

TimerClass

1.0

Generated by Doxygen 1.8.17

Thu Mar 19 2020 18:59:10

1 Modules Index	1
1 Modules Index	1
1.1 Modules List	1
2 Data Type Index	2
2.1 Data Types List	2
3 File Index	2
3.1 File List	2
4 Module Documentation	2
4.1 timeclass Module Reference	2
4.1.1 Detailed Description	3
4.1.2 Function/Subroutine Documentation	4
4.2 timerclass Module Reference	10
4.2.1 Detailed Description	11
4.2.2 Function/Subroutine Documentation	12
5 Data Type Documentation	18
5.1 timeclass::time Module Reference	18
5.1.1 Detailed Description	19
5.1.2 Member Function/Subroutine Documentation	19
5.1.3 Member Data Documentation	28
5.2 timerclass::ttimer Module Reference	28
5.2.1 Detailed Description	30
5.2.2 Member Function/Subroutine Documentation	30
5.2.3 Member Data Documentation	37
6 File Documentation	38
6.1 TimeClass.f03 File Reference	38
6.2 TimeClass.f03	39
6.3 TimerClass.f03 File Reference	42
6.4 TimerClass.f03	43
Index	49

1 Modules Index

1.1 Modules List

Here is a list of all modules with brief descriptions:

timeclass	
The TimeClass module contains the TTime class used by the TTimer class for practical timing	2
timerclass	
Module containing the TTimer class for practical timing	10

2 Data Type Index

2.1 Data Types List

Here are the data types with brief descriptions:

timeclass::ttime	
The TTime class contains all time functionality with regard to a single time stamp	18
timerclass::ttimer	
A class for keeping track of time in calculations, by creating a list of timestamps	28

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

TimeClass.f03	38
TimerClass.f03	42

4 Module Documentation

4.1 timeclass Module Reference

The **TimeClass** module contains the [TTime](#) class used by the [TTimer](#) class for practical timing.

Data Types

- module [time](#)
The TTime class contains all time functionality with regard to a single time stamp.

Functions/Subroutines

- pure type([ttime](#)) function [constructor](#) ()
Constructor for the TTime class.
- subroutine [settime](#) (this)
Function to set the TTime instance to the current time, with millisecond resolution.
- pure subroutine [calculatejdn](#) (this)
Calculating the Julian Day number based on the Gregorian date set in the TTime object.
- pure subroutine [setjdn](#) (this, JDN, IO)
Set the Julian Day Number of a TTime instance manually.
- pure integer(kind=4) function [getjuliandaynumber](#) (this)
Function returning the Julian Day Number as a 4-byte integer.
- pure subroutine [setgregoriandatefromjdn](#) (this, IO)

Subroutine which transforms the set Julian Day Number into a Gregorian Calender date.

- pure subroutine `copy` (this, from)

Function to copy one `TTime` instance to the current one via the "=" assignment. This is the function performing the actual operator overloading.

- pure type(`ttime`) function `add` (this, that)

Function to add two `TTime` instance via the "+" operator. This is the function performing the actual operator overloading.

- pure type(`ttime`) function `subtract` (this, that)

Function to subtract two `TTime` instance via the "-" operator. This is the function performing the actual operator overloading.

- pure logical function `isleapyear` (this)

Function returning true/false if the year of the `TTime` instance is a leap year.

- pure character(len=255) function `gettimestring` (this, fmt)

Returns a string with the time as a string.

- pure real(dp) function `gettimeseconds` (this)

Function returning the time in (fractional) seconds (double precision).

- subroutine `destructor` (this)

Destructor of the `TTime` object instance.

4.1.1 Detailed Description

The **TimeClass module** contains the `TTime class` used by the `TTimer class` for practical timing.

Author

Dr. Dr. Danny E. P. Vanpoucke

Version

2.0-3 (upgrades deprecated timing module)

Date

19-03-2020

Copyright

<https://dannyvanpoucke.be>

Warning

Internally, Julian Day Numbers are used to compare dates. As a result, *negative* dates are not accepted. If such dates are created (e.g., due to a subtraction), then the date is set to zero.

This module makes use of:

- nothing; this module is fully independent

4.1.2 Function/Subroutine Documentation

4.1.2.1 add() `pure type(ttime) function timeclass::add (`
 `class(ttime), intent(in) this,`
 `class(ttime), intent(in) that) [private]`

Function to add two [TTime](#) instance via the "+" operator. This is the function performing the actual operator overloading.

Adding two full dates is maybe a bit strange to do. In our case, we don't just add the days and add the months, but we add the days of the year (via their Julian Day Numbers) and transform these back to months and days. (why keep life simple if we can make it complicated?)

usage:

`total = this + that`

This line also calls the [assignment operator](#).

Parameters

in	<i>this</i>	The TTime instance before the "+" operator.
in	<i>that</i>	The TTime instance after the "+" operator.

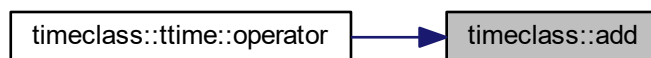
Returns

Total The [TTime](#) instance representing the sum.

Definition at line [264](#) of file [TimeClass.f03](#).

Referenced by [timeclass::time::operator\(\)](#).

Here is the caller graph for this function:



4.1.2.2 calculatejdn() `pure subroutine timeclass::calculatejdn (`
 `class(ttime), intent(inout) this) [private]`

Calculating the Julian Day number based on the Gregorian date set in the [TTime](#) object.

In practice the Gregorian calendar date set in the [TTime](#) instance is transformed into a Julian Day Number, and stored in the instance as [time::jdn](#).

Parameters

<code>in, out</code>	<code>this</code>	The TTime instance being called.
----------------------	-------------------	--------------------------------------------------

Definition at line 139 of file [TimeClass.f03](#).

4.1.2.3 constructor() `pure type(ttime) function timeclass::constructor [private]`

Constructor for the [TTime](#) class.

Note

This constructor does not set the time. It just enters zero's.

usage:

```
Type(TTime) :: T
t = ttime()
```

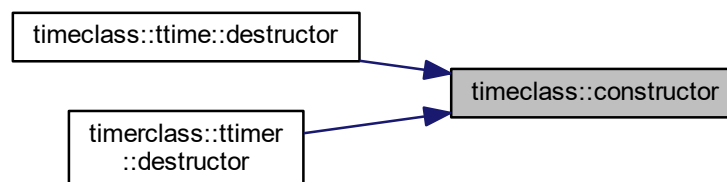
Returns

Time An instance of the TTime class.

Definition at line 98 of file [TimeClass.f03](#).

Referenced by [timeclass::time::destructor\(\)](#), and [timerclass::timer::destructor\(\)](#).

Here is the caller graph for this function:



4.1.2.4 copy() `pure subroutine timeclass::copy (class(ttime), intent(inout) this, class(ttime), intent(in) from) [private]`

Function to copy one [TTime](#) instance to the current one via the "=" assignment. This is the function performing the actual operator overloading.

usage:

```
tnew = told
```

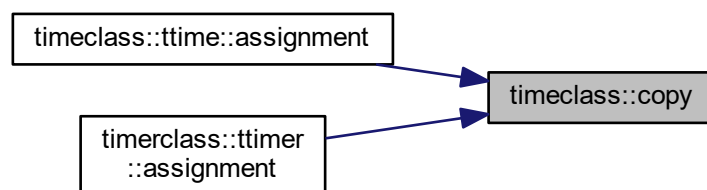
Parameters

in, out	<i>this</i>	The TTime instance before the "=" assignment.
in	<i>from</i>	The TTime instance after the "=" assignment.

Definition at line 233 of file [TimeClass.f03](#).

Referenced by [timeclass::time::assignment\(\)](#), and [timerclass::timer::assignment\(\)](#).

Here is the caller graph for this function:



4.1.2.5 destructor() `subroutine timeclass::destructor (`
`type(ttime) this) [private]`

Destructor of the [TTime](#) object instance.

This subroutine is automatically called upon finalization of the instance.

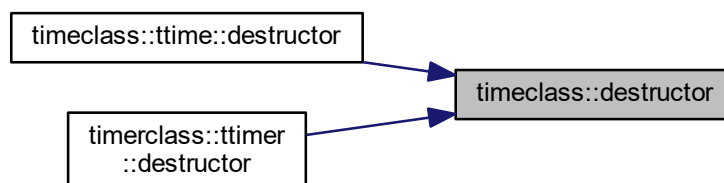
Parameters

in, out	<i>this</i>	The instance of the TTime class in need of destruction.
---------	-------------	-------------------------------------------------------------------------

Definition at line 419 of file [TimeClass.f03](#).

Referenced by [timeclass::time::destructor\(\)](#), and [timerclass::timer::destructor\(\)](#).

Here is the caller graph for this function:



4.1.2.6 getjuliandaynumber() `pure integer(kind=4) function timeclass::getjuliandaynumber (class(ttime), intent(in) this) [private]`

Function returning the Julian Day Number as a 4-byte integer.

Use `integer(kind=4)`.

Parameters

<code>in, out</code>	<code>this</code>	The TTime instance being called.
----------------------	-------------------	--------------------------------------------------

Returns

JDN The integer Julian Day Number.

Definition at line [184](#) of file [TimeClass.f03](#).

4.1.2.7 gettimeseconds() `pure real(dp) function timeclass::gettimeseconds (class(ttime), intent(in) this) [private]`

Function returning the time in (fractional) seconds (double precision).

Parameters

<code>in</code>	<code>this</code>	The TTime instance.
-----------------	-------------------	-------------------------------------

Returns

sec total number of seconds representing the "time" as a double precision value.

Definition at line [402](#) of file [TimeClass.f03](#).


```
4.1.2.8 gettimeofday() pure character(len=255) function timeclass::gettimeofday (
    class(ttime), intent(in) this,
    character(len=*), intent(in), optional fmt ) [private]
```

Returns a string with the time as a string.

Format options for *fmt*:

- full: dd/mm/yyyy hh:mm:ss.mmm
- date: dd/mm/yyyy
- time: hh:mm:ss.mmm
- days: Gives the total time in fractional days (best used for time-differences). Uses the Julian Day Number.
- hours: Same as days, but transformed to hours.
- seconds: Same as days, but transformed to seconds.

Parameters

in	<i>this</i>	The TTime instance transform into a string.
in	<i>fmt</i>	String representing the possible formatting. [OPTIONAL , DEFAULT = full]

Returns

TS String with formatted time.

Definition at line [356](#) of file [TimeClass.f03](#).

```
4.1.2.9 isleapyear() pure logical function timeclass::isleapyear (
    class(ttime), intent(in) this ) [private]
```

Function returning true/false if the year of the [TTime](#) instance is a leap year.

A leap year is a multiple of 4, but not 100, unless 400

Parameters

in	<i>this</i>	The TTime instance to check the leap-year.
----	-------------	------------------------------------------------------------

Returns

Leap Boolean indicating if the year is a leap-year.

Definition at line [327](#) of file [TimeClass.f03](#).

4.1.2.10 setgregoriandatefromjdn() `pure subroutine timeclass::setgregoriandatefromjdn (`
`class(ttime), intent(inout) this,`
`integer, intent(out), optional IO) [private]`

Subroutine which transforms the set Julian Day Number into a Gregorian Calendar date.

Note

The routine is only valid for a JDN \geq 0.

Parameters

in, out	<i>this</i>	The TTime instance being called.
out	<i>IO</i>	Returns 0 on success, and -1 for failure. [OPTIONAL]

Definition at line 199 of file [TimeClass.f03](#).

4.1.2.11 setjdn() `pure subroutine timeclass::setjdn (`
`class(ttime), intent(inout) this,`
`integer(kind=4), intent(in) JDN,`
`integer, intent(out), optional IO) [private]`

Set the Julian Day Number of a [TTime](#) instance manually.

Note

The Julian Day Number should be \geq 0. For negative values it is set to 0, and an error value is set to IO

Parameters

in, out	<i>this</i>	The TTime instance being called.
in	<i>JDN</i>	A positive integer(kind=4) value representing a valid Julian Day Number.
out	<i>IO</i>	Integer value returning 0 upon success, and a negative value(=JDN) in case of failure.

Definition at line 158 of file [TimeClass.f03](#).

4.1.2.12 settime() `subroutine timeclass::settime (`
`class(ttime), intent(inout) this) [private]`

Function to set the [TTime](#) instance to the current time, with millisecond resolution.

Note

As this subroutine uses the `date_and_time` intrinsic it is an *impure* subroutine.

Parameters

in, out	<i>this</i>	The TTime instance being called.
---------	-------------	--------------------------------------------------

Definition at line [116](#) of file [TimeClass.f03](#).

4.1.2.13 subtract() `pure type(ttime) function timeclass::subtract (`
`class(ttime), intent(in) this,`
`class(ttime), intent(in) that) [private]`

Function to subtract two [TTime](#) instance via the "-" operator. This is the function performing the actual operator overloading.

usage:

```
total = this - that
```

This line also calls the [assignment operator](#).

Note

The result should remain a positive number.

Parameters

in	<i>this</i>	The TTime instance before the "-" operator.
in	<i>that</i>	The TTime instance after the "-" operator.

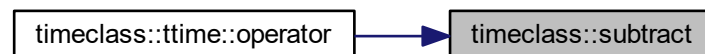
Returns

Total The [TTime](#) instance representing the difference.

Definition at line [300](#) of file [TimeClass.f03](#).

Referenced by [timeclass::time::operator\(\)](#).

Here is the caller graph for this function:

**4.2 timerclass Module Reference**

Module containing the [TTimer class](#) for practical timing.

Data Types

- module [timer](#)

A class for keeping track of time in calculations, by creating a list of timestamps.

Functions/Subroutines

- pure type([ttimer](#)) function [constructor](#) ()
Constructor for the [TTimer](#) instances.
- integer function [start](#) (this)
Start the [TTimer](#) instance (clean start, everything is reset). If the timer was already running it is reset first.
- integer function [resume](#) (this)
Restart the [TTimer](#) instance after a pause. If the timer is not paused, nothing will happen and TS=-1 is returned.
- integer function [addtimeflag](#) (this)
Add an additional timestamp without changing the status (running/paused) of the [TTimer](#) instance.
- integer function [interrupt](#) (this, IO)
Puts the [TTimer](#) instance on hold.
- integer function [stoptimer](#) (this, IO)
Stops the [TTimer](#) instance (terminal fashion...no restart possible).
- pure subroutine [reset](#) (this)
Start the [TTimer](#) instance. If the timer was already running it is reset first.
- integer function [addtimestamp](#) (this)
Add a timestamp to a running [TTimer](#) instance, returning the index of the timestamp.
- pure real(dp) function [getelapsedtime_total](#) (this, INCL_PAUSE)
Returns the number of seconds which elapsed between the start and stop timestamps.
- pure real(dp) function [getelapsedtime_steps](#) (this, Tstart, Tend, INCL_PAUSE)
Returns the number of seconds which elapsed between two timestamps. This is always a positive value.
- pure character(len=50) function [getelapsedtimestring](#) (this, TSTART, TSTOP, INCL_PAUSE, FMT)
Returns a string representing the elapsed time. +.
- subroutine [printelapsedtimereport](#) (this, message, Tstart, Tstop, UN, INCL_PAUSE)
Print small timings report to unit UN.
- pure subroutine [copy](#) (this, from)
Function to copy one [TTimer](#) instance to the current one via the "=" assignment.
- subroutine [destructor](#) (this)
Destructor of the [TTimer](#) class. Cleans up the instance upon finalization.

4.2.1 Detailed Description

Module containing the [TTimer](#) class for practical timing.

Author

Dr. Dr. Danny E. P. Vanpoucke

Version

2.0-3 (upgrades deprecated timing module)

Date

19-03-2020

Copyright

<https://dannyvanpoucke.be>

This module makes use of:

- [TimeClass](#)

4.2.2 Function/Subroutine Documentation

4.2.2.1 addtimeflag() `integer function timerclass::addtimeflag (`
`class(ttimer), intent(inout) this) [private]`

Add an additional timestamp without changing the status (running/paused) of the [TTimer](#) instance.

If the timer is stopped nothing will happen, and -1 is returned.

Parameters

<code>in, out</code>	<code>this</code>	The TTimer instance.
----------------------	-------------------	--------------------------------------

Returns

TS The index of the timestamp.

Definition at line [165](#) of file [TimerClass.f03](#).

4.2.2.2 addtimestamp() `integer function timerclass::addtimestamp (`
`class(ttimer), intent(inout) this) [private]`

Add a timestamp to a running [TTimer](#) instance, returning the index of the timestamp.

In case the timer is not running, nothing is done and -1 is returned.

Parameters

<code>in, out</code>	<code>this</code>	The TTimer instance.
----------------------	-------------------	--------------------------------------

Returns

TS The index of the timestamp.

Definition at line [279](#) of file [TimerClass.f03](#).

4.2.2.3 constructor() `pure type(ttimer) function timerclass::constructor [private]`

Constructor for the [TTimer](#) instances.

Usage:

```
Type(TTimer) :: T
t=ttimer()
```

Returns

Returns a [TTimer](#) object

Definition at line [104](#) of file [TimerClass.f03](#).

4.2.2.4 copy() `pure subroutine timerclass::copy (`
`class(ttimer), intent(inout) this,`
`class(ttimer), intent(in) from) [private]`

Function to copy one [TTimer](#) instance to the current one via the "=" assignment.

Parameters

in, out	<i>this</i>	The TTimer instance before the "=" assignment.
in	<i>from</i>	The TTimer instance after the "=" assignment.

Definition at line [504](#) of file [TimerClass.f03](#).

4.2.2.5 destructor() `subroutine timerclass::destructor (`
`type(ttimer) this) [private]`

Destructor of the [TTimer](#) class. Cleans up the instance upon finalization.

Parameters

in, out	<i>this</i>	The TTimer instance being destroyed automatically.
---------	-------------	--------------------------------------------------------------------

Definition at line [529](#) of file [TimerClass.f03](#).

4.2.2.6 getelapsedtime_steps() `pure real(dp) function timerclass::getelapsedtime_steps (`
`class(ttimer), intent(in) this,`
`integer, intent(in) Tstart,`
`integer, intent(in) Tend,`
`logical, intent(in), optional INCL_PAUSE) [private]`

Returns the number of seconds which elapsed between two timestamps. This is always a positive value.

NOTE:

- If the timestamps are out of range, then -1 is returned.
- If the start comes after end, then they are exchanged such that a positive time is obtained.

Parameters

in	<i>this</i>	The TTimer instance.
in	<i>Tstart, Tend</i>	The indices of the start and end time.
in	<i>INCL_PAUSE</i>	Logical indicating if the time during which the timer was paused should be included as well. [OPTIONAL, DEFAULT = .false.]

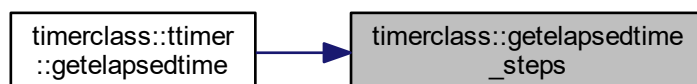
Returns

Seconds The (fractional) number of seconds elapsed, as double precision real.

Definition at line 348 of file [TimerClass.f03](#).

Referenced by [timerclass::timer::getelapsedtime\(\)](#).

Here is the caller graph for this function:



4.2.2.7 getelapsedtime_total() `pure real(dp) function timerclass::getelapsedtime_total (`
`class(ttimer), intent(in) this,`
`logical, intent(in), optional INCL_PAUSE) [private]`

Returns the number of seconds which elapsed between the start and stop timestamps.

Parameters

in	<i>this</i>	The TTimer instance.
in	<i>INCL_PAUSE</i>	Logical indicating if the time during which the timer was paused should be included as well. [OPTIONAL , DEFAULT = .false.]

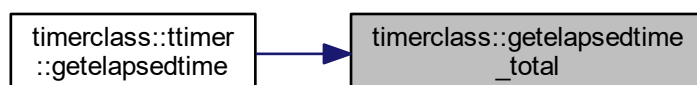
Returns

Seconds The (fractional) number of seconds elapsed, as double precision real.

Definition at line 320 of file [TimerClass.f03](#).

Referenced by [timerclass::timer::getelapsedtime\(\)](#).

Here is the caller graph for this function:



4.2.2.8 getelapsedtimestring() `pure character(len=50) function timerclass::getelapsedtimestring (`

```

    class(ttimer), intent(in) this,
    integer, intent(in), optional TSTART,
    integer, intent(in), optional TSTOP,
    logical, intent(in), optional INCL_PAUSE,
    character(len=*), intent(in), optional FMT ) [private]

```

Returns a string representing the elapsed time. +.

Parameters

in	<i>this</i>	The TTimer instance.
in	<i>TSTART, TSTOP</i>	Indices of the start and end-times. [OPTIONAL, DEFAULT: TSTART=1, TSTOP=index of last timestamp]
in	<i>INCL_PAUSE</i>	Logical indicating if the time during which the timer was paused should be included as well. [OPTIONAL, DEFAULT = .false.]
in	<i>FMT</i>	string indicating the format for the time-string. <ul style="list-style-type: none"> • sec : xxx.xxx secs • hms : xxxx h xx min xx.xxx secs • dhms: xx days xx h xx min xx.xxx secs • hour: xxx.xxx hours • day : xxx.xxx days [OPTIONAL, DEFAULT= sec]

Returns

str The string with the formatted time.

Definition at line 402 of file [TimerClass.f03](#).

4.2.2.9 interrupt() `integer function timerclass::interrupt (`

```

    class(ttimer), intent(inout) this,
    integer, intent(out), optional IO ) [private]

```

Puts the [TTimer](#) instance on hold.

If the timer was not running nothing will happen. The optional IO parameter will return an error code.

IO values:

- 0 : all is well
- -1 : The timer was not running, so nothing to pause.
- -2 : The timer was already stopped/paused, so nothing to pause.

In case of error, TS will be set to -1.

Parameters

in, out	<i>this</i>	The TTimer instance.
out	<i>IO</i>	Optional parameter giving the error-status. [OPTIONAL ; DEFAULT= 0]

Returns

TS The index of the final timestamp.

Definition at line 193 of file [TimerClass.f03](#).

4.2.2.10 printelapsedtimereport() subroutine timerclass::printelapsedtimereport (

```

    class(ttimer), intent(inout) this,
    character(len=*), intent(in) message,
    integer, intent(in) Tstart,
    integer, intent(in) Tstop,
    integer, intent(in) UN,
    logical, intent(in), optional INCL_PAUSE ) [private]

```

Print small timings report to unit UN.

Parameters

in	<i>this</i>	The TTimer instance.
in	<i>message</i>	String containing a message to include.
in	<i>Tstart, Tstop</i>	Integer indexes of the start and stop times
in	<i>UN</i>	Integer indicating the unit to write the report to.
in	<i>INCL_PAUSE</i>	Logical indicating if the time during which the timer was paused should be included as well. [OPTIONAL, DEFAULT = .false.]

Definition at line 470 of file [TimerClass.f03](#).

4.2.2.11 reset() pure subroutine timerclass::reset (

```

    class(ttimer), intent(inout) this ) [private]

```

Start the [TTimer](#) instance. If the timer was already running it is reset first.

Parameters

in, out	<i>this</i>	The TTimer instance.
---------	-------------	--------------------------------------

Definition at line 255 of file [TimerClass.f03](#).

4.2.2.12 resume() `integer function timerclass::resume (`
`class(ttimer), intent(inout) this) [private]`

Restart the [TTimer](#) instance after a pause. If the timer is not paused, nothing will happen and TS=-1 is returned.

Parameters

<code>in, out</code>	<code>this</code>	The TTimer instance.
----------------------	-------------------	--------------------------------------

Returns

TS The index of the first timestamp.

Definition at line 142 of file [TimerClass.f03](#).

4.2.2.13 start() `integer function timerclass::start (`
`class(ttimer), intent(inout) this) [private]`

Start the [TTimer](#) instance (clean start, everything is reset). If the timer was already running it is reset first.

Parameters

<code>in, out</code>	<code>this</code>	The TTimer instance.
----------------------	-------------------	--------------------------------------

Returns

TS The index of the first timestamp.

Definition at line 124 of file [TimerClass.f03](#).

4.2.2.14 stoptimer() `integer function timerclass::stoptimer (`
`class(ttimer), intent(inout) this,`
`integer, intent(out), optional IO) [private]`

Stops the [TTimer](#) instance (terminal fashion...no restart possible).

If the timer was not running nothing will happen. The optional IO parameter will return an error code.

IO values:

- 0 : all is well
- -1 : The timer was not running, so nothing to stop.
- -2 : The timer was already stopped, so nothing to stop.

In case of error, TS will be set to -1.

Parameters

in, out	this	The TTimer instance.
out	IO	Optional parameter giving the error-status. [OPTIONAL ; DEFAULT= 0]

Returns

TS The index of the final timestamp.

Definition at line 229 of file [TimerClass.f03](#).

5 Data Type Documentation

5.1 timeclass::ttime Module Reference

The TTime class contains all time functionality with regard to a single time stamp.

Public Member Functions

- procedure, pass, public [settime](#)
Function to set the [TTime](#) instance to the current time, with millisecond resolution.
- procedure, pass, public [getjuliandaynumber](#)
Function returning the Julian Day Number as a 4-byte integer.
- procedure, pass, public [isleapyear](#)
Function returning true/false if the year of the [TTime](#) instance is a leap year.
- procedure, pass, public [gettimestring](#)
Returns a string with the time as a string.
- procedure, pass, public [gettimeseconds](#)
Function returning the time in (fractional) seconds (double precision).
- generic, public [assignment](#) => [copy](#)
Function to copy one [TTime](#) instance to the current one via the "=" assignment. This is the function performing the actual operator overloading.
- generic, public [operator](#) => [add](#)
Function to add two [TTime](#) instance via the "+" operator. This is the function performing the actual operator overloading.
- generic, public [operator](#) => [subtract](#)
Function to subtract two [TTime](#) instance via the "-" operator. This is the function performing the actual operator overloading.

Protected Member Functions

- final [destructor](#)
Destructor of the [TTime](#) object instance.

Private Member Functions

- procedure, pass(this) [calculatejdn](#)
Calculating the Julian Day number based on the Gregorian date set in the [TTime](#) object.
- procedure, pass(this) [setjdn](#)
Set the Julian Day Number of a [TTime](#) instance manually.
- procedure, pass(this) [setgregoriandatefromjdn](#)
Subroutine which transforms the set Julian Day Number into a Gregorian Calender date.
- procedure, pass(this) [copy](#)
Function to copy one [TTime](#) instance to the current one via the "=" assignment. This is the function performing the actual operator overloading.
- procedure, pass(this) [add](#)
Function to add two [TTime](#) instance via the "+" operator. This is the function performing the actual operator overloading.
- procedure, pass(this) [subtract](#)
Function to subtract two [TTime](#) instance via the "-" operator. This is the function performing the actual operator overloading.
- pure type([ttime](#)) function [constructor](#) ()
Constructor for the [TTime](#) class.

Private Attributes

- integer [year](#)
The year.
- integer [month](#)
The month (as integer).
- integer [day](#)
Day of the month.
- integer(kind=4) [jdn](#)
The Julian Day Number, as a long-int (4-byte)
- real [daysecs](#)
Number of seconds of the day, with millisecond resolution.

5.1.1 Detailed Description

The TTime class contains all time functionality with regard to a single time stamp.

Definition at line 50 of file [TimeClass.f03](#).

5.1.2 Member Function/Subroutine Documentation

5.1.2.1 [add\(\)](#) procedure, pass(this) timeclass::ttime::add [private]

Function to add two [TTime](#) instance via the "+" operator. This is the function performing the actual operator overloading.

Adding two full dates is maybe a bit strange to do. In our case, we don't just add the days and add the months, but we add the days of the year (via their Julian Day Numbers) and transform these back to months and days. (why keep life simple if we can make it complicated?)

usage:

```
total = this + that
```

This line also calls the [assignment operator](#).

Parameters

in	this	The TTime instance before the "+" operator.
in	that	The TTime instance after the "+" operator.

Returns

Total The [TTime](#) instance representing the sum.

Definition at line 65 of file [TimeClass.f03](#).

5.1.2.2 assignment() `generic, public timeclass::ttime::assignment`

Function to copy one [TTime](#) instance to the current one via the "=" assignment. This is the function performing the actual operator overloading.

usage:

```
tnew = told
```

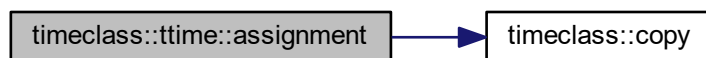
Parameters

in, out	this	The TTime instance before the "=" assignment.
in	from	The TTime instance after the "=" assignment.

Definition at line 70 of file [TimeClass.f03](#).

References [timeclass::copy\(\)](#).

Here is the call graph for this function:

**5.1.2.3 calculatejdn()** `procedure, pass(this) timeclass::ttime::calculatejdn [private]`

Calculating the Julian Day number based on the Gregorian date set in the [TTime](#) object.

In practice the Gregorian calendar date set in the [TTime](#) instance is transformed into a Julian Day Number, and stored in the instance as [time::jdn](#).

Parameters

in, out	this	The TTime instance being called.
---------	------	--------------------------------------------------

Definition at line 60 of file [TimeClass.f03](#).

5.1.2.4 constructor() `pure type(ttime) function timeclass::ttime::constructor [private]`

Constructor for the [TTime](#) class.

Note

This constructor does not set the time. It just enters zero's.

usage:

```
Type(TTime) :: T
t = ttime()
```

Returns

Time An instance of the TTime class.

Definition at line 98 of file [TimeClass.f03](#).

5.1.2.5 copy() `procedure, pass(this) timeclass::ttime::copy [private]`

Function to copy one [TTime](#) instance to the current one via the "=" assignment. This is the function performing the actual operator overloading.

usage:

```
tnew = told
```

Parameters

in, out	this	The TTime instance before the "=" assignment.
in	from	The TTime instance after the "=" assignment.

Definition at line 64 of file [TimeClass.f03](#).

5.1.2.6 destructor() `final timeclass::ttime::destructor [final], [protected]`

Destructor of the [TTime](#) object instance.

This subroutine is automatically called upon finalization of the instance.

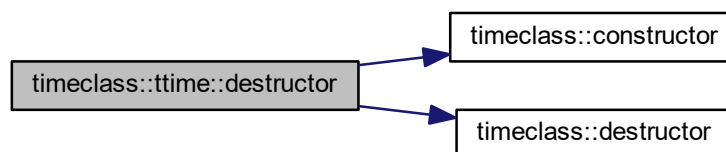
Parameters

<code>in, out</code>	<code>this</code>	The instance of the TTime class in need of destruction.
----------------------	-------------------	-------------------------------------------------------------------------

Definition at line 74 of file [TimeClass.f03](#).

References [timeclass::constructor\(\)](#), and [timeclass::destructor\(\)](#).

Here is the call graph for this function:



5.1.2.7 `getjuliandaynumber()` procedure, pass, public `timeclass::ttime::getjuliandaynumber`

Function returning the Julian Day Number as a 4-byte integer.

Use `integer(kind=4)`.

Parameters

<code>in, out</code>	<code>this</code>	The TTime instance being called.
----------------------	-------------------	--------------------------------------------------

Returns

JDN The integer Julian Day Number.

Definition at line 63 of file [TimeClass.f03](#).

5.1.2.8 `gettimeseconds()` procedure, pass, public `timeclass::ttime::gettimeseconds`

Function returning the time in (fractional) seconds (double precision).

Parameters

<code>in</code>	<code>this</code>	The TTime instance.
-----------------	-------------------	-------------------------------------

Returns

sec total number of seconds representing the "time" as a double precision value.

Definition at line 69 of file [TimeClass.f03](#).

5.1.2.9 gettimestring() `procedure, pass, public timeclass::ttime::gettimestring`

Returns a string with the time as a string.

Format options for fmt:

- full: dd/mm/yyyy hh:mm:ss.mmm
- date: dd/mm/yyyy
- time: hh:mm:ss.mmm
- days: Gives the total time in fractional days (best used for time-differences). Uses the Julian Day Number.
- hours: Same as days, but transformed to hours.
- seconds: Same as days, but transformed to seconds.

Parameters

in	<i>this</i>	The TTime instance transform into a string.
in	<i>fmt</i>	String representing the possible formatting. [OPTIONAL , DEFAULT = full]

Returns

TS String with formatted time.

Definition at line 68 of file [TimeClass.f03](#).

5.1.2.10 isleapyear() `procedure, pass, public timeclass::ttime::isleapyear`

Function returning true/false if the year of the [TTime](#) instance is a leap year.

A leap year is a multiple of 4, but not 100, unless 400

Parameters

in	<i>this</i>	The TTime instance to check the leap-year.
----	-------------	------------------------------------------------------------

Returns

Leap Boolean indicating if the year is a leap-year.

Definition at line 67 of file [TimeClass.f03](#).

5.1.2.11 operator() [1/2] `generic, public timeclass::ttime::operator`

Function to add two [TTime](#) instance via the "+" operator. This is the function performing the actual operator overloading.

Adding two full dates is maybe a bit strange to do. In our case, we don't just add the days and add the months, but we add the days of the year (via their Julian Day Numbers) and transform these back to months and days. (why keep life simple if we can make it complicated?)

usage:

```
total = this + that
```

This line also calls the [assignment operator](#).

Parameters

in	<i>this</i>	The TTime instance before the "+" operator.
in	<i>that</i>	The TTime instance after the "+" operator.

Returns

Total The [TTime](#) instance representing the sum.

Definition at line 71 of file [TimeClass.f03](#).

References [timeclass::add\(\)](#).

Here is the call graph for this function:



5.1.2.12 operator() [2/2] `generic, public timeclass::ttime::operator`

Function to subtract two [TTime](#) instance via the "-" operator. This is the function performing the actual operator overloading.

usage:

```
total = this - that
```

This line also calls the [assignment operator](#).

Note

The result should remain a positive number.

Parameters

in	<i>this</i>	The TTime instance before the "-" operator.
in	<i>that</i>	The TTime instance after the "-" operator.

Returns

Total The [TTime](#) instance representing the difference.

Definition at line 72 of file [TimeClass.f03](#).

References [timeclass::subtract\(\)](#).

Here is the call graph for this function:



5.1.2.13 setgregoriandatefromjdn() procedure, pass(this) timeclass::time::setgregoriandatefromjdn [private]

Subroutine which transforms the set Julian Day Number into a Gregorian Calendar date.

Note

The routine is only valid for a JDN \geq 0.

Parameters

in, out	<i>this</i>	The TTime instance being called.
out	<i>IO</i>	Returns 0 on success, and -1 for failure. [OPTIONAL]

Definition at line 62 of file [TimeClass.f03](#).

5.1.2.14 setjdn() procedure, pass(this) timeclass::time::setjdn [private]

Set the Julian Day Number of a [TTime](#) instance manually.

Note

The Julian Day Number should be ≥ 0 . For negative values it is set to 0, and an error value is set to IO

Parameters

in, out	<i>this</i>	The TTime instance being called.
in	<i>JDN</i>	A positive integer(kind=4) value representing a valid Julian Day Number.
out	<i>IO</i>	Integer value returning 0 upon success, and a negative value(=JDN) in case of failure.

Definition at line 61 of file [TimeClass.f03](#).

5.1.2.15 **settime()** `procedure, pass, public timeclass::ttime::settime`

Function to set the [TTime](#) instance to the current time, with millisecond resolution.

Note

As this subroutine uses the `date_and_time` intrinsic it is an *impure* subroutine.

Parameters

in, out	<i>this</i>	The TTime instance being called.
---------	-------------	--------------------------------------------------

Definition at line 59 of file [TimeClass.f03](#).

5.1.2.16 **subtract()** `procedure, pass(this) timeclass::ttime::subtract [private]`

Function to subtract two [TTime](#) instance via the "-" operator. This is the function performing the actual operator overloading.

usage:

```
total = this - that
```

This line also calls the [assignment operator](#).

Note

The result should remain a positive number.

Parameters

in	<i>this</i>	The TTime instance before the "-" operator.
in	<i>that</i>	The TTime instance after the "-" operator.

Returns

Total The [TTime](#) instance representing the difference.

Definition at line 66 of file [TimeClass.f03](#).

5.1.3 Member Data Documentation

5.1.3.1 day `integer timeclass::ttime::day [private]`

Day of the month.

Definition at line 54 of file [TimeClass.f03](#).

5.1.3.2 daysecs `real timeclass::ttime::daysecs [private]`

Number of seconds of the day, with millisecond resolution.

Definition at line 56 of file [TimeClass.f03](#).

5.1.3.3 jdn `integer(kind=4) timeclass::ttime::jdn [private]`

The Julian Day Number, as a long-int (4-byte)

Definition at line 55 of file [TimeClass.f03](#).

5.1.3.4 month `integer timeclass::ttime::month [private]`

The month (as integer).

Definition at line 53 of file [TimeClass.f03](#).

5.1.3.5 year `integer timeclass::ttime::year [private]`

The year.

Definition at line 52 of file [TimeClass.f03](#).

The documentation for this module was generated from the following file:

- [TimeClass.f03](#)

5.2 timerclass::timer Module Reference

A class for keeping track of time in calculations, by creating a list of timestamps.

Public Member Functions

- procedure, pass, public [start](#)
Start the [TTimer](#) instance (clean start, everything is reset). If the timer was already running it is reset first.
- procedure, pass, public [interrupt](#)
Puts the [TTimer](#) instance on hold.
- procedure, pass, public [resume](#)
Restart the [TTimer](#) instance after a pause. If the timer is not paused, nothing will happen and TS=-1 is returned.
- procedure, pass, public [addtimeflag](#)
Add an additional timestamp without changing the status (running/paused) of the [TTimer](#) instance.
- procedure, pass, public [stoptimer](#)
Stops the [TTimer](#) instance (terminal fashion...no restart possible).
- procedure, pass, public [reset](#)
Start the [TTimer](#) instance. If the timer was already running it is reset first.
- procedure, pass, public [printelapsedtimereport](#)
Print small timings report to unit UN.
- procedure, pass, public [getelapsedtimestring](#)
Returns a string representing the elapsed time. +.
- generic, public [assignment](#) => [copy](#)
Function to copy one [TTimer](#) instance to the current one via the "=" assignment.
- generic, public [getelapsedtime](#) => [getelapsedtime_total](#), [getelapsedtime_steps](#)
Generic interface to the [GetElapsedTime_total](#) and [GetElapsedTime_steps](#) methods of the [TTimer](#) class.

Protected Member Functions

- final [destructor](#)
Destructor of the [TTimer](#) class. Cleans up the instance upon finalization.

Private Member Functions

- procedure, pass(this) [getelapsedtime_total](#)
Returns the number of seconds which elapsed between the start and stop timestamps.
- procedure, pass(this) [getelapsedtime_steps](#)
Returns the number of seconds which elapsed between two timestamps. This is always a positive value.
- procedure, pass(this) [copy](#)
Function to copy one [TTimer](#) instance to the current one via the "=" assignment.
- procedure, pass(this) [addtimestamp](#)
Add a timestamp to a running [TTimer](#) instance, returning the index of the timestamp.
- pure type([ttimer](#)) function [constructor](#) ()
Constructor for the [TTimer](#) instances.

Private Attributes

- integer [ntimes](#)
The number of timestamps stored.
- integer [maxtimes](#)
The size of the timestamp list.
- type(ttime), dimension(:), allocatable [timestamps](#)
Allocatable list of timestamps.
- logical, dimension(:), allocatable [timedinterval](#)
logical indicating if the timer was running during an interval between two timestamps or not
- logical [running](#)
True if the timer is running.
- logical [paused](#)
True if the timer is paused.
- logical [stopped](#)
True if the timer is stopped (final end)

5.2.1 Detailed Description

A class for keeping track of time in calculations, by creating a list of timestamps.

Definition at line 50 of file [TimerClass.f03](#).

5.2.2 Member Function/Subroutine Documentation

5.2.2.1 `addtimeflag()` procedure, pass, public timerclass::ttimer::addtimeflag

Add an additional timestamp without changing the status (running/paused) of the [TTimer](#) instance.

If the timer is stopped nothing will happen, and -1 is returned.

Parameters

<code>in, out</code>	<code>this</code>	The TTimer instance.
----------------------	-------------------	----------------------

Returns

TS The index of the timestamp.

Definition at line 66 of file [TimerClass.f03](#).

5.2.2.2 `addtimestamp()` procedure, pass(this) timerclass::ttimer::addtimestamp [private]

Add a timestamp to a running [TTimer](#) instance, returning the index of the timestamp.

In case the timer is not running, nothing is done and -1 is returned.

Parameters

<i>in, out</i>	<i>this</i>	The TTimer instance.
----------------	-------------	--------------------------------------

Returns

TS The index of the timestamp.

Definition at line 74 of file [TimerClass.f03](#).

5.2.2.3 assignment() `generic, public timerclass::ttimer::assignment`

Function to copy one [TTimer](#) instance to the current one via the "=" assignment.

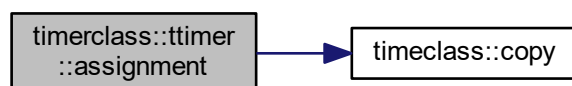
Parameters

<i>in, out</i>	<i>this</i>	The TTimer instance before the "=" assignment.
<i>in</i>	<i>from</i>	The TTimer instance after the "=" assignment.

Definition at line 75 of file [TimerClass.f03](#).

References [timeclass::copy\(\)](#).

Here is the call graph for this function:

**5.2.2.4 constructor()** `pure type(ttimer) function timerclass::ttimer::constructor [private]`

Constructor for the [TTimer](#) instances.

Usage:

```
Type(TTimer) :: T
t=ttimer()
```

Returns

Returns a [TTimer](#) object

Definition at line 104 of file [TimerClass.f03](#).

5.2.2.5 `copy()` `procedure, pass(this) timerclass::ttimer::copy [private]`

Function to copy one [TTimer](#) instance to the current one via the "=" assignment.

Parameters

in, out	<i>this</i>	The TTimer instance before the "=" assignment.
in	<i>from</i>	The TTimer instance after the "=" assignment.

Definition at line 73 of file [TimerClass.f03](#).

5.2.2.6 `destructor()` `final timerclass::ttimer::destructor [final], [protected]`

Destructor of the [TTimer](#) class. Cleans up the instance upon finalization.

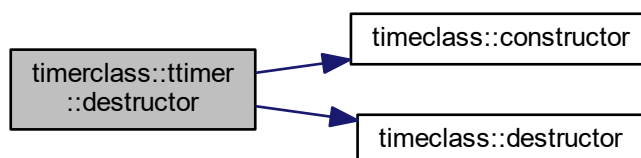
Parameters

in, out	<i>this</i>	The TTimer instance being destroyed automatically.
---------	-------------	--------------------------------------------------------------------

Definition at line 81 of file [TimerClass.f03](#).

References [timeclass::constructor\(\)](#), and [timeclass::destructor\(\)](#).

Here is the call graph for this function:



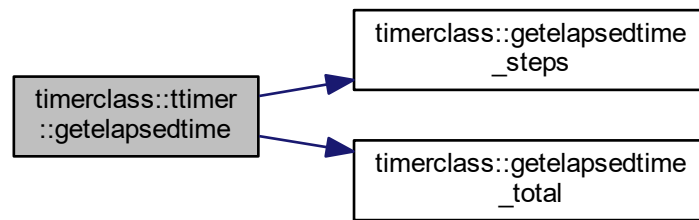
5.2.2.7 `getelapsedtime()` `generic, public timerclass::ttimer::getelapsedtime`

Generic interface to the [GetElapsedTime_total](#) and [GetElapsedTime_steps](#) methods of the [TTimer](#) class.

Definition at line 76 of file [TimerClass.f03](#).

References [timerclass::getelapsedtime_steps\(\)](#), and [timerclass::getelapsedtime_total\(\)](#).

Here is the call graph for this function:



5.2.2.8 getelapsedtime_steps() `procedure, pass(this) timerclass::ttimer::getelapsedtime_steps [private]`

Returns the number of seconds which elapsed between two timestamps. This is always a positive value.

NOTE:

- If the timestamps are out of range, then -1 is returned.
- If the start comes after end, then they are exchanged such that a positive time is obtained.

Parameters

in	<i>this</i>	The TTimer instance.
in	<i>Tstart, Tend</i>	The indices of the start and end time.
in	<i>INCL_PAUSE</i>	Logical indicating if the time during which the timer was paused should be included as well. [OPTIONAL , DEFAULT = .false.]

Returns

Seconds The (fractional) number of seconds elapsed, as double precision real.

Definition at line 72 of file [TimerClass.f03](#).

5.2.2.9 getelapsedtime_total() `procedure, pass(this) timerclass::ttimer::getelapsedtime_total [private]`

Returns the number of seconds which elapsed between the start and stop timestamps.

Parameters

in	<i>this</i>	The TTimer instance.
in	<i>INCL_PAUSE</i>	Logical indicating if the time during which the timer was paused should be included as well. [OPTIONAL , DEFAULT = .false.]

Returns

Seconds The (fractional) number of seconds elapsed, as double precision real.

Definition at line 71 of file [TimerClass.f03](#).

5.2.2.10 getelapsedtimestring() procedure, pass, public timerclass::ttimer::getelapsedtimestring

Returns a string representing the elapsed time. +.

Parameters

in	<i>this</i>	The TTimer instance.
in	<i>TSTART,TSTOP</i>	Indices of the start and end-times. [OPTIONAL , DEFAULT : TSTART=1, TSTOP=index of last timestamp]
in	<i>INCL_PAUSE</i>	Logical indicating if the time during which the timer was paused should be included as well. [OPTIONAL , DEFAULT = .false.]
in	<i>FMT</i>	string indicating the format for the time-string. <ul style="list-style-type: none"> • sec : xxx.xxx secs • hms : xxxx h xx min xx.xxx secs • dhms: xx days xx h xx min xx.xxx secs • hour: xxx.xxx hours • day : xxx.xxx days [OPTIONAL, DEFAULT= sec]

Returns

str The string with the formatted time.

Definition at line 70 of file [TimerClass.f03](#).

5.2.2.11 interrupt() procedure, pass, public timerclass::ttimer::interrupt

Puts the [TTimer](#) instance on hold.

If the timer was not running nothing will happen. The optional IO parameter will return an error code.

IO values:

- 0 : all is well
- -1 : The timer was not running, so nothing to pause.
- -2 : The timer was already stopped/paused, so nothing to pause.

In case of error, TS will be set to -1.

Parameters

in, out	<i>this</i>	The TTimer instance.
out	<i>IO</i>	Optional parameter giving the error-status. [OPTIONAL ; DEFAULT= 0]

Returns

TS The index of the final timestamp.

Definition at line 64 of file [TimerClass.f03](#).

5.2.2.12 printelapseditimereport() procedure, pass, public timerclass::ttimer::printelapseditimereport

Print small timings report to unit UN.

Parameters

in	<i>this</i>	The TTimer instance.
in	<i>message</i>	String containing a message to include.
in	<i>Tstart, Tstop</i>	Integer indexes of the start and stop times
in	<i>UN</i>	Integer indicating the unit to write the report to.
in	<i>INCL_PAUSE</i>	Logical indicating if the time during which the timer was paused should be included as well. [OPTIONAL, DEFAULT = .false.]

Definition at line 69 of file [TimerClass.f03](#).

5.2.2.13 reset() procedure, pass, public timerclass::ttimer::reset

Start the [TTimer](#) instance. If the timer was already running it is reset first.

Parameters

in, out	<i>this</i>	The TTimer instance.
---------	-------------	--------------------------------------

Definition at line 68 of file [TimerClass.f03](#).

5.2.2.14 resume() `procedure, pass, public timerclass::ttimer::resume`

Restart the [TTimer](#) instance after a pause. If the timer is not paused, nothing will happen and TS=-1 is returned.

Parameters

in, out	<i>this</i>	The TTimer instance.
---------	-------------	--------------------------------------

Returns

TS The index of the first timestamp.

Definition at line 65 of file [TimerClass.f03](#).

5.2.2.15 start() `procedure, pass, public timerclass::ttimer::start`

Start the [TTimer](#) instance (clean start, everything is reset). If the timer was already running it is reset first.

Parameters

in, out	<i>this</i>	The TTimer instance.
---------	-------------	--------------------------------------

Returns

TS The index of the first timestamp.

Definition at line 63 of file [TimerClass.f03](#).

5.2.2.16 stoptimer() `procedure, pass, public timerclass::ttimer::stoptimer`

Stops the [TTimer](#) instance (terminal fashion...no restart possible).

If the timer was not running nothing will happen. The optional IO parameter will return an error code.

IO values:

- 0 : all is well
- -1 : The timer was not running, so nothing to stop.
- -2 : The timer was already stopped, so nothing to stop.

In case of error, TS will be set to -1.

Parameters

in, out	<i>this</i>	The TTimer instance.
out	<i>IO</i>	Optional parameter giving the error-status. [OPTIONAL ; DEFAULT= 0]

Returns

TS The index of the final timestamp.

Definition at line 67 of file [TimerClass.f03](#).

5.2.3 Member Data Documentation

5.2.3.1 maxtimes `integer timerclass::ttimer::maxtimes [private]`

The size of the timestamp list.

Definition at line 54 of file [TimerClass.f03](#).

5.2.3.2 ntimes `integer timerclass::ttimer::ntimes [private]`

The number of timestamps stored.

Definition at line 53 of file [TimerClass.f03](#).

5.2.3.3 paused `logical timerclass::ttimer::paused [private]`

True if the timer is paused.

Definition at line 59 of file [TimerClass.f03](#).

5.2.3.4 running `logical timerclass::ttimer::running [private]`

True if the timer is running.

Definition at line 58 of file [TimerClass.f03](#).

5.2.3.5 stopped `logical timerclass::ttimer::stopped [private]`

True if the timer is stopped (final end)

Definition at line 60 of file [TimerClass.f03](#).

5.2.3.6 timedinterval `logical, dimension(:), allocatable timerclass::ttimer::timedinterval [private]`

logical indicating if the timer was running during an interval between two timestamps or not

Definition at line 56 of file [TimerClass.f03](#).

5.2.3.7 timestamps `type(ttime), dimension(:), allocatable timerclass::ttimer::timestamps [private]`

Allocatable list of timestamps.

Definition at line 55 of file [TimerClass.f03](#).

The documentation for this module was generated from the following file:

- [TimerClass.f03](#)

6 File Documentation

6.1 TimeClass.f03 File Reference

Data Types

- module [timeclass::ttime](#)
The TTime class contains all time functionality with regard to a single time stamp.
- module [timeclass::ttime](#)
The TTime class contains all time functionality with regard to a single time stamp.

Modules

- module [timeclass](#)
*The **TimeClass module** contains the [TTime class](#) used by the [TTimer class](#) for practical timing.*

Functions/Subroutines

- pure type(ttime) function `timeclass::constructor` ()
Constructor for the `TTime` class.
- subroutine `timeclass::settime` (this)
Function to set the `TTime` instance to the current time, with millisecond resolution.
- pure subroutine `timeclass::calculatejdn` (this)
Calculating the Julian Day number based on the Gregorian date set in the `TTime` object.
- pure subroutine `timeclass::setjdn` (this, JDN, IO)
Set the Julian Day Number of a `TTime` instance manually.
- pure integer(kind=4) function `timeclass::getjuliandaynumber` (this)
Function returning the Julian Day Number as a 4-byte integer.
- pure subroutine `timeclass::setgregoriandatefromjdn` (this, IO)
Subroutine which transforms the set Julian Day Number into a Gregorian Calendar date.
- pure subroutine `timeclass::copy` (this, from)
Function to copy one `TTime` instance to the current one via the "=" assignment. This is the function performing the actual operator overloading.
- pure type(ttime) function `timeclass::add` (this, that)
Function to add two `TTime` instance via the "+" operator. This is the function performing the actual operator overloading.
- pure type(ttime) function `timeclass::subtract` (this, that)
Function to subtract two `TTime` instance via the "-" operator. This is the function performing the actual operator overloading.
- pure logical function `timeclass::isleapyear` (this)
Function returning true/false if the year of the `TTime` instance is a leap year.
- pure character(len=255) function `timeclass::gettimestring` (this, fmt)
Returns a string with the time as a string.
- pure real(dp) function `timeclass::gettimeseconds` (this)
Function returning the time in (fractional) seconds (double precision).
- subroutine `timeclass::destructor` (this)
Destructor of the `TTime` object instance.

6.2 TimeClass.f03

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 !MIT License
00003 !
00004 !Copyright (c) Dr. Dr. Danny E.P. Vanpoucke, https://dannyvanpoucke.be
00005 !
00006 !Permission is hereby granted, free of charge, to any person obtaining a copy
00007 !of this software and associated documentation files (the "Software"), to deal
00008 !in the Software without restriction, including without limitation the rights
00009 !to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 !copies of the Software, and to permit persons to whom the Software is
00011 !furnished to do so, subject to the following conditions:
00012 !
00013 !The above copyright notice and this permission notice shall be included in all
00014 !copies or substantial portions of the Software.
00015 !
00016 !THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 !IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 !FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 !AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 !LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 !OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 !SOFTWARE.
00023 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00024
00025 !+++++
00040 module timeclass
00041     implicit none
00042     private
00043
00044

```



```

00045      !+++++
00050      type, public :: ttime
00051      private
00052          integer :: year
00053          integer :: month
00054          integer :: day
00055          integer(kind=4) :: jdn
00056          real :: daysecs
00057      contains
00058      private
00059          procedure, pass(this), public :: settime
00060          procedure, pass(this) :: calculatejdn
00061          procedure, pass(this) :: setjdn
00062          procedure, pass(this) :: setgregoriandatefromjdn
00063          procedure, pass(this), public :: getjuliandaynumber
00064          procedure, pass(this) :: copy
00065          procedure, pass(this) :: add
00066          procedure, pass(this) :: subtract
00067          procedure, pass(this), public :: isleapyear
00068          procedure, pass(this), public :: gettimestring
00069          procedure, pass(this), public :: gettimeseconds
00070          generic, public :: assignment(=) => copy
00071          generic, public :: operator(+) => add
00072          generic, public :: operator(-) => subtract
00073          final :: destructor
00074      end type ttime
00075
00076      ! This is the only way a constructor can be created, as no "initial" exists, emulates the C++
00077      constructor behaviour
00078      interface ttime
00079          module procedure constructor
00080      end interface ttime
00081
00082      contains
00083      !+++++
00084      pure function constructor() Result (Time)
00085      type(ttime) :: time
00086
00087          time%year=0
00088          time%month=0
00089          time%day=0
00090          time%JDN=0
00091          time%daysecs=0
00092      end function constructor
00093      !+++++
00094      subroutine settime(this)
00095          class(ttime), intent(inout) :: this
00096          integer time_array(8)
00097
00098          call date_and_time(values=time_array) ! this function seems to be impure
00099          this%year=time_array(1)
00100          this%month=time_array(2)
00101          this%day=time_array(3)
00102          this%daysecs=((real(time_array(5))*60.0 + real(time_array(6)))*60.0+real(time_array(7)))&
00103              & + 0.001*real(time_array(8))
00104          ! Transform the Gregorian date into a Julian Day Number
00105          call this%CalculateJDN()
00106      end subroutine settime
00107      !+++++
00108      pure subroutine calculatejdn(this)
00109          class(ttime), intent(inout) :: this
00110
00111          this%JDN=int((1461*(this%year+4800+int((this%month-14)/12))/4)+&
00112              &int((367*(this%month-2-12*int((this%month-14)/12))/12)-&
00113              &int((3*int((this%year+4900+int((this%month-14)/12))/100))/4)&
00114              &+this%day-32075
00115      end subroutine calculatejdn
00116      !+++++
00117      pure subroutine setjdn(this,JDN,IO)
00118          class(ttime), intent(inout) :: this
00119          integer(kind=4), intent(in) :: jdn
00120          integer, intent(out), optional :: io
00121
00122          if (jdn>=0) then
00123              this%JDN=jdn
00124          else
00125              this%JDN=0
00126          end if
00127          call this%SetGregorianDateFromJDN()
00128
00129          if (present(io)) then
00130              io=0
00131              if (this%JDN<0) io=-1
00132          end if
00133      end subroutine setjdn

```

```

00174     end subroutine setjdn
00175     !+++++
00183     pure function getjuliandaynumber(this) Result(JDN)
00184         class(ttime), intent(in) :: this
00185         integer(kind=4) :: jdn
00186
00187         jdn=this%JDN
00188
00189     end function getjuliandaynumber
00190     !+++++
00198     pure subroutine setgregoriandatefromjdn(this, IO)
00199         class(ttime), intent(inout) :: this
00200         integer, intent(out), optional :: io
00201
00202         integer(kind=4), parameter :: y=4716, j=1401, m=2, n=12, r=4, p=1461, v=3, u=5, s=153, w=2,
b=274277, c=-38
00203         integer(kind=4) :: f, e, g, h
00204
00205         if (present(io)) then
00206             io=0
00207             if (this%JDN<0) io=-1
00208             if (io<0) return
00209         end if
00210         f=this%JDN + j +int((int((4*this%JDN + b)/146097)*3)/4) + c
00211         e=r*f + v
00212         g=int(mod(e,p)/r)
00213         h=u*g+w
00214         this%day=int((mod(h,s))/u)+1
00215         this%month=mod(int(h/s)+m,n)+1
00216         this%year=int(e/p)-y+int((n+m-this%month)/n)
00217
00218     end subroutine setgregoriandatefromjdn
00219     !+++++
00232     pure subroutine copy(this,from)
00233         class(ttime), intent(inout) :: this
00234         class(ttime), intent(in) :: from
00235
00236         this%year=from%year
00237         this%month=from%month
00238         this%day=from%day
00239         this%daysecs=from%daysecs
00240         this%JDN=from%JDN
00241
00242     end subroutine copy
00243     !+++++
00263     pure function add(this,that) Result(Total)
00264         class(ttime), intent(in) :: this, that
00265         Type(ttime) :: total
00266
00267         integer :: overflow, ios
00268         integer(kind=4) :: days
00269
00270         total = ttime()
00271         total%daysecs=this%daysecs+that%daysecs
00272         overflow=0
00273         if (total%daysecs>60.0) then
00274             overflow=int((total%daysecs - modulo(total%daysecs,60.0))/60.0)
00275             total%daysecs=modulo(total%daysecs,60.0)
00276         end if
00277         ! now using Julian Day Numbers:
00278         days=this%GetJulianDayNumber()+that%GetJulianDayNumber()+overflow
00279         call total%SetJDN(days,io=ios) ! this also sets the days, months and years
00280
00281     end function add
00282     !+++++
00299     pure function subtract(this,that) Result(Total)
00300         class(ttime), intent(in) :: this, that
00301         Type(ttime) :: total
00302
00303         integer(kind=4) :: overflow, days
00304         integer :: ios
00305         !Using julian day numbers and day seconds, this gets a bit more simple
00306
00307         total = ttime()
00308         total%daysecs=this%daysecs-that%daysecs
00309         overflow=0
00310         do while (total%daysecs<0.0)
00311             overflow=overflow+1
00312             total%daysecs=total%daysecs+86400.0
00313         end do
00314         days=this%GetJulianDayNumber()-that%GetJulianDayNumber()-overflow
00315         call total%SetJDN(days,ios)
00316
00317     end function subtract
00318     !+++++
00326     pure function isleapyear(this) Result(Leap)
00327         class(ttime), intent(in) :: this

```

```

00328         logical :: leap
00329
00330         leap=.false.
00331         if (mod(this%year,4)==0) then
00332             leap=.true.
00333             if (mod(this%year,100)==0) then
00334                 leap=.false.
00335                 if (mod(this%year,400)==0) leap=.true.
00336             end if
00337         end if
00338
00339     end function isleapyear
00340     !+++++
00355     pure function gettimestring(this,fmt) Result(TS)
00356     use, intrinsic :: iso_fortran_env
00357         class(ttime), intent(in) :: this
00358         character(len=*), intent(in), optional :: fmt
00359         character(len=255) :: ts
00360
00361         integer, parameter :: dp = real64
00362         integer :: h, m
00363         real :: s
00364         real(dp) :: fullt
00365         character(len=50) :: fmtstr
00366
00367         s=mod(this%daysecs,60.0)
00368         m=mod(int((this%daysecs-s)/60),60)
00369         h=int(this%daysecs/3600)
00370         fmtstr="full"
00371         if (present(fmt)) fmtstr=fmt
00372
00373         select case(trim(adjustl(fmtstr)))
00374             case('full')
00375                 write(ts,'(2(I2,A),I4,2(A,I2),A,F6.3)') this%day,"/",this%month,"/",this%year,"
00376                 ",h,":",m,":",s
00377             case('date')
00378                 write(ts,'(2(I2,A),I4)') this%day,"/",this%month,"/",this%year
00379             case('time')
00380                 write(ts,'(2(I2,A),F6.3)') h,":",m,":",s
00381             case('days')
00382                 fullt=this%GetJulianDayNumber()*1.0_dp + (this%daysecs/86400.0_dp)
00383                 write(ts,'(F20.8,A)') fullt," days"
00384             case('hours')
00385                 fullt=(this%GetJulianDayNumber()*1.0_dp + (this%daysecs/86400.0_dp))*24.0_dp
00386                 write(ts,'(F20.8,A)') fullt," hours"
00387             case('seconds')
00388                 fullt=(this%GetJulianDayNumber()*86400.0_dp + this%daysecs)
00389                 write(ts,'(F20.8,A)') fullt," secs"
00390             case default
00391                 write(ts,'(2(I2,A),I4,2(A,I2),A,F6.3)') this%day,"/",this%month,"/",this%year,"
00392                 ",h,":",m,":",s
00393             end select
00394
00395     end function gettimestring
00396     !+++++
00401     pure function gettimeseconds(this) Result(sec)
00402     use, intrinsic :: iso_fortran_env
00403         class(ttime), intent(in) :: this
00404         integer, parameter :: dp = real64
00405         real(dp) :: sec
00406
00407         sec=this%GetJulianDayNumber()*86400.0_dp + this%daysecs
00408
00409     end function gettimeseconds
00410     !+++++
00418     subroutine destructor(this)
00419         type(ttime) :: this
00420
00421     end subroutine destructor
00422
00423
00424 end module

```

6.3 TimerClass.f03 File Reference

Data Types

- module `timerclass::ttimer`

A class for keeping track of time in calculations, by creating a list of timestamps.

- module `timerclass::ttimer`

A class for keeping track of time in calculations, by creating a list of timestamps.

Modules

- module `timerclass`
Module containing the `TTimer` class for practical timing.

Functions/Subroutines

- pure type(timer) function `timerclass::constructor` ()
Constructor for the `TTimer` instances.
- integer function `timerclass::start` (this)
Start the `TTimer` instance (clean start, everything is reset). If the timer was already running it is reset first.
- integer function `timerclass::resume` (this)
Restart the `TTimer` instance after a pause. If the timer is not paused, nothing will happen and `TS=-1` is returned.
- integer function `timerclass::addtimeflag` (this)
Add an additional timestamp without changing the status (running/paused) of the `TTimer` instance.
- integer function `timerclass::interrupt` (this, IO)
Puts the `TTimer` instance on hold.
- integer function `timerclass::stoptimer` (this, IO)
Stops the `TTimer` instance (terminal fashion...no restart possible).
- pure subroutine `timerclass::reset` (this)
Start the `TTimer` instance. If the timer was already running it is reset first.
- integer function `timerclass::addtimestamp` (this)
Add a timestamp to a running `TTimer` instance, returning the index of the timestamp.
- pure real(dp) function `timerclass::getelapsedtime_total` (this, INCL_PAUSE)
Returns the number of seconds which elapsed between the start and stop timestamps.
- pure real(dp) function `timerclass::getelapsedtime_steps` (this, Tstart, Tend, INCL_PAUSE)
Returns the number of seconds which elapsed between two timestamps. This is always a positive value.
- pure character(len=50) function `timerclass::getelapsedtimestring` (this, TSTART, TSTOP, INCL_PAUSE, F↔MT)
Returns a string representing the elapsed time. +.
- subroutine `timerclass::printelapsedtimereport` (this, message, Tstart, Tstop, UN, INCL_PAUSE)
Print small timings report to unit UN.
- pure subroutine `timerclass::copy` (this, from)
Function to copy one `TTimer` instance to the current one via the "=" assignment.
- subroutine `timerclass::destructor` (this)
Destructor of the `TTimer` class. Cleans up the instance upon finalization.

6.4 TimerClass.f03

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 !MIT License
00003 !
00004 !Copyright (c) Dr. Dr. Danny E.P. Vanpoucke, https://dannyvanpoucke.be
00005 !
00006 !Permission is hereby granted, free of charge, to any person obtaining a copy
00007 !of this software and associated documentation files (the "Software"), to deal
00008 !in the Software without restriction, including without limitation the rights
00009 !to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 !copies of the Software, and to permit persons to whom the Software is
00011 !furnished to do so, subject to the following conditions:
00012 !
00013 !The above copyright notice and this permission notice shall be included in all
00014 !copies or substantial portions of the Software.
00015 !
00016 !THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 !IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 !FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 !AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

```

```

00020 !LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 !OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 !SOFTWARE.
00023 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00024
00025
00026
00027
00028 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00040 module timerclass
00041     use timeclass
00042     implicit none
00043     private
00044
00045     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00050     type, public :: ttimer
00051     private
00052     !! Some EXTRA INFO on \ref ttimer
00053     integer :: ntimes
00054     integer :: maxtimes
00055     Type(ttime), allocatable :: timestamps(:)
00056     logical, allocatable :: timedinterval(:)
00058     logical :: running
00059     logical :: paused
00060     logical :: stopped
00061     contains
00062     private
00063     procedure, pass(this), public :: start
00064     procedure, pass(this), public :: interrupt
00065     procedure, pass(this), public :: resume
00066     procedure, pass(this), public :: addtimeflag
00067     procedure, pass(this), public :: stoptimer
00068     procedure, pass(this), public :: reset
00069     procedure, pass(this), public :: printelapsedtimereport
00070     procedure, pass(this), public :: getelapsedtimestring
00071     procedure, pass(this) :: getelapsedtime_total
00072     procedure, pass(this) :: getelapsedtime_steps
00073     procedure, pass(this) :: copy
00074     procedure, pass(this), :: addtimestamp
00075     generic, public :: assignment(=) => copy
00076     generic, public :: getelapsedtime => getelapsedtime_total, getelapsedtime_steps
00081     final :: destructor
00083     end type ttimer
00084
00085
00086     ! This is the only way a constructor can be created, as no "initial" exists, emulates the C++
    constructor behaviour
00087     interface ttimer
00088         module procedure constructor
00089     end interface ttimer
00090
00091     contains
00092     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00103     pure function constructor() Result(Timer)
00104     Type(ttimer) :: timer
00105
00106     timer%ntimes=0
00107     timer%maxtimes=10
00108     allocate(timer%timestamps(1:10))
00109     allocate(timer%timedInterval(1:10))
00110     timer%timedInterval(:)=.false.
00111     timer%running=.false.
00112     timer%paused=.false.
00113     timer%stopped=.false.
00114
00115     end function constructor
00116     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00123     function start(this) Result(TS)
00124     class(ttimer), intent(inout) :: this
00125     integer :: ts
00126
00127     if (this%running) call this%reset()
00128     this%running=.true.
00129     ts=this%AddTimestamp()
00130     this%ntimes=ts
00131     this%timedInterval(ts)=.true. ! this timer-interval is accounted
00132
00133     end function start
00134     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00141     function resume(this) Result(TS)
00142     class(ttimer), intent(inout) :: this
00143     integer :: ts
00144
00145     ts=-1
00146     if (this%paused) then
00147         this%running=.true.
00148         this%paused=.false.

```

```

00149         ts=this%AddTimestamp()
00150         this%ntimes=ts
00151         this%timedInterval(ts)=.true. ! this timer-interval is accounted
00152     end if
00153
00154 end function resume
00155 !+++++
00164 function addtimeflag(this) Result(TS)
00165     class(ttimer), intent(inout) :: this
00166     integer :: ts
00167
00168     ts=-1
00169     if (this%running.or.this%paused) then
00170         ts=this%AddTimestamp()
00171         this%ntimes=ts
00172         this%timedInterval(ts)=this%running ! is this timer-interval accounted?
00173     end if
00174
00175 end function addtimeflag
00176 !+++++
00192 function interrupt(this,IO) Result(TS)
00193     class(ttimer), intent(inout) :: this
00194     integer, intent(out), optional :: io
00195     integer :: ts
00196
00197     ts=-1
00198     if (this%running) then
00199         if ((.not.(this%stopped)).and.(.not.(this%paused))) then
00200             if (present(io)) io=0
00201             ts=this%AddTimestamp()
00202             this%paused=.true.
00203             this%timedInterval(ts)=.false. ! as the timer is paused, the following interval should
not be accounted
00204         else
00205             if (present(io)) io=-2
00206         end if
00207     else
00208         if (present(io)) io=-1
00209     end if
00210
00211 end function interrupt
00212 !+++++
00228 function stoptimer(this,IO) Result(TS)
00229     class(ttimer), intent(inout) :: this
00230     integer, intent(out), optional :: io
00231     integer :: ts
00232
00233     ts=-1
00234     if (this%running) then
00235         if (.not.(this%stopped)) then
00236             if (present(io)) io=0
00237             ts=this%AddTimestamp()
00238             this%stopped=.true.
00239             this%timedInterval(ts)=.false. ! as the timer is stopped, the following interval
should not be accounted
00240         else
00241             if (present(io)) io=-2
00242         end if
00243     else
00244         if (present(io)) io=-1
00245     end if
00246
00247 end function stoptimer
00248 !+++++
00254 pure subroutine reset(this)
00255     class(ttimer), intent(inout) :: this
00256
00257     this%ntimes=0
00258     this%maxtimes=10
00259     if (allocated(this%timestamps)) deallocate(this%timestamps)
00260     allocate(this%timestamps(1:10))
00261     if (allocated(this%timedInterval)) deallocate(this%timedInterval)
00262     allocate(this%timedInterval(1:10))
00263     this%timedInterval(:)=.false.
00264     this%running=.false.
00265     this%paused=.false.
00266     this%stopped=.false.
00267
00268 end subroutine reset
00269 !+++++
00278 function addtimestamp(this)Result(TS)
00279     class(ttimer), intent(inout) :: this
00280     integer:: ts
00281
00282     type(ttime), allocatable :: tmp(:)
00283     logical, allocatable :: tmp1(:)
00284

```

```

00285     if (this%running) then
00286         ts=this%ntimes+1
00287         if (ts>this%maxtimes) then ! we need to extend the arrays
00288             allocate(tmp(1:this%maxtimes))
00289             allocate(tmp1(1:this%maxtimes))
00290             tmp(1:this%maxtimes)=this%timestamps(1:this%maxtimes)
00291             tmp1(1:this%maxtimes)=this%timedInterval(1:this%maxtimes)
00292             this%maxtimes=this%maxtimes+10
00293             if (allocated(this%timestamps)) deallocate(this%timestamps)
00294             allocate(this%timestamps(1:this%maxtimes))
00295             if (allocated(this%timedInterval)) deallocate(this%timedInterval)
00296             allocate(this%timedInterval(1:this%maxtimes))
00297             this%timestamps(1:this%maxtimes-10)=tmp(1:this%maxtimes-10)
00298             this%timedInterval(1:this%maxtimes-10)=tmp1(1:this%maxtimes-10)
00299             this%timedInterval(this%maxtimes-10:this%maxtimes)=.false.
00300             deallocate(tmp)
00301             deallocate(tmp1)
00302         end if
00303         this%timestamps(ts)=ttime()
00304         call this%timestamps(ts)%SetTime()
00305         this%ntimes=ts
00306     else
00307         ts=-1
00308     end if
00309
00310 end function addtimestamp
00311 !+++++
00319 pure function getelapsedtime_total(this,INCL_PAUSE) Result(sec)
00320 use, intrinsic :: iso_fortran_env
00321 class(timer), intent(in) :: this
00322 logical, intent(in), optional :: incl_pause
00323 integer, parameter :: dp = real64
00324 real(dp) :: sec
00325
00326 logical :: pause
00327
00328 pause=.false.
00329 if (present(incl_pause)) pause=incl_pause
00330
00331 sec=this%GetElapsedTime_steps(1,this%maxtimes,incl_pause=pause)
00332
00333 end function getelapsedtime_total
00334 !+++++
00347 pure function getelapsedtime_steps(this,Tstart,Tend,INCL_PAUSE) Result(sec)
00348 use, intrinsic :: iso_fortran_env
00349 class(timer), intent(in) :: this
00350 integer, intent(in) :: tstart, tend
00351 logical, intent(in), optional :: incl_pause
00352 integer, parameter :: dp = real64
00353 real(dp) :: sec
00354
00355 type(ttime) :: elap, tmp
00356 integer :: t1, t2, nr
00357
00358 if ((tstart<1).or.(tend<1).or.(tstart>this%ntimes).or.(tend>this%ntimes)) then
00359     sec=-1.0
00360     return
00361 end if
00362
00363 if (tstart>tend) then
00364     t1=tend
00365     t2=tstart
00366 else
00367     t1=tstart
00368     t2=tend
00369 end if
00370
00371 elap=this%timestamps(t2)-this%timestamps(t1)
00372 sec=elap%GetTimeSeconds()
00373 if (present(incl_pause)) then
00374     if (incl_pause) then ! pauses should be excluded, so we subtract the again
00375         do nr=1,this%ntimes-1
00376             if (.not.this%timedInterval(nr)) then
00377                 tmp=this%timestamps(nr+1)-this%timestamps(nr)
00378                 sec=sec-tmp%GetTimeSeconds()
00379             end if
00380         end do
00381     end if
00382 end if
00383
00384 end function getelapsedtime_steps
00385 !+++++
00401 pure function getelapsedtimestring(this,TSTART, TSTOP, INCL_PAUSE, FMT) Result(str)
00402 use, intrinsic :: iso_fortran_env
00403 class(timer), intent(in) :: this
00404 integer, intent(in), optional :: tstart, tstop

```

```

00405     logical, intent(in), optional :: incl_pause
00406     character(len=*), intent(in), optional :: fmt
00407     character(len=50) :: str
00408
00409     integer, parameter :: dp = real64
00410     real(dp) :: sec
00411     integer :: t1, t2, nr, hour, day, minute
00412     character(len=4) :: opt
00413     character(len=255) :: line
00414
00415     t1=1
00416     t2=this%ntimes
00417     if(present(tstart)) then
00418         if ((tstart>0).and.(tstart<=this%ntimes)) t1=tstart
00419     end if
00420     if(present(tstop)) then
00421         if ((tstop>0).and.(tstop<=this%ntimes)) t2=tstop
00422     end if
00423     if (t1>t2) then
00424         nr=t1
00425         t1=t2
00426         t2=nr
00427     end if
00428
00429     sec=this%GetElapsedTime(t1,t2,incl_pause)
00430     !now transform to a string
00431     opt='sec'
00432     if (present(fmt)) opt=trim(adjustl(fmt))
00433
00434     select case(trim(adjustl(opt)))
00435     case ('sec')
00436         write(line,'(F30.3,A)') sec," secs "
00437     case ('hour')
00438         write(line,'(F30.3,A)') sec/3600.0_dp," hours "
00439     case ('day')
00440         write(line,'(F30.3,A)') sec/86400.0_dp," days "
00441     case ('hms')
00442         hour=floor(sec/3600.0_dp)
00443         sec=sec-(hour*3600.0_dp)
00444         minute=floor(sec/60.0_dp)
00445         sec=sec-(minute*60.0_dp)
00446         write(line,'(I0,A,I2,A,F6.3)') hour," h ",minute," min ",sec," secs "
00447     case ('dhms')
00448         day=floor(sec/86400.0_dp)
00449         sec=sec-(day*86400.0_dp)
00450         hour=floor(sec/3600.0_dp)
00451         sec=sec-(hour*3600.0_dp)
00452         minute=floor(sec/60.0_dp)
00453         sec=sec-(minute*60.0_dp)
00454         write(line,'(I0,A,2(I2,A),F6.3)') day," days ",hour," h ",minute," min ",sec," secs "
00455     end select
00456     write(str,'(3A)') " ",trim(adjustl(line))," "
00457
00458 end function getelapsedTimeString
00459 !+++++
00460 subroutine printelapsedtimereport(this, message, Tstart, Tstop, UN, INCL_PAUSE)
00470 use, intrinsic :: iso_fortran_env
00471 class(ttimer), intent(inout) :: this
00472 character(len=*), intent(in) :: message
00473 integer, intent(in) :: Tstart, Tstop
00474 integer, intent(in) :: UN
00475 logical, intent(in), optional :: INCL_PAUSE
00476
00477 character(len=50) :: line
00478 integer, parameter :: dp = real64
00479 real(dp) :: sec
00480 integer :: ih, im
00481
00482 write(un,"(2A)",advance='NO') trim(message)," : "
00483 line=this%GetElapsedTimeString(tstart,tstop,incl_pause,'sec')
00484 write(un,"(A)") trim(adjustl(line))
00485 sec=this%GetElapsedTime(tstart,tstop,incl_pause)
00486
00487 ih=floor(sec/3600.0_dp)
00488 sec=sec-dble(ih)*3600.0_dp
00489 im=floor(sec/60.0_dp)
00490 sec=sec-dble(im)*60.0_dp
00491
00492 write(un,'(I8,A)') ih," hours"
00493 write(un,'(I8,A)') im," minutes"
00494 write(un,'(F8.3,A)') sec," seconds"
00495
00496 end subroutine printelapsedtimereport
00497 !+++++
00503 pure subroutine copy(this,from)
00504 class(ttimer), intent(inout) :: this
00505 class(ttimer), intent(in) :: from

```



```
00506
00507     integer :: nr
00508
00509     this%maxtimes=from%maxtimes
00510     this%ntimes=from%ntimes
00511     this%paused=from%paused
00512     this%running=from%running
00513     this%stopped=from%stopped
00514     allocate(this%timedInterval(1:this%maxtimes))
00515     this%timedInterval(1:this%maxtimes)=from%timedInterval(1:this%maxtimes)
00516     allocate(this%timestamps(1:this%maxtimes))
00517     do nr=1, this%ntimes
00518         this%timestamps(nr)=from%timestamps(nr)
00519     end do
00520
00521     end subroutine copy
00522     !+++++
00528     subroutine destructor(this)
00529         Type(ttimer) :: this
00530
00531         if (allocated(this%timestamps)) deallocate(this%timestamps)
00532         if (allocated(this%timedInterval)) deallocate(this%timedInterval)
00533
00534     end subroutine destructor
00535
00536
00537 end module timerclass
```

Index

- add
 - timeclass, 4
 - timeclass::time, 19
- addtimeflag
 - timerclass, 12
 - timerclass::ttimer, 30
- addtimestamp
 - timerclass, 12
 - timerclass::ttimer, 30
- assignment
 - timeclass::time, 20
 - timerclass::ttimer, 31
- calculatejdn
 - timeclass, 4
 - timeclass::time, 20
- constructor
 - timeclass, 5
 - timeclass::time, 21
 - timerclass, 12
 - timerclass::ttimer, 31
- copy
 - timeclass, 5
 - timeclass::time, 21
 - timerclass, 12
 - timerclass::ttimer, 31
- day
 - timeclass::time, 28
- daysecs
 - timeclass::time, 28
- destructor
 - timeclass, 6
 - timeclass::time, 21
 - timerclass, 13
 - timerclass::ttimer, 32
- getelapsedtime
 - timerclass::ttimer, 32
- getelapsedtime_steps
 - timerclass, 13
 - timerclass::ttimer, 33
- getelapsedtime_total
 - timerclass, 14
 - timerclass::ttimer, 33
- getelapsedtimestring
 - timerclass, 15
 - timerclass::ttimer, 34
- getjuliandaynumber
 - timeclass, 7
 - timeclass::time, 22
- gettimesecs
 - timeclass, 7
 - timeclass::time, 22
- gettimestring
 - timeclass, 7
- timeclass::time, 23
- interrupt
 - timerclass, 15
 - timerclass::ttimer, 34
- isleapyear
 - timeclass, 8
 - timeclass::time, 23
- jdn
 - timeclass::time, 28
- maxtimes
 - timerclass::ttimer, 37
- month
 - timeclass::time, 28
- ntimes
 - timerclass::ttimer, 37
- operator
 - timeclass::time, 24
- paused
 - timerclass::ttimer, 37
- printelapsedtimereport
 - timerclass, 16
 - timerclass::ttimer, 35
- reset
 - timerclass, 16
 - timerclass::ttimer, 35
- resume
 - timerclass, 16
 - timerclass::ttimer, 35
- running
 - timerclass::ttimer, 37
- setgregoriandatefromjdn
 - timeclass, 8
 - timeclass::time, 25
- setjdn
 - timeclass, 9
 - timeclass::time, 25
- settime
 - timeclass, 9
 - timeclass::time, 27
- start
 - timerclass, 17
 - timerclass::ttimer, 36
- stopped
 - timerclass::ttimer, 37
- stoptimer
 - timerclass, 17
 - timerclass::ttimer, 36
- subtract
 - timeclass, 10

- timeclass::ttime, 27
- timeclass, 2
 - add, 4
 - calculatejdn, 4
 - constructor, 5
 - copy, 5
 - destructor, 6
 - getjuliandaynumber, 7
 - gettimeseconds, 7
 - gettimestring, 7
 - isleapyear, 8
 - setgregoriandatefromjdn, 8
 - setjdn, 9
 - settime, 9
 - subtract, 10
- TimeClass.f03, 38, 39
- timeclass::ttime, 18
 - add, 19
 - assignment, 20
 - calculatejdn, 20
 - constructor, 21
 - copy, 21
 - day, 28
 - daysecs, 28
 - destructor, 21
 - getjuliandaynumber, 22
 - gettimeseconds, 22
 - gettimestring, 23
 - isleapyear, 23
 - jdn, 28
 - month, 28
 - operator, 24
 - setgregoriandatefromjdn, 25
 - setjdn, 25
 - settime, 27
 - subtract, 27
 - year, 28
- timedinterval
 - timerclass::ttimer, 37
- timerclass, 10
 - addtimeflag, 12
 - addtimestamp, 12
 - constructor, 12
 - copy, 12
 - destructor, 13
 - getelapsedtime_steps, 13
 - getelapsedtime_total, 14
 - getelapsedtimestring, 15
 - interrupt, 15
 - printelapsedtimereport, 16
 - reset, 16
 - resume, 16
 - start, 17
 - stoptimer, 17
- TimerClass.f03, 42, 43
- timerclass::ttimer, 28
 - addtimeflag, 30
 - addtimestamp, 30
 - assignment, 31
 - constructor, 31
 - copy, 31
 - destructor, 32
 - getelapsedtime, 32
 - getelapsedtime_steps, 33
 - getelapsedtime_total, 33
 - getelapsedtimestring, 34
 - interrupt, 34
 - maxtimes, 37
 - ntimes, 37
 - paused, 37
 - printelapsedtimereport, 35
 - reset, 35
 - resume, 35
 - running, 37
 - start, 36
 - stopped, 37
 - stoptimer, 36
 - timedinterval, 37
 - timestamps, 38
- timestamps
 - timerclass::ttimer, 38
- year
 - timeclass::ttime, 28