

Button Debounce

Generated by Doxygen 1.10.0

1 ButtonDebounce	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 ButtonDebounce Class Reference	7
4.1.1 Constructor & Destructor Documentation	8
4.1.1.1 ButtonDebounce()	8
4.1.2 Member Function Documentation	8
4.1.2.1 anyPressed() [1/2]	8
4.1.2.2 anyPressed() [2/2]	8
4.1.2.3 getButtonHistory()	9
4.1.2.4 isLongPressed()	9
4.1.2.5 isPressed()	9
4.1.2.6 setLongPressDuration()	10
4.1.2.7 setLongPressFunction()	10
4.1.2.8 stillPressed()	10
4.1.2.9 updateButton()	10
5 File Documentation	11
5.1 ButtonDebounce.h File Reference	11
5.1.1 Detailed Description	11
5.2 ButtonDebounce.h	12
Index	15

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-elliott-debounce-your-noisy-buttons-part-ii/> from Elliot Williams.

For usage you have to call the updateButton-Method regularly, I recommended 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:

ButtonDebounce	7
--	---

Chapter 3

File Index

3.1 File List

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

ButtonDebounce.h	
Register Button Presses with Software Debounce	11

Chapter 4

Class Documentation

4.1 ButtonDebounce Class Reference

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=false)
Checks if a Button is still pressed.
- bool [anyPressed](#) (bool execute=false)
Checks if a button is newly pressed or still pressed.
- bool [anyPressed](#) (uint8_t executeNumber=EXECUTENUMBERNONE)
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 Constructor & Destructor Documentation

4.1.1.1 ButtonDebounce()

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

Construct a new Button Debounce object.

Parameters

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

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

4.1.2 Member Function Documentation

4.1.2.1 anyPressed() [1/2]

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

Checks if a button is newly pressed or still pressed.

Parameters

<i>execute</i>	If the corresponding function should be executed
----------------	--

Returns

true if the button is pressed
false if the button is not pressed

4.1.2.2 anyPressed() [2/2]

```
bool ButtonDebounce::anyPressed (
    uint8_t executeNumber = EXECUTENUMBERNONE )
```

Checks if a button is newly pressed or still pressed.

Parameters

<i>executeNumber</i>	If and when the corresponding function should be executed
----------------------	---

Returns

true if the button is pressed
false if the button is not pressed

4.1.2.3 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.

4.1.2.4 isLongPressed()

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

Checks if the button is long pressed.

Parameters

<i>execute</i>	If the corresponding function should be executed
----------------	--

Returns

if the button is long pressed

4.1.2.5 isPressed()

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

Checks if the button is short pressed.

Parameters

<i>execute</i>	If the corresponding function should be executed
----------------	--

Returns

if the button is pressed

4.1.2.6 setLongPressDuration()

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

Set the duration that a long press is registered

4.1.2.7 setLongPressFunction()

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

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

4.1.2.8 stillPressed()

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

Checks if a Button is still pressed.

Parameters

<i>execute</i>	If the corresponding function should be executed
----------------	--

Returns

true if the button is pressed
false if the button is not pressed

4.1.2.9 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:

- [ButtonDebounce.h](#)
- ButtonDebounce.cpp

Chapter 5

File Documentation

5.1 ButtonDebounce.h File Reference

Register Button Presses with Software Debounce.

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

Classes

- class [ButtonDebounce](#)

Macros

- `#define EXECUTENUMBERALL 3`
Button Class using software debounce.
- `#define EXECUTENUMBERISPRESSED 2`
- `#define EXECUTENUMBERSTILLPRESSED 1`
- `#define EXECUTENUMBERNONE 0`

5.1.1 Detailed Description

Register Button Presses with Software Debounce.

Author

Timo Raab

Version

1.2

Date

2023-12-23

Copyright

Copyright (c) 2023

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-elliott-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 #define EXECUTENUMBERALL 3
00031 #define EXECUTENUMBERISPRESSED 2
00032 #define EXECUTENUMBERSTILLPRESSED 1
00033 #define EXECUTENUMBERNONE 0
00034
00035 class ButtonDebounce {
00036
00037     private:
00038         unsigned char _pin;           // Pin
00039         bool _pullUp;                // Use of internal pull up resistor,
00040                                     // if not pull down is assumend,
00041                                     // standard: true
00042
00043         //Standard Operation
00044         bool _executeAtRelease;       // Choose, when the button press should be registered
00045                                     // Standard: at buttonPress (_executeAtRelease = false)
00046         void (*_bFunc)();             // function call at button press
00047
00048         //Long Press Operation, only available with setter-functions
00049         //Long Press is only available with execute at start
00050         unsigned long _longPressDuration; // Duration till long press triggers
00051         void (*_bFuncLong)();         // function call at long press activation
00052
00053         //Internal handling
00054         uint8_t _buttonHistory;       // saves history for debounce
00055         bool _isPressedTemp;          // for longPress needed
00056         unsigned long _pressTimeTemp; // time when button is pressed for longPress
00057
00058         uint8_t readButton();          // read current button status
00059
00060
00061     public:
00062         ButtonDebounce(unsigned char pin, bool pullUp = true, bool executeAtRelease = false, void
00074         (*bFunction)() = nullptr) ;
00075
00085         bool setPullUp(bool pullUp);
00086         bool setExecuteAtRelease(bool executeAtRelease);
00087         bool setFunction(void (*bFunction)());
00089         bool setLongPressDuration(unsigned long duration);
00091         bool setLongPressFunction(void (*bFunction)());
00093
00099         bool isPressed(bool execute = false);
00100
00101
00107         bool isLongPressed(bool execute = false);
00108

```



```
00116         bool stillPressed(bool execute = false);
00117
00125         bool anyPressed(bool execute = false);
00126
00134         bool anyPressed(uint8_t executeNumber = EXECUTENUMBERNONE);
00135
00142         void updateButton();
00143
00149         uint8_t getButtonHistory();
00150     };
00151
00152 #endif
00153
00154 //EOF
```


Index

- anyPressed
 - ButtonDebounce, [8](#)
- ButtonDebounce, [1](#), [7](#)
 - anyPressed, [8](#)
 - ButtonDebounce, [8](#)
 - getButtonHistory, [9](#)
 - isLongPressed, [9](#)
 - isPressed, [9](#)
 - setLongPressDuration, [10](#)
 - setLongPressFunction, [10](#)
 - stillPressed, [10](#)
 - updateButton, [10](#)
- ButtonDebounce.h, [11](#)
- getButtonHistory
 - ButtonDebounce, [9](#)
- isLongPressed
 - ButtonDebounce, [9](#)
- isPressed
 - ButtonDebounce, [9](#)
- setLongPressDuration
 - ButtonDebounce, [10](#)
- setLongPressFunction
 - ButtonDebounce, [10](#)
- stillPressed
 - ButtonDebounce, [10](#)
- updateButton
 - ButtonDebounce, [10](#)