TimerClass

1.0

Generated by Doxygen 1.8.16

Sat Oct 26 2019 17:25:15

1 Modules Index

1.1 Modules List

Here is a list of all modules with brief descriptions:

2 Data Type Index

2.1 Data Types List

Here are the data types with brief descriptions:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Module Documentation

4.1 timeclass Module Reference

Time class used by the timerclass for practical timing.

Data Types

- interface ttime

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)

  Transforms the set Gregorian calender date into a Julian Day Number.
- pure subroutine setjdn (this, JDN, IO)

  Set the Julian Day Number.
- 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 Date Number into a Gregorian Calender date.
- pure subroutine copy (this, from)

  Function to copy one TTime instance to the current one via the "=" assignment.

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

  Function to add two TTime instance via the "+" operator.
- pure type(ttime) function subtract (this, that)

  Function to subtract two TTime instance via the "-" operator.
- 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. This subroutine is automatically called upon finalization of the instance.

4.1.1 Detailed Description

Time class used by the timerclass for practical timing.

Internally we use Julian Day Numbers to compare dates. As a result we do not accept "negative" dates. 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 should be 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.

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 and transform these back to months and days. (why keep life simple if we can make it complicated?

Parameters

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

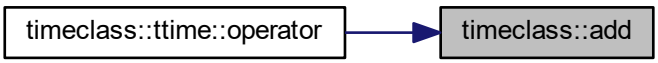
Returns

Total The TTime instance representing the sum.

Definition at line 231 of file TimeClass.f03.

Referenced by timeclass::ttime::operator().

Here is the caller graph for this function:



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

Transforms the set Gregorian calender date into a Julian Day Number.

Parameters

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

Definition at line 120 of file TimeClass.f03.

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

Constructor for the TTime class.

Note that this constructor does not set the time. It just enters zero's

usage: Type(TTime) :: T T = TTime()

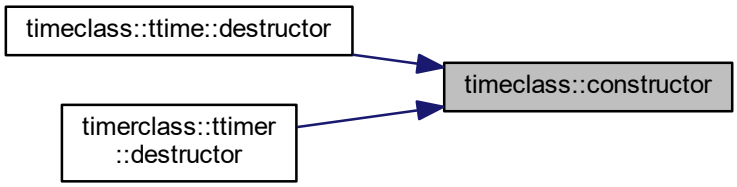
Returns

Time An instance of the TTime class.

Definition at line 83 of file TimeClass.f03.

Referenced by timeclass::ttime::destructor(), and timerclass::ttimer::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.

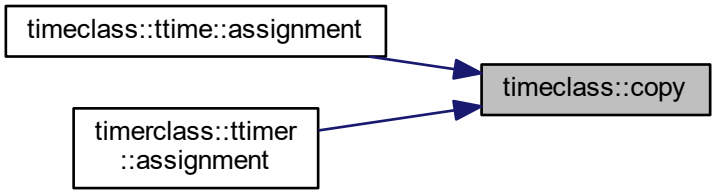
Parameters

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

Definition at line 208 of file TimeClass.f03.

Referenced by timeclass::ttime::assignment(), and timerclass::ttimer::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.

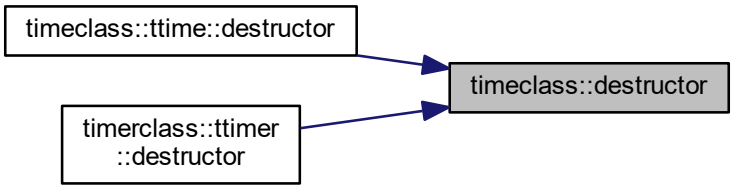
Parameters

in,out	*this*	The instance of the TTime class in need of destruction.

Definition at line 376 of file TimeClass.f03.

Referenced by timeclass::ttime::destructor(), and timerclass::ttimer::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

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

Returns

JDN The integer Julian Day Number.

Definition at line 165 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

in	*this*	The TTime instance.

Returns

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

Definition at line 360 of file TimeClass.f03.

4.1.2.8 gettimestring() `pure character(len=255) function timeclass::gettimestring ( 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	*this*	The TTime instance transform into a string.
in	*fmt*	String representing the possible formatting. [**OPTIONAL**, **DEFAULT** = full]

Returns

TS String with formatted time.

Definition at line 314 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	*this*	The TTime instance to check the leap-year.

Returns

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

Definition at line 285 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 Date Number into a Gregorian Calender date.

NOTE: The routine is only valid for a JDN>=0.

Parameters

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

Definition at line 181 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.

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

Definition at line 139 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.

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

Parameters

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

Definition at line 100 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.

NOTE: The result should remain a positive number.

Parameters

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

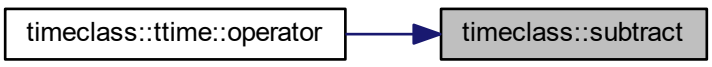
Returns

Total The TTime instance representing the difference.

Definition at line 259 of file TimeClass.f03.

Referenced by timeclass::ttime::operator().

Here is the caller graph for this function:



4.2 timerclass Module Reference

Timer class for practical timing.

Data Types

- interface ttimer

Functions/Subroutines

- pure type(ttimer) function constructor ()

  Constructor for the TTimer instances.
- integer function start (this)

  Start the timer (clean start, everything is reset). If the timer was already running it is reset first.
- integer function resume (this)

  Restart the timer after a pause. If the timer is not paused, nothing will happen and TS=-1 is returned.
- integer function addtimeflag (this)

  Allows for the introduction of an additional time-stamp without changing the timer-status (running/paused). If the Timer is stopped nothing will happen, and -1 is returned.
- integer function interrupt (this, IO)

  Puts the timer on hold. If the timer was not running nothing will happen. The optional IO parameter will return an error code.
  IO values:
- integer function stoptimer (this, IO)

  Stops the timer. If the timer was not running nothing will happen. The optional IO parameter will return an error code.
  IO values:
- pure subroutine reset (this)

  Start the timer. If the timer was already running it is reset first.
- integer function addtimestamp (this)

  Add a timestamp to a running timer, 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

Timer class for practical timing.

Version

2.0 of the timing module.

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 )`

Allows for the introduction of an additional time-stamp without changing the timer-status (running/paused). If the Timer is stopped nothing will happen, and -1 is returned.

Parameters

in,out	*this*	The TTimer instance.

Returns

TS The index of the timestamp.

Definition at line 146 of file TimerClass.f03.

4.2.2.2 addtimestamp() `integer function timerclass::addtimestamp (`

```
class(ttimer), intent(inout) this )
```

Add a timestamp to a running timer, returning the index of the timestamp.

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

Parameters

in,out	*this*	The TTimer instance.

Returns

TS The index of the timestamp.

Definition at line 257 of file TimerClass.f03.

4.2.2.3 constructor() `pure type(ttimer) function timerclass::constructor ( )`

Constructor for the TTimer instances.

Usage: Type(TTimer) :: T T=TTimer()

Returns

Returns a TTimer object

Definition at line 86 of file TimerClass.f03.

4.2.2.4 copy() `pure subroutine timerclass::copy (`

```
class(ttimer), intent(inout) this,
class(ttimer), intent(in) from )
```

Function to copy one TTimer instance to the current one via the "=" assignment.

Parameters

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

Definition at line 481 of file TimerClass.f03.

4.2.2.5 destructor()

```
subroutine timerclass::destructor (
    type(ttimer) this )
```

Destructor of the TTimer class. Cleans up the instance upon finalization.

Definition at line 504 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 )
```

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

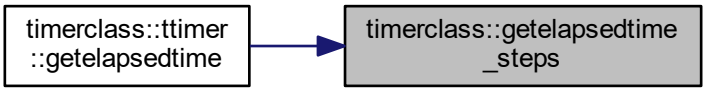
Returns

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

Definition at line 325 of file TimerClass.f03.

Referenced by timerclass::ttimer::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 )
```

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

Parameters

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

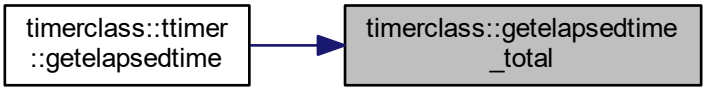
Returns

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

Definition at line 297 of file TimerClass.f03.

Referenced by timerclass::ttimer::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 )
```

Returns a string representing the elapsed time. +.

Parameters

in	*this*	The TTimer instance.
in	*TSTART,TSTOP*	Indices of the start and end-times. [**OPTIONAL**, **DEFAULT:** TSTART=1, TSTOP=index of last timestamp]
in	*INCL_PAUSE*	Logical indicating if the time during which the timer was paused should be included as well. [**OPTIONAL**, **DEFAULT** = .false.]
in	*FMT*	string indicating the format for the time-string. • 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 379 of file TimerClass.f03.

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

```
class(ttimer), intent(inout) this,
integer, intent(out), optional IO )
```

Puts the timer 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	*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 173 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 )
```

Print small timings report to unit UN.

Parameters

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

Definition at line 447 of file TimerClass.f03.

4.2.2.11 reset()

```
pure subroutine timerclass::reset (
    class(ttimer), intent(inout) this )
```

Start the timer. If the timer was already running it is reset first.

Parameters

in,out	*this*	The TTimer instance.

Definition at line 234 of file TimerClass.f03.

4.2.2.12 resume() integer function timerclass::resume (class(ttimer), intent(inout) *this*)

Restart the timer after a pause. If the timer is not paused, nothing will happen and TS=-1 is returned.

Parameters

in,out	*this*	The TTimer instance.

Returns

TS The index of the first timestamp.

Definition at line 124 of file TimerClass.f03.

4.2.2.13 start() integer function timerclass::start (class(ttimer), intent(inout) *this*)

Start the timer (clean start, everything is reset). If the timer was already running it is reset first.

Parameters

in,out	*this*	The TTimer instance.

Returns

TS The index of the first timestamp.

Definition at line 106 of file TimerClass.f03.

4.2.2.14 stoptimer() integer function timerclass::stoptimer (class(ttimer), intent(inout) *this*, integer, intent(out), optional *IO*)

Stops the timer. 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 208 of file TimerClass.f03.

5 Data Type Documentation

5.1 timeclass::ttime Interface Reference

Public Member Functions

- procedure, pass, public settime

  Set the time to the current time.
- procedure, pass(this) calculatejdn

  Private function calculating the Julian Day number based on the Gregorian date set in the TTime object.
- procedure, pass(this) setjdn

  private function to set the Julian Day Number
- procedure, pass(this) setgregoriandatefromjdn

  Private function transforming a Julian Day number into a Gregorian calender date.
- procedure, pass, public getjuliandaynumber

  returns the Julian day
- procedure, pass(this) copy

  Copy content from other TTime instance, private, accessed via the assignment statement.
- procedure, pass(this) add

  Add two TTime instances.
- procedure, pass(this) subtract

  Add two TTime instances.
- procedure, pass, public isleapyear

  Returns true if the year component is a leap year.
- procedure, pass, public gettimestring

  returns a formatted time-string
- procedure, pass, public gettimeseconds

  returns the time as a fractional number of seconds, double precision
- generic, public assignment => copy

  This is how copy is used.
- generic, public operator => add

  This is how add is used.
- generic, public operator => subtract

  This is how subtract is used.
- final destructor

Public 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.

Private Member Functions

- pure type(ttime) function constructor ()
  Constructor for the TTime class.

5.1.1 Detailed Description

Definition at line 39 of file TimeClass.f03.

5.1.2 Member Function/Subroutine Documentation

5.1.2.1 add() `procedure, pass(this) timeclass::ttime::add ( )`

Add two TTime instances.

Definition at line 54 of file TimeClass.f03.

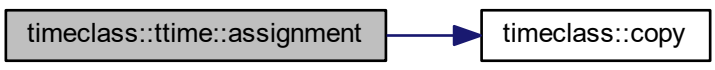
5.1.2.2 assignment() `generic, public timeclass::ttime::assignment ( )`

This is how copy is used.

Definition at line 59 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 function calculating the Julian Day number based on the Gregorian date set in the TTime object.

Definition at line 49 of file TimeClass.f03.

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

Constructor for the TTime class.

Note that 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 83 of file TimeClass.f03.

5.1.2.5 copy() procedure, pass(this) timeclass::ttime::copy ( )

Copy content from other TTime instance, private, accessed via the assignment statement.

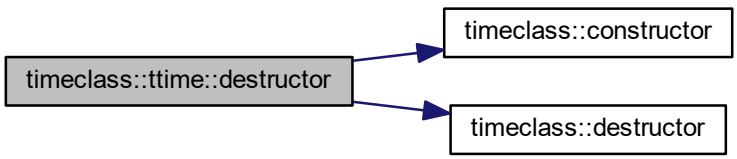
Definition at line 53 of file TimeClass.f03.

5.1.2.6 destructor() final timeclass::ttime::destructor ( ) [final]

Definition at line 63 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 ( )

returns the Julian day

Definition at line 52 of file TimeClass.f03.

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

returns the time as a fractional number of seconds, double precision

Definition at line 58 of file TimeClass.f03.

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

returns a formatted time-string

Definition at line 57 of file TimeClass.f03.

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

Returns true if the year component is a leap year.

Definition at line 56 of file TimeClass.f03.

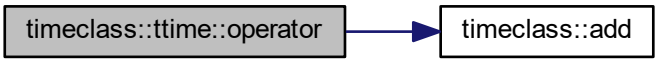
5.1.2.11 operator() [1/2] generic, public timeclass::ttime::operator ( )

This is how add is used.

Definition at line 60 of file TimeClass.f03.

References timeclass::add().

Here is the call graph for this function:



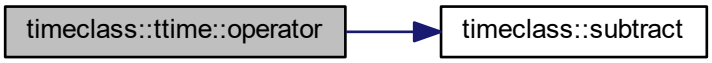
5.1.2.12 operator() [2/2] generic, public timeclass::ttime::operator ( )

This is how subtract is used.

Definition at line 61 of file TimeClass.f03.

References timeclass::subtract().

Here is the call graph for this function:



5.1.2.13 setgregoriandatefromjdn() procedure, pass(this) timeclass::ttime::setgregoriandatefromjdn ( )

Private function transforming a Julian Day number into a Gregorian calender date.

Definition at line 51 of file TimeClass.f03.

5.1.2.14 setjdn() procedure, pass(this) timeclass::ttime::setjdn ( )

private function to set the Julian Day Number

Definition at line 50 of file TimeClass.f03.

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

Set the time to the current time.

Definition at line 48 of file TimeClass.f03.

5.1.2.16 subtract() procedure, pass(this) timeclass::ttime::subtract ( )

Add two TTime instances.

Definition at line 55 of file TimeClass.f03.

5.1.3 Member Data Documentation

5.1.3.1 day `integer timeclass::ttime::day`

Day of the month.

Definition at line 43 of file TimeClass.f03.

5.1.3.2 daysecs `real timeclass::ttime::daysecs`

Number of seconds of the day, with millisecond resolution.

Definition at line 45 of file TimeClass.f03.

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

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

Definition at line 44 of file TimeClass.f03.

5.1.3.4 month `integer timeclass::ttime::month`

The month (as integer).

Definition at line 42 of file TimeClass.f03.

5.1.3.5 year `integer timeclass::ttime::year`

The year.

Definition at line 41 of file TimeClass.f03.

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

- TimeClass.f03

5.2 timerclass::ttimer Interface Reference

Public Member Functions

- procedure, pass, public start

  Clean start of the timer (includes a reset)
- procedure, pass, public interrupt

  Temporarily interrupt the timer.
- procedure, pass, public resume

  Resume timer after a pause.
- procedure, pass, public addtimeflag

  Add a timestamp and don't change the setting (running/paused) of the timer.
- procedure, pass, public stoptimer

  Stop the timer (terminal fashion...no restart possible)
- procedure, pass, public reset

  Reset the timer.
- procedure, pass, public printelapsedtimereport

  Print several lines with timing information.
- procedure, pass, public getelapsedtimestring

  return a string with the elapsed time
- procedure, pass(this) getelapsedtime_total

  Return the elapsed time in seconds between begin and end.
- procedure, pass(this) getelapsedtime_steps

  Return the elapsed time in seconds between two timestamps.
- procedure, pass(this) copy

  Make a copy of a timer object.
- generic, public assignment => copy

  This is how copy is used.
- generic, public getelapsedtime => getelapsedtime_total, getelapsedtime_steps
- final destructor
- pure type(ttimer) function constructor ()

  Constructor for the TTimer instances.

Public Attributes

- integer ntimes

  The number of timestamps stored.
- integer maxtimes

  The size of the timestamp list.
- type(ttime), dimension(:), allocatable 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)

Private Member Functions

- procedure, pass, private addtimestamp

  adds a timestamp, for internal purposes only

5.2.1 Detailed Description

Definition at line 41 of file TimerClass.f03.

5.2.2 Member Function/Subroutine Documentation

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

Add a timestamp and don't change the setting (running/paused) of the timer.

Definition at line 56 of file TimerClass.f03.

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

adds a timestamp, for internal purposes only

Definition at line 64 of file TimerClass.f03.

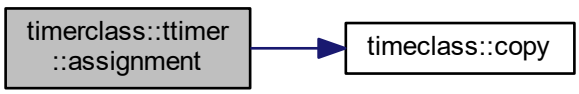
5.2.2.3 assignment() `generic, public timerclass::ttimer::assignment ( )`

This is how copy is used.

Definition at line 65 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 ( )`

Constructor for the TTimer instances.

Usage: Type(TTimer) :: T T=TTimer()

Returns

Returns a TTimer object

Definition at line 86 of file TimerClass.f03.

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

Make a copy of a timer object.

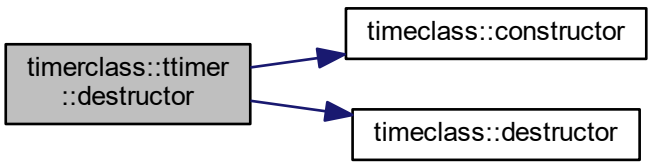
Definition at line 63 of file TimerClass.f03.

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

Definition at line 67 of file TimerClass.f03.

References timeclass::constructor(), and timeclass::destructor().

Here is the call graph for this function:

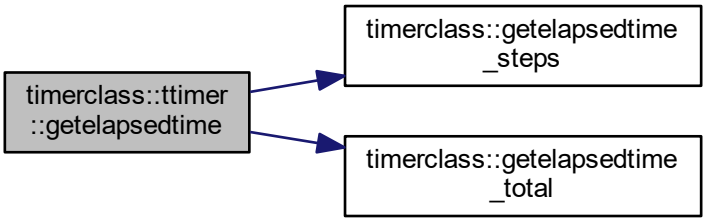


5.2.2.7 getelapsedtime() `generic, public timerclass::ttimer::getelapsedtime ( )`

Definition at line 66 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 ( )`

Return the elapsed time in seconds between two timestamps.

Definition at line 62 of file TimerClass.f03.

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

Return the elapsed time in seconds between begin and end.

Definition at line 61 of file TimerClass.f03.

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

return a string with the elapsed time

Definition at line 60 of file TimerClass.f03.

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

Temporarily interrupt the timer.

Definition at line 54 of file TimerClass.f03.

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

Print several lines with timing information.

Definition at line 59 of file TimerClass.f03.

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

Reset the timer.

Definition at line 58 of file TimerClass.f03.

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

Resume timer after a pause.

Definition at line 55 of file TimerClass.f03.

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

Clean start of the timer (includes a reset)

Definition at line 53 of file TimerClass.f03.

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

Stop the timer (terminal fashion...no restart possible)

Definition at line 57 of file TimerClass.f03.

5.2.3 Member Data Documentation

5.2.3.1 maxtimes `integer timerclass::ttimer::maxtimes`

The size of the timestamp list.

Definition at line 44 of file TimerClass.f03.

5.2.3.2 ntimes `integer timerclass::ttimer::ntimes`

The number of timestamps stored.

Definition at line 43 of file TimerClass.f03.

5.2.3.3 paused `logical timerclass::ttimer::paused`

True if the timer is paused.

Definition at line 49 of file TimerClass.f03.

5.2.3.4 running `logical timerclass::ttimer::running`

True if the timer is running.

Definition at line 48 of file TimerClass.f03.

5.2.3.5 stopped `logical timerclass::ttimer::stopped`

True if the timer is stopped (final end)

Definition at line 50 of file TimerClass.f03.

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

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

Definition at line 46 of file TimerClass.f03.

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

Definition at line 45 of file TimerClass.f03.

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

- TimerClass.f03

6 File Documentation

6.1 TimeClass.f03 File Reference

Data Types

- interface timeclass::ttime
- interface timeclass::ttime

Modules

- module timeclass

  Time class used by the timerclass 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)

  Transforms the set Gregorian calender date into a Julian Day Number.
- pure subroutine timeclass::setjdn (this, JDN, IO)

  Set the Julian Day Number.
- 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 Date Number into a Gregorian Calender date.
- pure subroutine timeclass::copy (this, from)

  Function to copy one TTime instance to the current one via the "=" assignment.
- pure type(ttime) function timeclass::add (this, that)

  Function to add two TTime instance via the "+" operator.
- pure type(ttime) function timeclass::subtract (this, that)

  Function to subtract two TTime instance via the "-" operator.
- 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. This subroutine is automatically called upon finalization of the instance.

6.2 TimeClass.f03

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!MIT License
!
!Copyright (c) 2019 Danny Vanpoucke, https://dannyvanpoucke.be
!
!Permission is hereby granted, free of charge, to any person obtaining a copy
!of this software and associated documentation files (the "Software"), to deal
!in the Software without restriction, including without limitation the rights
!to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
!copies of the Software, and to permit persons to whom the Software is
!furnished to do so, subject to the following conditions:
!
!The above copyright notice and this permission notice shall be included in all
!copies or substantial portions of the Software.
!
!THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
!IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
!FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
!AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
!LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
!OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
!SOFTWARE.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
module timeclass
    implicit none
    private

    type, public :: ttime
      private
        integer :: year
        integer :: month
        integer :: day
        integer(kind=4) :: jdn
        real     :: daysecs
    contains
      private
        procedure, pass(this),public :: settime
        procedure, pass(this)        :: calculatejdn
        procedure, pass(this)        :: setjdn
        procedure, pass(this)        :: setgregoriandatefromjdn
        procedure, pass(this),public :: getjuliandaynumber
        procedure, pass(this)        :: copy
        procedure, pass(this)        :: add
        procedure, pass(this)        :: subtract
        procedure, pass(this),public :: isleapyear
        procedure, pass(this),public :: gettimestring
        procedure, pass(this),public :: gettimeseconds
        generic, public :: assignment(=) => copy
        generic, public :: operator(+)    => add
        generic, public :: operator(-)    => subtract

        final :: destructor
    end type ttime

    interface ttime
        module procedure constructor
    end interface ttime

contains
    !++++++++++++++++++++++++++++++++++++++++++
    pure function constructor()Result(Time)
        type(ttime) :: time

        time%year=0
        time%month=0
        time%day=0
        time%JDN=0
        time%daysecs=0
    end function constructor
    !++++++++++++++++++++++++++++++++++++++++++++++++
    subroutine settime(this)
        class(ttime), intent(inout) :: this
        integer time_array(8)

        call date_and_time(values=time_array) ! this function seems to be impure
        this%year=time_array(1)
        this%month=time_array(2)
        this%day=time_array(3)
        this%daysecs=(((real(time_array(5))*60.0 + real(time_array(6)))*60.0)+real(time_array(7)))&
                    & + 0.001*real(time_array(8))
        ! Transform the Gregorian date into a Julian Day Number
        call this%CalculateJDN()
```

```
00111
00112     end subroutine settime
00113     !+++++++++++++++++++++++++++++++++++++++++++++++
00119     pure subroutine calculatejdn(this)
00120         class(ttime), intent(inout) :: this
00121
00122         this%JDN=int((1461*(this%year+4800+int((this%month-14)/12)))/4)+&
00123                 &int((367*(this%month-2-12*int((this%month-14)/12)))/12)-&
00124                 &int((3*int((this%year+4900+int((this%month-14)/12))/100))/4)&
00125                 &+this%day-32075
00126
00127     end subroutine calculatejdn
00128     !+++++++++++++++++++++++++++++++++++++++++++++++
00138     pure subroutine setjdn(this,JDN,IO)
00139         class(ttime), intent(inout) :: this
00140         integer(kind=4), intent(in) :: jdn
00141         integer, intent(out), optional :: io
00142
00143         if (jdn>=0) then
00144             this%JDN=jdn
00145         else
00146             this%JDN=0
00147         end if
00148         call this%SetGregorianDateFromJDN()
00149
00150         if (present(io)) then
00151             io=0
00152             if (this%JDN<0) io=-1
00153         end if
00154
00155     end subroutine setjdn
00156     !+++++++++++++++++++++++++++++++++++++++++++++++
00164     pure function getjuliandaynumber(this) Result(JDN)
00165         class(ttime), intent(in) :: this
00166         integer(kind=4) :: jdn
00167
00168         jdn=this%JDN
00169
00170     end function getjuliandaynumber
00171     !+++++++++++++++++++++++++++++++++++++++++++++++++
00180     pure subroutine setgregoriandatefromjdn(this, IO)
00181         class(ttime), intent(inout) :: this
00182         integer,intent(out),optional :: io
00183
00184         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
00185         integer(kind=4) :: f, e, g, h
00186
00187         if (present(io)) then
00188             io=0
00189             if (this%JDN<0) io=-1
00190             if (io<0) return
00191         end if
00192         f=this%JDN + j +int((int((4*this%JDN + b)/146097)*3)/4) + c
00193         e=r*f + v
00194         g=int(mod(e,p)/r)
00195         h=u*g+w
00196         this%day=int((mod(h,s))/u)+1
00197         this%month=mod(int(h/s)+m,n)+1
00198         this%year=int(e/p)-y+int((n+m-this%month)/n)
00199
00200     end subroutine setgregoriandatefromjdn
00201     !+++++++++++++++++++++++++++++++++++++++++++++++++
00207     pure subroutine copy(this,from)
00208         class(ttime), intent(inout) :: this
00209         class(ttime), intent(in) :: from
00210
00211         this%year=from%year
00212         this%month=from%month
00213         this%day=from%day
00214         this%daysecs=from%daysecs
00215         this%JDN=from%JDN
00216
00217     end subroutine copy
00218     !+++++++++++++++++++++++++++++++++++++++++++++++++
00230     pure function add(this,that) Result(Total)
00231         class(ttime), intent(in) :: this, that
00232         Type(ttime) :: total
00233
00234         integer :: overflow, ios
00235         integer(kind=4) :: days
00236
00237         total%daysecs=this%daysecs+that%daysecs
00238         overflow=0
00239         if (total%daysecs>60.0) then
00240             overflow=int((total%daysecs - modulo(total%daysecs,60.0))/60.0)
00241             total%daysecs=modulo(total%daysecs,60.0)
```

```
00242         end if
00243         ! now using Julian Day Numbers:
00244         days=this%GetJulianDayNumber()+that%GetJulianDayNumber()+overflow
00245         call total%SetJDN(days,io=ios) ! this also sets the days, months and years
00246
00247     end function add
00248     !+++++++++++++++++++++++++++++++++++++++++++++
00258     pure function subtract(this,that) Result(Total)
00259         class(ttime), intent(in) :: this, that
00260         Type(ttime) :: total
00261
00262         integer(kind=4) :: overflow, days
00263         integer :: ios
00264         !Using julian day numbers and day seconds, this gets a bit more simple
00265
00266         total%daysecs=this%daysecs-that%daysecs
00267         overflow=0
00268         do while (total%daysecs<0.0)
00269             overflow=overflow+1
00270             total%daysecs=total%daysecs+86400.0
00271         end do
00272         days=this%GetJulianDayNumber()-that%GetJulianDayNumber()-overflow
00273         call total%SetJDN(days,ios)
00274
00275     end function subtract
00276     !+++++++++++++++++++++++++++++++++++++++++++++
00284     pure function isleapyear(this) Result(Leap)
00285         class(ttime), intent(in) :: this
00286         logical :: leap
00287
00288         leap=.false.
00289         if (mod(this%year,4)==0) then
00290             leap=.true.
00291             if (mod(this%year,100)==0) then
00292                 leap=.false.
00293                 if (mod(this%year,400)==0) leap=.true.
00294             end if
00295         end if
00296
00297     end function isleapyear
00298     !+++++++++++++++++++++++++++++++++++++++++++++
00313     pure function gettimestring(this,fmt) Result(TS)
00314     use, intrinsic :: iso_fortran_env
00315         class(ttime), intent(in) :: this
00316         character(len=*), intent(in), optional :: fmt
00317         character(len=255) :: ts
00318
00319         integer, parameter :: dp = real64
00320         integer:: h, m
00321         real :: s
00322         real(dp) :: fullt
00323         character(len=50) :: fmtstr
00324
00325
00326         s=mod(this%daysecs,60.0)
00327         m=mod(int((this%daysecs-s)/60),60)
00328         h=int(this%daysecs/3600)
00329         fmtstr="full"
00330         if (present(fmt)) fmtstr=fmt
00331
00332         select case(trim(adjustl(fmtstr)))
00333             case('full')
00334                 write(ts,'(2(I2,A),I4,2(A,I2),A,F6.3)') this%day,"/",this%month,"/",this%year,"
      ",h,":",m,":",s
00335             case('date')
00336                 write(ts,'(2(I2,A),I4)') this%day,"/",this%month,"/",this%year
00337             case('time')
00338                 write(ts,'(2(I2,A),F6.3)') h,":",m,":",s
00339             case('days')
00340                 fullt=this%GetJulianDayNumber()*1.0_dp + (this%daysecs/86400.0_dp)
00341                 write(ts,'(F20.8,A)') fullt," days"
00342             case('hours')
00343                 fullt=(this%GetJulianDayNumber()*1.0_dp + (this%daysecs/86400.0_dp))*24.0_dp
00344                 write(ts,'(F20.8,A)') fullt," hours"
00345             case('seconds')
00346                 fullt=(this%GetJulianDayNumber()*86400.0_dp + this%daysecs)
00347                 write(ts,'(F20.8,A)') fullt," secs"
00348             case default
00349                 write(ts,'(2(I2,A),I4,2(A,I2),A,F6.3)') this%day,"/",this%month,"/",this%year,"
      ",h,":",m,":",s
00350         end select
00351
00352     end function gettimestring
00353     !+++++++++++++++++++++++++++++++++++++++++++++
00359     pure function gettimeseconds(this) Result(sec)
00360     use, intrinsic :: iso_fortran_env
00361         class(ttime), intent(in) :: this
```

```
00362         integer, parameter :: dp = real64
00363         real(dp) :: sec
00364
00365     sec=this%GetJulianDayNumber()*86400.0_dp + this%daysecs
00366
00367     end function gettimeseconds
00368     !++++++++++++++++++++++++++++++++++++++++++
00375     subroutine destructor(this)
00376         type(ttime) :: this
00377
00378     end subroutine destructor
00379
00380
00381 end module
```

6.3 TimerClass.f03 File Reference

Data Types

- interface timerclass::ttimer
- interface timerclass::ttimer

Modules

- module timerclass

  Timer class for practical timing.

Functions/Subroutines

- pure type(ttimer) function timerclass::constructor ()

  Constructor for the TTimer instances.
- integer function timerclass::start (this)

  Start the timer (clean start, everything is reset). If the timer was already running it is reset first.
- integer function timerclass::resume (this)

  Restart the timer after a pause. If the timer is not paused, nothing will happen and TS=-1 is returned.
- integer function timerclass::addtimeflag (this)

  Allows for the introduction of an additional time-stamp without changing the timer-status (running/paused). If the Timer is stopped nothing will happen, and -1 is returned.
- integer function timerclass::interrupt (this, IO)

  Puts the timer on hold. If the timer was not running nothing will happen. The optional IO parameter will return an error code.
  IO values:
- integer function timerclass::stoptimer (this, IO)

  Stops the timer. If the timer was not running nothing will happen. The optional IO parameter will return an error code.
  IO values:
- pure subroutine timerclass::reset (this)

  Start the timer. If the timer was already running it is reset first.
- integer function timerclass::addtimestamp (this)

  Add a timestamp to a running timer, 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

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!MIT License
!
!Copyright (c) 2019 Danny Vanpoucke, https://dannyvanpoucke.be
!
!Permission is hereby granted, free of charge, to any person obtaining a copy
!of this software and associated documentation files (the "Software"), to deal
!in the Software without restriction, including without limitation the rights
!to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
!copies of the Software, and to permit persons to whom the Software is
!furnished to do so, subject to the following conditions:
!
!The above copyright notice and this permission notice shall be included in all
!copies or substantial portions of the Software.
!
!THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
!IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
!FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
!AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
!LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
!OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
!SOFTWARE.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!




!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
module timerclass
    use timeclass
    implicit none


    type, public :: ttimer
        private
        integer :: ntimes
        integer :: maxtimes
        Type(ttime), allocatable :: timestamps(:)
        logical, allocatable :: timedinterval(:)
        logical :: running
        logical :: paused
        logical :: stopped
    contains
        private
        procedure, pass(this), public :: start
        procedure, pass(this), public :: interrupt
        procedure, pass(this), public :: resume
        procedure, pass(this), public :: addtimeflag
        procedure, pass(this), public :: stoptimer
        procedure, pass(this), public :: reset
        procedure, pass(this), public :: printelapsedtimereport
        procedure, pass(this), public :: getelapsedtimestring
        procedure, pass(this)         :: getelapsedtime_total
        procedure, pass(this)         :: getelapsedtime_steps
        procedure, pass(this)         :: copy
        procedure, pass(this), private:: addtimestamp
        generic, public :: assignment(=) => copy
        generic, public :: getelapsedtime => getelapsedtime_total, getelapsedtime_steps
        final :: destructor
    end type ttimer

    ! This is the only way a constructor can be created, as no "initial" exists
    interface ttimer
        module procedure constructor
    end interface ttimer

contains
    !++++++++++++++++++++++++++++++++++++++++++++++
    pure function constructor()Result(Timer)
        Type(ttimer) :: timer
```

```
00087
00088     timer%ntimes=0
00089     timer%maxtimes=10
00090     allocate(timer%timestamps(1:10))
00091     allocate(timer%timedInterval(1:10))
00092     timer%timedInterval(:)=.false.
00093     timer%running=.false.
00094     timer%paused=.false.
00095     timer%stopped=.false.
00096
00097     end function constructor
00098     !+++++++++++++++++++++++++++++++++++++++++++++
00105     function start(this) Result(TS)
00106         class(ttimer), intent(inout) :: this
00107         integer :: ts
00108
00109         if (this%running) call this%reset()
00110         this%running=.true.
00111         ts=this%AddTimestamp()
00112         this%ntimes=ts
00113         this%timedInterval(ts)=.true. ! this timer-interval is accounted
00114
00115     end function start
00116     !+++++++++++++++++++++++++++++++++++++++++++++
00123     function resume(this) Result(TS)
00124         class(ttimer), intent(inout) :: this
00125         integer :: ts
00126
00127         ts=-1
00128         if (this%paused) then
00129             this%running=.true.
00130             this%paused=.false.
00131             ts=this%AddTimestamp()
00132             this%ntimes=ts
00133             this%timedInterval(ts)=.true. ! this timer-interval is accounted
00134         end if
00135
00136     end function resume
00137     !+++++++++++++++++++++++++++++++++++++++++++++
00145     function addtimeflag(this) Result(TS)
00146         class(ttimer), intent(inout) :: this
00147         integer :: ts
00148
00149         ts=-1
00150         if (this%running.or.this%paused) then
00151             ts=this%AddTimestamp()
00152             this%ntimes=ts
00153             this%timedInterval(ts)=this%running ! is this timer-interval accounted?
00154         end if
00155
00156     end function addtimeflag
00157     !+++++++++++++++++++++++++++++++++++++++++++++
00172     function interrupt(this,IO) Result(TS)
00173         class(ttimer), intent(inout) :: this
00174         integer, intent(out), optional :: io
00175         integer :: ts
00176
00177         ts=-1
00178         if (this%running) then
00179             if ((.not.(this%stopped)).and.(.not.(this%paused))) then
00180                 if (present(io)) io=0
00181                 ts=this%AddTimestamp()
00182                 this%paused=.true.
00183                 this%timedInterval(ts)=.false. ! as the timer is pauzed, the following interval should
    not be accounted
00184             else
00185                 if (present(io)) io=-2
00186             end if
00187         else
00188             if (present(io)) io=-1
00189         end if
00190
00191     end function interrupt
00192     !+++++++++++++++++++++++++++++++++++++++++++++
00207     function stoptimer(this,IO) Result(TS)
00208         class(ttimer), intent(inout) :: this
00209         integer, intent(out), optional :: io
00210         integer :: ts
00211
00212         ts=-1
00213         if (this%running) then
00214             if (.not.(this%stopped)) then
00215                 if (present(io)) io=0
00216                 ts=this%AddTimestamp()
00217                 this%stopped=.true.
00218                 this%timedInterval(ts)=.false. ! as the timer is stopped, the following interval
    should not be accounted
```

```
00219             else
00220                 if (present(io)) io=-2
00221             end if
00222         else
00223             if (present(io)) io=-1
00224         end if
00225
00226     end function stoptimer
00227     !+++++++++++++++++++++++++++++++++++++++++++++
00233     pure subroutine reset(this)
00234         class(ttimer), intent(inout) :: this
00235
00236         this%ntimes=0
00237         this%maxtimes=10
00238         if (allocated(this%timestamps)) deallocate(this%timestamps)
00239         allocate(this%timestamps(1:10))
00240         if (allocated(this%timedInterval)) deallocate(this%timedInterval)
00241         allocate(this%timedInterval(1:10))
00242         this%timedInterval(:)=.false.
00243         this%running=.false.
00244         this%paused=.false.
00245         this%stopped=.false.
00246
00247     end subroutine reset
00248     !+++++++++++++++++++++++++++++++++++++++++++++
00256     function addtimestamp(this)Result(TS)
00257         class(ttimer), intent(inout) :: this
00258         integer:: ts
00259
00260         type(ttime), allocatable :: tmp(:)
00261         logical, allocatable :: tmpl(:)
00262
00263         if (this%running) then
00264             ts=this%ntimes+1
00265             if (ts>this%maxtimes) then ! we need to extend the arrays
00266                 allocate(tmp(1:this%maxtimes))
00267                 allocate(tmpl(1:this%maxtimes))
00268                 tmp(1:this%maxtimes)=this%timestamps(1:this%maxtimes)
00269                 tmpl(1:this%maxtimes)=this%timedInterval(1:this%maxtimes)
00270                 this%maxtimes=this%maxtimes+10
00271                 if (allocated(this%timestamps)) deallocate(this%timestamps)
00272                 allocate(this%timestamps(1:this%maxtimes))
00273                 if (allocated(this%timedInterval)) deallocate(this%timedInterval)
00274                 allocate(this%timedInterval(1:this%maxtimes))
00275                 this%timestamps(1:this%maxtimes-10)=tmp(1:this%maxtimes-10)
00276                 this%timedInterval(1:this%maxtimes-10)=tmpl(1:this%maxtimes-10)
00277                 this%timedInterval(this%maxtimes-10:this%maxtimes)=.false.
00278                 deallocate(tmp)
00279                 deallocate(tmpl)
00280             end if
00281             this%timestamps(ts)=ttime()
00282             call this%timestamps(ts)%SetTime()
00283             this%ntimes=ts
00284         else
00285             ts=-1
00286         end if
00287
00288     end function addtimestamp
00289     !+++++++++++++++++++++++++++++++++++++++++++++
00296     pure function getelapsedtime_total(this,INCL_PAUSE) Result(sec)
00297     use, intrinsic :: iso_fortran_env
00298         class(ttimer), intent(in) :: this
00299         logical, intent(in), optional :: incl_pause
00300         integer, parameter :: dp = real64
00301         real(dp) :: sec
00302
00303         logical :: pauze
00304
00305     pauze=.false.
00306     if (present(incl_pause)) pauze=incl_pause
00307
00308     sec=this%GetElapsedTime_steps(1,this%maxtimes,incl_pause=pauze)
00309
00310     end function getelapsedtime_total
00311     !+++++++++++++++++++++++++++++++++++++++++++++
00324     pure function getelapsedtime_steps(this,Tstart,Tend,INCL_PAUSE) Result(sec)
00325     use, intrinsic :: iso_fortran_env
00326         class(ttimer), intent(in) :: this
00327         integer, intent(in) :: tstart, tend
00328         logical, intent(in), optional :: incl_pause
00329         integer, parameter :: dp = real64
00330         real(dp) :: sec
00331
00332         type(ttime) :: elap, tmp
00333         integer :: t1, t2, nr
00334
00335     if ((tstart<1).or.(tend<1).or.(tstart>this%ntimes).or.(tend>this%ntimes)) then
```

```
00336         sec=-1.0
00337         return
00338     end if
00339
00340     if (tstart>tend) then
00341         t1=tend
00342         t2=tstart
00343     else
00344         t1=tstart
00345         t2=tend
00346     end if
00347
00348     elap=this%timestamps(t2)-this%timestamps(t1)
00349     sec=elap%GetTimeSeconds()
00350     if (present(incl_pause)) then
00351         if (incl_pause) then ! pauses should be excluded, so we subtract the again
00352             do nr=1,this%ntimes-1
00353                 if (.not.this%timedInterval(nr)) then
00354                     tmp=this%timestamps(nr+1)-this%timestamps(nr)
00355                     sec=sec-tmp%GetTimeSeconds()
00356                 end if
00357
00358             end do
00359         end if
00360     end if
00361
00362     end function getelapsedtime_steps
00363     !++++++++++++++++++++++++++++++++++++++++++++++
00378     pure function getelapsedtimestring(this,TSTART, TSTOP, INCL_PAUSE, FMT) Result(str)
00379     use, intrinsic :: iso_fortran_env
00380         class(ttimer), intent(in) :: this
00381         integer, intent(in), optional :: tstart, tstop
00382         logical, intent(in), optional :: incl_pause
00383         character(len=*), intent(in), optional :: fmt
00384         character(len=50) :: str
00385
00386         integer, parameter :: dp = real64
00387         real(dp) :: sec
00388         integer :: t1, t2, nr, hour, day, minute
00389         character(len=4) :: opt
00390         character(len=255) :: line
00391
00392         t1=1
00393         t2=this%ntimes
00394         if(present(tstart)) then
00395             if ((tstart>0).and.(tstart<=this%ntimes)) t1=tstart
00396         end if
00397         if(present(tstop)) then
00398             if ((tstop>0).and.(tstop<=this%ntimes)) t2=tstop
00399         end if
00400         if (t1>t2) then
00401             nr=t1
00402             t1=t2
00403             t2=nr
00404         end if
00405
00406         sec=this%GetElapsedTime(t1,t2,incl_pause)
00407         !now transform to a string
00408         opt='sec'
00409         if (present(fmt)) opt=trim(adjustl(fmt))
00410
00411         select case(trim(adjustl(opt)))
00412             case ('sec')
00413                 write(line,'(F30.3,A)') sec," secs "
00414             case ('hour')
00415                 write(line,'(F30.3,A)') sec/3600.0_dp," hours "
00416             case ('day')
00417                 write(line,'(F30.3,A)') sec/86400.0_dp," days "
00418             case ('hms')
00419                 hour=floor(sec/3600.0_dp)
00420                 sec=sec-(hour*3600.0_dp)
00421                 minute=floor(sec/60.0_dp)
00422                 sec=sec-(minute*60.0_dp)
00423                 write(line,'(I0,A,I2,A,F6.3)') hour," h ",minute," min ",sec," secs "
00424             case ('dhms')
00425                 day=floor(sec/86400.0_dp)
00426                 sec=sec-(day*864000_dp)
00427                 hour=floor(sec/3600.0_dp)
00428                 sec=sec-(hour*3600.0_dp)
00429                 minute=floor(sec/60.0_dp)
00430                 sec=sec-(minute*60.0_dp)
00431                 write(line,'(I0,A,2(I2,A),F6.3)') day," days ",hour," h ",minute," min ",sec," secs "
00432         end select
00433         write(str,'(3A)') " ",trim(adjustl(line))," "
00434
00435     end function getelapsedtimestring
00436     !++++++++++++++++++++++++++++++++++++++++++++++
```

```
00446     subroutine printelapsedtimereport(this, message, Tstart, Tstop, UN, INCL_PAUSE)
00447     use, intrinsic :: iso_fortran_env
00448         class(ttimer), intent(inout) :: this
00449         character(len=*), intent(in) :: message
00450         integer, intent(in) :: Tstart, Tstop
00451         integer, intent(in) :: UN
00452         logical, intent(in), optional :: INCL_PAUSE
00453
00454         character(len=50) :: line
00455         integer, parameter :: dp = real64
00456         real(dp) :: sec
00457         integer :: ih, im
00458
00459         write(un,"(2A)",advance='NO') trim(message)," : "
00460         line=this%GetElapsedTimeString(tstart,tstop,incl_pause,'sec')
00461         write(un,"(A)") trim(adjustl(line))
00462         sec=this%GetElapsedTime(tstart,tstop,incl_pause)
00463
00464         ih=floor(sec/3600.0_dp)
00465         sec=sec-dble(ih)*3600.0_dp
00466         im=floor(sec/60.0_dp)
00467         sec=sec-dble(im)*60.0_dp
00468
00469         write (un,'(I8,A)') ih," hours"
00470         write (un,'(I8,A)') im," minutes"
00471         write (un,'(F8.3,A)') sec," seconds"
00472
00473     end subroutine printelapsedtimereport
00474     !+++++++++++++++++++++++++++++++++++++++++++++++++
00480     pure subroutine copy(this,from)
00481         class(ttimer), intent(inout) :: this
00482         class(ttimer), intent(in) :: from
00483
00484     integer :: nr
00485
00486     this%maxtimes=from%maxtimes
00487     this%ntimes=from%ntimes
00488     this%paused=from%paused
00489     this%running=from%running
00490     this%stopped=from%stopped
00491     allocate(this%timedInterval(1:this%maxtimes))
00492     this%timedInterval(1:this%maxtimes)=from%timedInterval(1:this%maxtimes)
00493     allocate(this%timestamps(1:this%maxtimes))
00494     do nr=1, this%ntimes
00495         this%timestamps(nr)=from%timestamps(nr)
00496     end do
00497
00498     end subroutine copy
00499     !+++++++++++++++++++++++++++++++++++++++++++++++++
00503     subroutine destructor(this)
00504         Type(ttimer) :: this
00505
00506     if (allocated(this%timestamps)) deallocate(this%timestamps)
00507     if (allocated(this%timedInterval)) deallocate(this%timedInterval)
00508
00509     end subroutine destructor
00510
00511
00512 end module timerclass
```

Index