPressed(bool execute);
00113
00121         bool anyPressed(bool execute);
00122
00129         void updateButton();
00130
00136         uint8_t getButtonHistory();
00137 };
00138
00139 #endif
00140
00141 //EOF
```

Index