# Software Timers

1.4

Generated by Doxygen 1.8.8

Wed Sep 14 2016 14:26:50

# Contents



# 1 Software Timers Manual

## 1.1 Features

- Provide a simple, quick to use timer object.
- No hardware timer or interrupt is used.

- Less code to type, and shorter programs because of not having to pepper comparisons with millis() everywhere.

Developed and tested with Arduino IDE 1.6.10. May not run on Arduino 0.X.

### 1.2 Author

Volker Kuhlmann
http://volker.top.geek.nz/contact.html

### 1.3 Download

http://volker.top.geek.nz/arduino/

### 1.4 Copyright



### 1.5 TODO

- Empty

### 1.6 History

- v1.4, 13 Sep 2016 Volker Kuhlmann
  - Released with doxygen comments and documentation.
  - GPLv3

## 2 Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# 3 Class Index

## 3.1 Class List

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



# 4 File Index

## 4.1 File List

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



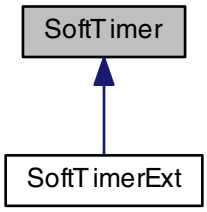
# 5 Class Documentation

## 5.1 SoftTimer Class Reference

Software timer object.

`#include <SoftTimer.h>`

Inheritance diagram for SoftTimer:




### Public Member Functions

- bool is_expired ()

*Return true if the timer has expired.*

- void start (uint16_t delay_ms)

  *Start timer with up to 65535 ms.*

- void startW (unsigned long delay_ms)

  *Start timer.*

- void startN (uint8_t delay_ms)

  *Start timer with up to 255 ms.*

- bool interval (uint16_t delay_ms)

  *Return true when timer expires, and restart. 65535 ms max.*

- bool intervalW (unsigned long delay_ms)

  *Return true when timer expires, and restart.*

- bool intervalN (uint8_t delay_ms)

  *Return true when timer expires, and restart. 255 ms max.*

#### Protected Attributes

- unsigned long _finish_ms

  *Expiry time, in milliseconds.*

### 5.1.1 Detailed Description

Software timer object.

Simple timer that does not use interrupts. It's easier to use and produces shorter code than using 4-byte comparisons with millis() everywhere.

### 5.1.2 Member Function Documentation

#### 5.1.2.1 bool SoftTimer::interval ( uint16_t *delay_ms* )

Return true when timer expires, and restart. 65535 ms max.

is_expired() and start() combined. Because the timer is restarted, true is returned only once (unless the timer has again expired before the next call).

```
SoftTimer timer;
if (timer.interval(1000)) {
  // Do something once per second
}
```

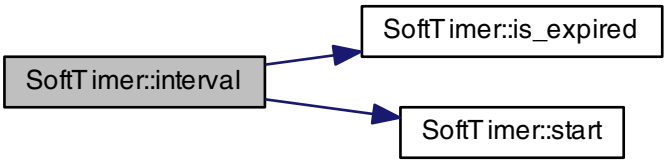
#### Parameters

| | |
|---|---|
| *delay_ms* | uint16_t Timer delay in milliseconds. |

**Returns**

bool True if timer has expired.

Here is the call graph for this function:




#### 5.1.2.2 bool SoftTimer::intervalN ( uint8_t *delay_ms* )

Return true when timer expires, and restart. 255 ms max.

is_expired() and startN() combined. See interval().

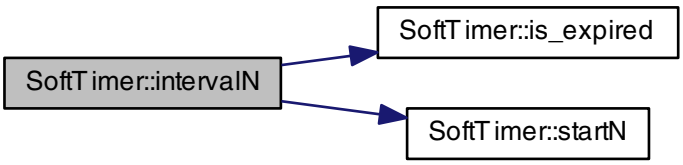
**Parameters**

| | |
|---|---|
| *delay_ms* | uint8_t Timer delay in milliseconds. |

**Returns**

bool True if timer has expired.

Here is the call graph for this function:




#### 5.1.2.3 bool SoftTimer::intervalW ( unsigned long *delay_ms* )

Return true when timer expires, and restart.

is_expired() and startW() combined. See interval().

Uses unsigned long instead of uint32_t because that is what millis() uses.

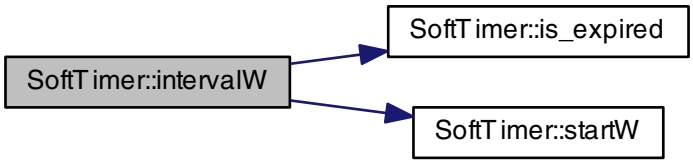
**Parameters**

| | |
|---|---|
| *delay_ms* | unsigned long Timer delay in milliseconds. |

**Returns**

bool True if timer has expired.

Here is the call graph for this function:




Here is the caller graph for this function:


#### 5.1.2.4 bool SoftTimer::is_expired ( )

Return true if the timer has expired.

Compares with the current value of millis().

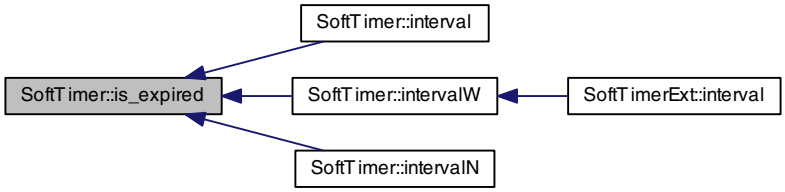
**Note**

The Arduino millisecond counter overflows after 49.7 days. This function fails if the timer's start and finish times cross the overflow.

**Returns**

bool True if timer has expired. Return true for as long as the timer is in the expired state.

Here is the caller graph for this function:




#### 5.1.2.5 void SoftTimer::start ( uint16_t *delay_ms* )

Start timer with up to 65535 ms.

Start the timer with a 16 bit (65535 ms max) delay from now.

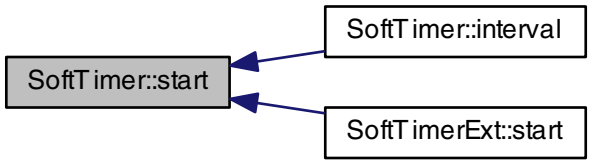
**Parameters**

| | |
|---|---|
| *delay_ms* | uint16_t Timer delay in milliseconds. |

**Returns**

void

Here is the caller graph for this function:




#### 5.1.2.6 void SoftTimer::startN ( uint8_t *delay_ms* )

Start timer with up to 255 ms.

Start the timer with an 8 bit (narrow, 255 ms max) delay from now.

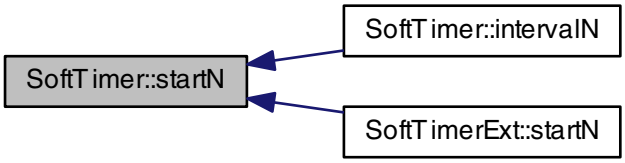
**Parameters**

| | |
|---|---|
| *delay_ms* | uint8_t Timer delay in milliseconds. |

**Returns**

void

Here is the caller graph for this function:




#### 5.1.2.7 void SoftTimer::startW ( unsigned long *delay_ms* )

Start timer.

Start the timer with a 32 bit (wide) delay from now.

Uses unsigned long instead of uint32_t because that is what millis() uses.

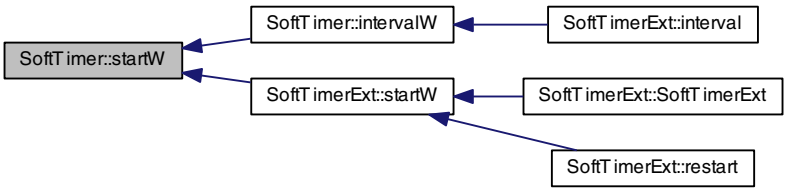
**Parameters**

| | |
|---|---|
| *delay_ms* | unsigned long Timer delay in milliseconds. |

**Returns**

void

Here is the caller graph for this function:




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

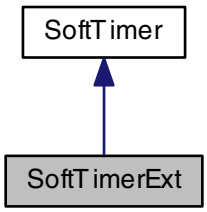
- SoftTimer.h
- SoftTimer.cpp

## 5.2 SoftTimerExt Class Reference

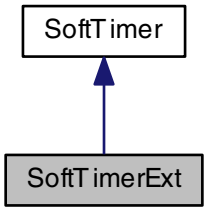
Software timer object with extended features.

`#include <SoftTimer.h>`

Inheritance diagram for SoftTimerExt:




Collaboration diagram for SoftTimerExt:




### Public Member Functions

- SoftTimerExt ()
  *Default constructor, does nothing.*
- SoftTimerExt (unsigned long delay_ms)
  *Save the timer interval without starting timer.*
- void restart ()
  *Restart timer (that was previously started).*
- bool interval ()
  *Return true when timer expires, and restart.*
- uint32_t remaining ()
  *Return remaining timer milliseconds.*
- uint32_t remaining_s ()
  *Return remaining timer seconds.*
- uint32_t elapsed ()
  *Return elapsed milliseconds since (re-)start.*

- uint32_t elapsed_s ()

  *Return elapsed seconds since (re-)start.*
- bool is_new_s ()

  *Return true if a new second has started since last call.*
- void start (uint16_t delay_ms)

  *Same as for SoftTimer.*
- void startW (unsigned long delay_ms)

  *Same as for SoftTimer.*
- void startN (uint8_t delay_ms)

  *Same as for SoftTimer.*

#### Protected Attributes

- unsigned long _length_ms

  *Timer interval in milliseconds.*

#### Private Attributes

- uint8_t _last_s

  *Hold the seconds when is_new_s() was last called.*

### 5.2.1 Detailed Description

Software timer object with extended features.

The timer can be restarted without having to give the interval again. Elapsed and remaining times are availble. Uses more memory.

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 SoftTimerExt::SoftTimerExt ( )

Default constructor, does nothing.

Empty constructor, but required.

#### 5.2.2.2 SoftTimerExt::SoftTimerExt ( unsigned long *delay_ms* )

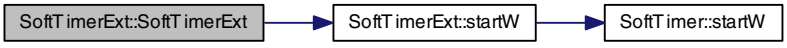
Save the timer interval without starting timer.

Constructor that stores the timer delay (in milliseconds). The timer is not started.

**Parameters**

| | |
|---|---|
| *delay_ms* | unsigned long Timer delay in milliseconds. |

Here is the call graph for this function:



### 5.2.3 Member Function Documentation

#### 5.2.3.1 uint32_t SoftTimerExt::elapsed ( )

Return elapsed milliseconds since (re-)start.

**Returns**

uint32_t Number of milliseconds since the timer was started.

Here is the caller graph for this function:


#### 5.2.3.2 uint32_t SoftTimerExt::elapsed_s ( )

Return elapsed seconds since (re-)start.

Uses integer division.

**Returns**

uint32_t Number of seconds since the timer was started.

Here is the call graph for this function:


#### 5.2.3.3 bool SoftTimerExt::interval ( )

Return true when timer expires, and restart.

Check if the timer has expired, and restart it if it has. Similar to the SoftTimer interval functions, but start(), startW(), or startN() must have been called at least once before, as for restart().

**Returns**

bool True if timer has expired.

Here is the call graph for this function:


#### 5.2.3.4 bool SoftTimerExt::is_new_s ( )

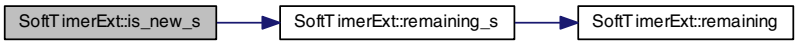
Return true if a new second has started since last call.

Returns true once for each second that has elapsed since the timer was started. This can be used e.g. to show elapsed time on a display, updating every second.

**Returns**

bool True if another second of the timer delay has elapsed.

Here is the call graph for this function:



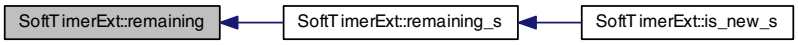

#### 5.2.3.5 uint32_t SoftTimerExt::remaining ( )

Return remaining timer milliseconds.

**Returns**

uint32_t Number of milliseconds before the timer expires.

Here is the caller graph for this function:




#### 5.2.3.6 uint32_t SoftTimerExt::remaining_s ( )

Return remaining timer seconds.

This returns ceil() of the remaining seconds. I.e. 0s returns 0, 0.001 .. 1.000s returns 1, 1.001s .. 2.000s returns 2.

Uses integer division.

**Returns**

uint32_t Number of seconds before the timer expires.

Here is the call graph for this function:


Here is the caller graph for this function:


#### 5.2.3.7 void SoftTimerExt::restart ( )

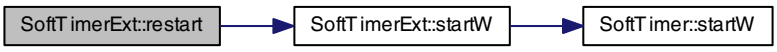
Restart timer (that was previously started).

This function only works correctly if the timer interval has been stored before, by calling start(), startW(), or startN() at least once before, or using the constructor with delay parameter. The start-functions can be called any time to change the timer delay for subsequent calls to restart().

**Returns**

void

Here is the call graph for this function:




#### 5.2.3.8 void SoftTimerExt::start ( uint16_t *delay_ms* )

Same as for SoftTimer.

Re-implementation, to also store the timer delay for use later.

**Parameters**

| | |
|---|---|
| *delay_ms* | uint16_t Timer delay in milliseconds. |

**Returns**

void

Here is the call graph for this function:


#### 5.2.3.9 void SoftTimerExt::startN ( uint8_t *delay_ms* )

Same as for SoftTimer.

Re-implementation, to also store the timer delay for use later.

**Parameters**

| | |
|---|---|
| *delay_ms* | uint8_t Timer delay in milliseconds. |

**Returns**

void

Here is the call graph for this function:


#### 5.2.3.10 void SoftTimerExt::startW ( unsigned long *delay_ms* )

Same as for SoftTimer.

Re-implementation, to also store the timer delay for use later.

**Parameters**

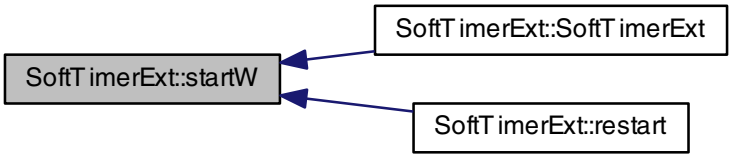
| | |
|---|---|
| *delay_ms* | unsigned long Timer delay in milliseconds. |

**Returns**

void

Here is the call graph for this function:


Here is the caller graph for this function:




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

- SoftTimer.h
- SoftTimer.cpp

# 6 File Documentation

## 6.1 mainpage_doxygen.h File Reference

Content of the index page for the doxygen documentation.

### 6.1.1 Detailed Description

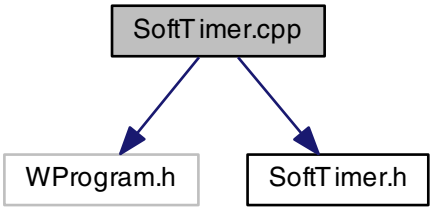
Content of the index page for the doxygen documentation.

## 6.2 SoftTimer.cpp File Reference

SoftTimer - Arduino library for simple timers.

```
#include <WProgram.h>
#include "SoftTimer.h"
```

Include dependency graph for SoftTimer.cpp:




#### 6.2.1 Detailed Description

SoftTimer - Arduino library for simple timers.

**Description**
Provide a simple, quick to use timer object. No hardware timer or interrupt is used. Less code to type, and shorter programs because of not having to pepper comparisons with millis() everywhere.

Developed with Arduino IDE 1.6.10. May not run on Arduino 0.X.

**TODO**

**Author**

Volker Kuhlmann
http://volker.top.geek.nz/contact.html

**Download**

http://volker.top.geek.nz/arduino/

**Copyright**



**History**

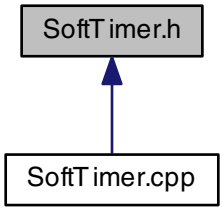
- v1.0, 24 Aug 2016 Volker Kuhlmann
  - Initial version. Based on code in a project from 19 Aug 2016.
- v1.1, 25 Aug 2016 Volker Kuhlmann
  - Add missing intervalS(). Change unsigned short to explicit uint8_t.
  - Add example.
- v1.2, 05 Sep 2016 Volker Kuhlmann
  - Rename from SimpleTimer to SoftTimer.

  - Rename startS/intervalS to startN/intervalN, for Narrow / Wide.
  - Add SoftTimerExt.
- v1.3, 09 Sep 2016 Volker Kuhlmann
  - Add interval().
- v1.4, 13 Sep 2016 Volker Kuhlmann
  - Doxygen comments.

## 6.3 SoftTimer.h File Reference

SoftTimer - Arduino library for simple software timers.

This graph shows which files directly or indirectly include this file:




### Classes

- class SoftTimer

  *Software timer object.*
- class SoftTimerExt

  *Software timer object with extended features.*

### 6.3.1 Detailed Description

SoftTimer - Arduino library for simple software timers.

**Description**
Provide a simple, quick to use timer object. No hardware timer or interrupt is used. Less code to type, and shorter programs because of not having to pepper comparisons with millis() everywhere.

Developed with Arduino IDE 1.6.10. May not run on Arduino 0.X.

### TODO

### Author

Volker Kuhlmann
http://volker.top.geek.nz/contact.html

### Download

http://volker.top.geek.nz/arduino/

### Copyright



### History

- v1.0, 24 Aug 2016 Volker Kuhlmann
  - Initial version. Based on code in a project from 19 Aug 2016.
- v1.1, 25 Aug 2016 Volker Kuhlmann
  - Add missing intervalS(). Change unsigned short to explicit uint8_t.
  - Add example.
- v1.2, 05 Sep 2016 Volker Kuhlmann
  - Rename from SimpleTimer to SoftTimer.
  - Rename startS/intervalS to startN/intervalN, for Narrow / Wide.
  - Add SoftTimerExt.
- v1.3, 09 Sep 2016 Volker Kuhlmann
  - Add interval().
- v1.4, 13 Sep 2016 Volker Kuhlmann
  - Doxygen comments.