

# My Project

Generated by Doxygen 1.8.6

Wed Jul 20 2016 09:43:03



# Contents

<b>1</b>	<b>GLT_documentation</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Module Documentation</b>	<b>5</b>
3.1	Library functions . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Function Documentation . . . . .	5
3.1.2.1	__attribute__ . . . . .	5
3.1.2.2	__attribute__ . . . . .	5
3.1.2.3	glt_finalize . . . . .	5
3.1.2.4	glt_init . . . . .	6
3.2	Barrier functions . . . . .	7
3.2.1	Detailed Description . . . . .	7
3.2.2	Function Documentation . . . . .	7
3.2.2.1	glt_barrier_create . . . . .	7
3.2.2.2	glt_barrier_free . . . . .	7
3.2.2.3	glt_barrier_wait . . . . .	7
3.3	Condition functions . . . . .	8
3.3.1	Detailed Description . . . . .	8
3.3.2	Function Documentation . . . . .	8
3.3.2.1	glt_cond_broadcast . . . . .	8
3.3.2.2	glt_cond_create . . . . .	8
3.3.2.3	glt_cond_free . . . . .	8
3.3.2.4	glt_cond_signal . . . . .	8
3.3.2.5	glt_cond_wait . . . . .	9
3.4	Mutex functions . . . . .	10
3.4.1	Detailed Description . . . . .	10
3.4.2	Function Documentation . . . . .	10
3.4.2.1	glt_mutex_create . . . . .	10
3.4.2.2	glt_mutex_free . . . . .	10

3.4.2.3	<a href="#">glt_mutex_lock</a>	10
3.4.2.4	<a href="#">glt_mutex_trylock</a>	10
3.4.2.5	<a href="#">glt_mutex_unlock</a>	11
3.5	<a href="#">Work-units functions</a>	12
3.5.1	<a href="#">Detailed Description</a>	12
3.5.2	<a href="#">Function Documentation</a>	12
3.5.2.1	<a href="#">glt_tasklet_cancel</a>	12
3.5.2.2	<a href="#">glt_tasklet_create</a>	13
3.5.2.3	<a href="#">glt_tasklet_create_to</a>	13
3.5.2.4	<a href="#">glt_tasklet_join</a>	13
3.5.2.5	<a href="#">glt_tasklet_malloc</a>	13
3.5.2.6	<a href="#">glt_tasklet_self</a>	13
3.5.2.7	<a href="#">glt_ult_cancel</a>	15
3.5.2.8	<a href="#">glt_ult_create</a>	15
3.5.2.9	<a href="#">glt_ult_create_to</a>	15
3.5.2.10	<a href="#">glt_ult_exit</a>	15
3.5.2.11	<a href="#">glt_ult_get_id</a>	15
3.5.2.12	<a href="#">glt_ult_join</a>	16
3.5.2.13	<a href="#">glt_ult_malloc</a>	16
3.5.2.14	<a href="#">glt_ult_migrate_self_to</a>	16
3.5.2.15	<a href="#">glt_ult_self</a>	16
3.5.2.16	<a href="#">glt_workunit_get_thread_id</a>	16
3.5.2.17	<a href="#">glt_yield</a>	17
3.5.2.18	<a href="#">glt_yield_to</a>	17
3.6	<a href="#">Timer functions</a>	18
3.6.1	<a href="#">Detailed Description</a>	18
3.6.2	<a href="#">Function Documentation</a>	18
3.6.2.1	<a href="#">glt_get_wtime</a>	18
3.6.2.2	<a href="#">glt_timer_create</a>	18
3.6.2.3	<a href="#">glt_timer_free</a>	18
3.6.2.4	<a href="#">glt_timer_get_secs</a>	19
3.6.2.5	<a href="#">glt_timer_start</a>	19
3.6.2.6	<a href="#">glt_timer_stop</a>	19
3.7	<a href="#">Util functions</a>	20
3.7.1	<a href="#">Detailed Description</a>	20
3.7.2	<a href="#">Function Documentation</a>	20
3.7.2.1	<a href="#">glt_get_num_threads</a>	20
3.7.2.2	<a href="#">glt_get_thread_num</a>	20
3.8	<a href="#">Scheduler functions</a>	21
3.8.1	<a href="#">Detailed Description</a>	21

3.8.2	Function Documentation	21
3.8.2.1	glt_scheduler_create_basic	21
3.8.2.2	glt_scheduler_config_free	22
3.8.2.3	glt_scheduler_create	22
3.8.2.4	glt_scheduler_exit	22
3.8.2.5	glt_scheduler_finish	22
3.8.2.6	glt_scheduler_free	22
3.8.2.7	glt_scheduler_get_data	23
3.8.2.8	glt_scheduler_get_size	23
3.8.2.9	glt_scheduler_get_total_size	23
3.8.2.10	glt_scheduler_has_to_stop	23
3.8.2.11	glt_scheduler_set_data	23
3.9	Key functions	25
3.9.1	Detailed Description	25
3.9.2	Function Documentation	25
3.9.2.1	glt_key_create	25
3.9.2.2	glt_key_free	25
3.9.2.3	glt_key_get	25
3.9.2.4	glt_key_set	25
	<b>Index</b>	<b>27</b>



## **Chapter 1**

## **GLT\_documentation**





## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

Library functions . . . . .	5
Barrier functions . . . . .	7
Condition functions . . . . .	8
Mutex functions . . . . .	10
Work-units functions . . . . .	12
Timer functions . . . . .	18
Util functions . . . . .	20
Scheduler functions . . . . .	21
Key functions . . . . .	25



## Chapter 3

# Module Documentation

### 3.1 Library functions

#### Functions

- void `__attribute__` ((constructor)) `glt_start`(void)  
*Entry point for the GLT dynamic library.*
- void `__attribute__` ((destructor)) `glt_end`(void)  
*Ending point for the GLT dynamic library.*
- void `glt_init` (int argc, char \*argv[])  
*GLT initialization function.*
- void `glt_finalize` ()  
*GLT finalization function.*

#### 3.1.1 Detailed Description

These functions start/stop and open/close the underlying GLT libraries. They are used in dynamic and static implementations.

#### 3.1.2 Function Documentation

##### 3.1.2.1 void `__attribute__` ( (constructor) )

Entry point for the GLT dynamic library.

`glt_start()` is the first called function when the GLT dynamic library is loaded

##### 3.1.2.2 void `__attribute__` ( (destructor) )

Ending point for the GLT dynamic library.

`glt_end()` is the last called function when the GLT dynamic library is unloaded

##### 3.1.2.3 void `glt_finalize` ( )

GLT finalization function.

`glt_finalize()` destroys the GLT environment. It is not mandatory and should be the last GLT function call.

#### 3.1.2.4 void glt\_init ( int *argc*, char \* *argv*[] )

GLT initialization function.

`glt_init()` sets the GLT environment up. It is mandatory and needs to be the first GLT function call.

##### Parameters

in	<i>argc</i>	
in	<i>argv</i>	

## 3.2 Barrier functions

### Functions

- void [glt\\_barrier\\_create](#) (int num\_waiters, GLT\_barrier \*barrier)  
*Creates a barrier.*
- void [glt\\_barrier\\_free](#) (GLT\_barrier \*barrier)  
*Destroys a barrier.*
- void [glt\\_barrier\\_wait](#) (GLT\_barrier \*barrier)  
*Waits into a barrier.*

### 3.2.1 Detailed Description

These functions manage the GLT barriers for the ULTs.

### 3.2.2 Function Documentation

#### 3.2.2.1 void glt\_barrier\_create ( int num\_waiters, GLT\_barrier \* barrier )

Creates a barrier.

[glt\\_barrier\\_create\(\)](#) creates a barrier for ULTs.

##### Parameters

in	<i>num_waiters</i>	Indicates the number of ULTs requested to continue
in, out	<i>barrier</i>	Handle to newly created GLT_barrier

#### 3.2.2.2 void glt\_barrier\_free ( GLT\_barrier \* barrier )

Destroys a barrier.

[glt\\_barrier\\_free\(\)](#) destroys a barrier for ULTs.

##### Parameters

in	<i>barrier</i>	Handle to the target GLT_barrier.
----	----------------	-----------------------------------

#### 3.2.2.3 void glt\_barrier\_wait ( GLT\_barrier \* barrier )

Waits into a barrier.

[glt\\_barrier\\_wait\(\)](#) Executed by a ULT, it waits until the number of waiters is achieved.

##### Parameters

in	<i>barrier</i>	Handle to the target GLT_barrier.
----	----------------	-----------------------------------

### 3.3 Condition functions

#### Functions

- void [glt\\_cond\\_create](#) (GLT\_cond \*cond)  
*Creates a condition.*
- void [glt\\_cond\\_free](#) (GLT\_cond \*cond)  
*Destroys a condition.*
- void [glt\\_cond\\_signal](#) (GLT\_cond cond)  
*Sends a signal for a condition.*
- void [glt\\_cond\\_wait](#) (GLT\_cond cond, GLT\_mutex mutex)  
*A ULT waits in this point for a signal.*
- void [glt\\_cond\\_broadcast](#) (GLT\_cond cond)  
*Broadcast a signal for a condition.*

#### 3.3.1 Detailed Description

These functions manage the GLT conditions for the ULTs.

#### 3.3.2 Function Documentation

##### 3.3.2.1 void [glt\\_cond\\_broadcast](#) ( GLT\_cond *cond* )

Broadcast a signal for a condition.

[glt\\_cond\\_broadcast\(\)](#) broadcasts a signal for ULTs.

##### Parameters

in	<i>cond</i>	Handle to the target GLT_condition.
----	-------------	-------------------------------------

##### 3.3.2.2 void [glt\\_cond\\_create](#) ( GLT\_cond \* *cond* )

Creates a condition.

[glt\\_cond\\_create\(\)](#) creates a condition for ULTs.

##### Parameters

in, out	<i>cond</i>	Handle to newly created GLT_condition
---------	-------------	---------------------------------------

##### 3.3.2.3 void [glt\\_cond\\_free](#) ( GLT\_cond \* *cond* )

Destroys a condition.

[glt\\_cond\\_free\(\)](#) destroys a condition for ULTs.

##### Parameters

in	<i>cond</i>	Handle to the target GLT_condition.
----	-------------	-------------------------------------

##### 3.3.2.4 void [glt\\_cond\\_signal](#) ( GLT\_cond *cond* )

Sends a signal for a condition.

[glt\\_cond\\_signal\(\)](#) sends a signal for ULTs.

## Parameters

in	<i>cond</i>	Handle to the target GLT_condition.
----	-------------	-------------------------------------

3.3.2.5 void glt\_cond\_wait ( GLT\_cond *cond*, GLT\_mutex *mutex* )

A ULT waits in this point for a signal.

`glt_cond_wait()` a ULT waits at this point for a signal to access the mutex.

## Parameters

in	<i>cond</i>	Handle to the target GLT_condition.
in	<i>mutex</i>	Handle to the target GLT_mutex.

## 3.4 Mutex functions

### Functions

- void [glt\\_mutex\\_create](#) (GLT\_mutex \*mutex)  
*Creates a mutex.*
- void [glt\\_mutex\\_lock](#) (GLT\_mutex mutex)  
*Locks a mutex.*
- void [glt\\_mutex\\_unlock](#) (GLT\_mutex mutex)  
*Unlocks a mutex.*
- void [glt\\_mutex\\_free](#) (GLT\_mutex \*mutex)  
*Destroys a mutex.*
- void [glt\\_mutex\\_trylock](#) (GLT\_bool \*locked, GLT\_mutex mutex)  
*Tries to lock a mutex.*

### 3.4.1 Detailed Description

These functions manage the GLT mutexes for the ULTs.

### 3.4.2 Function Documentation

#### 3.4.2.1 void [glt\\_mutex\\_create](#) ( GLT\_mutex \* *mutex* )

Creates a mutex.

[glt\\_mutex\\_create\(\)](#) creates a mutex for ULTs.

Parameters

<i>in, out</i>	<i>mutex</i>	Handle to newly created GLT_mutex
----------------	--------------	-----------------------------------

#### 3.4.2.2 void [glt\\_mutex\\_free](#) ( GLT\_mutex \* *mutex* )

Destroys a mutex.

[glt\\_mutex\\_free\(\)](#) destroys a mutex for ULTs.

Parameters

<i>in</i>	<i>mutex</i>	Handle to the target GLT_mutex.
-----------	--------------	---------------------------------

#### 3.4.2.3 void [glt\\_mutex\\_lock](#) ( GLT\_mutex *mutex* )

Locks a mutex.

[glt\\_mutex\\_lock\(\)](#) locks (if possible) a mutex.

Parameters

<i>in</i>	<i>mutex</i>	Handle to the target GLT_mutex.
-----------	--------------	---------------------------------

#### 3.4.2.4 void [glt\\_mutex\\_trylock](#) ( GLT\_bool \* *locked*, GLT\_mutex *mutex* )

Tries to lock a mutex.

[glt\\_mutex\\_trylock\(\)](#) tries to lock a mutex.



## Parameters

in	<i>mutex</i>	Handle to the target GLT_mutex.
out	<i>locked</i>	GLT_bool with the value 1 if the mutex has been locked or 0 if it was not possible.

3.4.2.5 void glt\_mutex\_unlock ( GLT\_mutex *mutex* )

Unlocks a mutex.

`glt_mutex_unlock()` unlocks a mutex.

## Parameters

in	<i>mutex</i>	Handle to the target GLT_mutex.
----	--------------	---------------------------------

## 3.5 Work-units functions

### Functions

- GLT\_ult \* [glt\\_ult\\_malloc](#) (int number\_of\_ult)  
*ULT allocation.*
- GLT\_tasklet \* [glt\\_tasklet\\_malloc](#) (int number\_of\_tasklets)  
*ULT allocation.*
- void [glt\\_ult\\_create](#) (void(\*thread\_func)(void \*), void \*arg, GLT\_ult \*new\_ult)  
*ULT creation.*
- void [glt\\_ult\\_create\\_to](#) (void(\*thread\_func)(void \*), void \*arg, GLT\_ult \*new\_ult, int dest)  
*ULT creation in a given destination.*
- void [glt\\_tasklet\\_create](#) (void(\*thread\_func)(void \*), void \*arg, GLT\_tasklet \*new\_ult)  
*Tasklet creation.*
- void [glt\\_tasklet\\_create\\_to](#) (void(\*thread\_func)(void \*), void \*arg, GLT\_tasklet \*new\_ult, int dest)  
*Tasklet creation.*
- void [glt\\_yield](#) ()  
*ULT yields to another ready ULT.*
- void [glt\\_yield\\_to](#) (GLT\_ult ult)  
*ULT yields to an specific ULT.*
- void [glt\\_ult\\_join](#) (GLT\_ult \*ult)  
*Joins an specific ULT.*
- void [glt\\_tasklet\\_join](#) (GLT\_tasklet \*tasklet)  
*Joins an specific Tasklet.*
- void [glt\\_ult\\_get\\_id](#) (GLT\_ult\_id \*id, GLT\_ult ult)  
*Return the unique id of a ULT.*
- void [glt\\_workunit\\_get\\_thread\\_id](#) (GLT\_thread\_id \*id)  
*Return the unique id of a thread.*
- void [glt\\_ult\\_migrate\\_self\\_to](#) (GLT\_thread\_id id)  
*Migrates the current ULT to another thread ready queue.*
- void [glt\\_ult\\_self](#) (GLT\_ult \*ult)  
*Obtains the current ULT handle.*
- void [glt\\_tasklet\\_self](#) (GLT\_tasklet \*tasklet)  
*Obtains the current Tasklet handle.*
- void [glt\\_ult\\_cancel](#) (GLT\_ult ult)  
*Cancels an specific ULT.*
- void [glt\\_tasklet\\_cancel](#) (GLT\_tasklet tasklet)  
*Cancels an specific Tasklet.*
- void [glt\\_ult\\_exit](#) ()  
*Exits the current ULT.*

### 3.5.1 Detailed Description

These functions create, map, schedule, join, and execute work-units.

### 3.5.2 Function Documentation

#### 3.5.2.1 void [glt\\_tasklet\\_cancel](#) ( GLT\_tasklet *tasklet* )

Cancels an specific Tasklet.

[glt\\_tasklet\\_cancel\(\)](#) cancels a given GLT\_tasklet.

## Parameters

in	<i>tasklet</i>	Handle to the target GLT_tasklet.
----	----------------	-----------------------------------

3.5.2.2 void glt\_tasklet\_create ( void(\*)(void \*) *thread\_func*, void \* *arg*, GLT\_tasklet \* *new\_ult* )

Tasklet creation.

`glt_tasklet_create()` creates a GLT\_tasklet.

## Parameters

in	<i>thread_func</i>	Is the function pointer to be executed by the GLT_tasklet.
in	<i>arg</i>	Are the arguments for <i>thread_func</i> .
out	<i>new_ult</i>	Handle to a newly created GLT_tasklet.

3.5.2.3 void glt\_tasklet\_create\_to ( void(\*)(void \*) *thread\_func*, void \* *arg*, GLT\_tasklet \* *new\_ult*, int *dest* )

Tasklet creation.

`glt_tasklet_create()` creates a GLT\_tasklet.

## Parameters

in	<i>thread_func</i>	Is the function pointer to be executed by the GLT_tasklet.
in	<i>arg</i>	Are the arguments for <i>thread_func</i> .
out	<i>new_ult</i>	Handle to a newly created GLT_tasklet.
in	<i>dest</i>	Number of the GLT_thread that should execute the newly created GLT_tasklet.

3.5.2.4 void glt\_tasklet\_join ( GLT\_tasklet \* *tasklet* )

Joins an specific Tasklet.

`glt_tasklet_join()` joins a given GLT\_tasklet.

## Parameters

in	<i>tasklet</i>	Handle to the target GLT_tasklet.
----	----------------	-----------------------------------

3.5.2.5 GLT\_tasklet\* glt\_tasklet\_malloc ( int *number\_of\_tasklets* )

ULT allocation.

`glt_tasklet_malloc()` allocates memory for a given number of GLT\_tasklet.

## Parameters

in	<i>number_of_tasklets</i>	Indicates the total number of GLT_tasklets that needs to be allocated.
----	---------------------------	--

## Returns

The pointer to the allocated memory.

3.5.2.6 void glt\_tasklet\_self ( GLT\_tasklet \* *tasklet* )

Obtains the current Tasklet handle.

`glt_tasklet_self()` returns the current `GLT_tasklet` handler.

## Parameters

out	<i>tasklet</i>	Handler of the the current GLT_tasklet.
-----	----------------	---

## 3.5.2.7 void glt\_ult\_cancel ( GLT\_ult ult )

Cancels an specific ULT.

`glt_ult_cancel()` cancels a given GLT\_ult.

## Parameters

in	<i>ult</i>	Handle to the target GLT_ult.
----	------------	-------------------------------

## 3.5.2.8 void glt\_ult\_create ( void(\*)(void \*) thread\_func, void \* arg, GLT\_ult \* new\_ult )

ULT creation.

`glt_ult_create()` creates a GLT\_ult.

## Parameters

in	<i>thread_func</i>	Is the function pointer to be executed by the GLT_ult.
in	<i>arg</i>	Are the arguments for thread_func.
out	<i>new_ult</i>	Handle to a newly created GLT_ult.

## 3.5.2.9 void glt\_ult\_create\_to ( void(\*)(void \*) thread\_func, void \* arg, GLT\_ult \* new\_ult, int dest )

ULT creation in a given destination.

`glt_ult_create_to()` creates a GLT\_ult in a particular destination.

## Parameters

in	<i>thread_func</i>	Is the function pointer to be executed by the GLT_ult.
in	<i>arg</i>	Are the arguments for thread_func.
out	<i>new_ult</i>	Handle to a newly created GLT_ult.
in	<i>dest</i>	Number of the GLT_thread that should execute the newly created GLT_ult.

## 3.5.2.10 void glt\_ult\_exit ( )

Exits the current ULT.

`glt_ult_exit()` cancels the current GLT\_ult.

## 3.5.2.11 void glt\_ult\_get\_id ( GLT\_ult\_id \* id, GLT\_ult ult )

Return the unique id of a ULT.

`glt_ult_get_id()` returns the id of a given GLT\_ult.

## Parameters

in	<i>ult</i>	Handle to the target GLT_ult.
out	<i>id</i>	Identifier if the the target GLT_ult.

### 3.5.2.12 void glt\_ult\_join ( GLT\_ult \* *ult* )

Joins an specific ULT.

`glt_ult_join()` joins a given GLT\_ult.

#### Parameters

in	<i>ult</i>	Handle to the target GLT_ult.
----	------------	-------------------------------

### 3.5.2.13 GLT\_ult\* glt\_ult\_malloc ( int *number\_of\_ult* )

ULT allocation.

`glt_ult_malloc()` allocates memory for a given number of GLT\_ult.

#### Parameters

in	<i>number_of_ult</i>	Indicates the total number of GLT_ult that needs to be allocated.
----	----------------------	---

#### Returns

The pointer to the allocated memory.

### 3.5.2.14 void glt\_ult\_migrate\_self\_to ( GLT\_thread\_id *id* )

Migrates the current ULT to another thread ready queue.

`glt_ult_migrate_self_to()` moves the current GLT\_ult to another GLT\_thread ready queue.

#### Parameters

in	<i>The</i>	identifier of the the GLT_thread destination.
----	------------	---

### 3.5.2.15 void glt\_ult\_self ( GLT\_ult \* *ult* )

Obtains the current ULT handle.

`glt_ult_self()` returns the current GLT\_ult handler.

#### Parameters

out	<i>ult</i>	Handler of the the current GLT_ult.
-----	------------	-------------------------------------

### 3.5.2.16 void glt\_workunit\_get\_thread\_id ( GLT\_thread\_id \* *id* )

Return the unique id of a thread.

`glt_workunit_get_thread_id()` returns the id of the current GLT\_thread.

## Parameters

out	<i>id</i>	Identifier of the the current GLT_thread.
-----	-----------	---

## 3.5.2.17 void glt\_yield ( )

ULT yields to another ready ULT.

`glt_yield()` puts the current GLT\_ult in the ready queue and allows another ready GLT\_ult to be executed.

## 3.5.2.18 void glt\_yield\_to ( GLT\_ult ult )

ULT yields to an specific ULT.

`glt_yield_to()` puts the current GLT\_ult in the ready queue and allows an specific ready GLT\_ult to be executed.

## Parameters

in	<i>ult</i>	Handle to the target GLT_ult.
----	------------	-------------------------------

## 3.6 Timer functions

### Functions

- double [glt\\_get\\_wtime](#) ()  
*Returns the current time.*
- void [glt\\_timer\\_create](#) (GLT\_timer \*timer)  
*Creates a timer.*
- void [glt\\_timer\\_free](#) (GLT\_timer \*timer)  
*Destroys a timer.*
- void [glt\\_timer\\_start](#) (GLT\_timer timer)  
*Inits a timer.*
- void [glt\\_timer\\_stop](#) (GLT\_timer timer)  
*Stops a timer.*
- void [glt\\_timer\\_get\\_secs](#) (GLT\_timer timer, double \*secs)  
*Obtains the elapsed time.*

### 3.6.1 Detailed Description

These functions simplify the use of timers.

### 3.6.2 Function Documentation

#### 3.6.2.1 double [glt\\_get\\_wtime](#) ( )

Returns the current time.

[glt\\_get\\_wtime](#) () returns the time.

#### Returns

The time in seconds.

#### 3.6.2.2 void [glt\\_timer\\_create](#) ( GLT\_timer \* *timer* )

Creates a timer.

[glt\\_timer\\_create](#) () creates a timer.

#### Parameters

<i>in, out</i>	<i>timer</i>	Handle to newly created GLT_timer.
----------------	--------------	------------------------------------

#### 3.6.2.3 void [glt\\_timer\\_free](#) ( GLT\_timer \* *timer* )

Destroys a timer.

[glt\\_timer\\_free](#) () destroys a timer.

#### Parameters



in	<i>timer</i>	Handle to the target GLT_timer.
----	--------------	---------------------------------

#### 3.6.2.4 void glt\_timer\_get\_secs ( GLT\_timer *timer*, double \* *secs* )

Obtains the elapsed time.

`glt_timer_get_secs()` given a timer. It calculates the elapsed time in seconds.

Parameters

in	<i>timer</i>	Handle to the target GLT_timer.
out	<i>secs</i>	Seconds.

#### 3.6.2.5 void glt\_timer\_start ( GLT\_timer *timer* )

Initiates a timer.

`glt_timer_start()` initiates a timer.

Parameters

in	<i>timer</i>	Handle to the target GLT_timer.
----	--------------	---------------------------------

#### 3.6.2.6 void glt\_timer\_stop ( GLT\_timer *timer* )

Stops a timer.

`glt_timer_stop()` stops a timer.

Parameters

in	<i>timer</i>	Handle to the target GLT_timer.
----	--------------	---------------------------------

## 3.7 Util functions

### Functions

- `int glt_get_thread_num()`  
*Obtains the number of the current thread.*
- `int glt_get_num_threads()`  
*Returns the total number of threads.*

### 3.7.1 Detailed Description

These functions return values from the environment set up.

### 3.7.2 Function Documentation

#### 3.7.2.1 `int glt_get_num_threads()`

Returns the total number of threads.

`glt_get_num_threads()` returns the number threads.

#### Returns

The number of `c\ GLT_threads`.

#### 3.7.2.2 `int glt_get_thread_num()`

Obtains the number of the current thread.

`glt_get_thread_num()` returns the number of the current thread.

#### Returns

The number of the current `c\ GLT_thread`.

## 3.8 Scheduler functions

### Functions

- void [glt\\_scheduler\\_config\\_free](#) (GLT\_sched\_config \*config)  
*Destroys the scheduler configuration.*
- void [glt\\_scheduler\\_create](#) (GLT\_sched\_def \*def, int num\_threads, int \*threads\_id, GLT\_sched\_config config, GLT\_sched \*newsched)  
*Creates a new scheduler.*
- void [glt\\_scheduleduler\\_create\\_basic](#) (GLT\_sched\_predef predef, int num\_threads, int \*threads\_id, GLT\_sched\_config config, GLT\_sched \*newsched)  
*Creates a new scheduler with predefined behaviour.*
- void [glt\\_scheduler\\_free](#) (GLT\_sched \*sched)  
*Destroys a scheduler.*
- void [glt\\_scheduler\\_finish](#) (GLT\_sched sched)  
*Finalizes a scheduler.*
- void [glt\\_scheduler\\_exit](#) (GLT\_sched sched)  
*Stops a scheduler.*
- void [glt\\_scheduler\\_has\\_to\\_stop](#) (GLT\_sched sched, GLT\_bool \*stop)  
*Requires to a scheduler to stop.*
- void [glt\\_scheduler\\_set\\_data](#) (GLT\_sched sched, void \*data)  
*Sets new data to a scheduler.*
- void [glt\\_scheduler\\_get\\_data](#) (GLT\_sched sched, void \*\*data)  
*gets data from a scheduler.*
- void [glt\\_scheduler\\_get\\_size](#) (GLT\_sched sched, size\_t \*size)  
*gets the current size from the scheduler.*
- void [glt\\_scheduler\\_get\\_total\\_size](#) (GLT\_sched sched, size\_t \*size)  
*gets the total size from the scheduler.*

### 3.8.1 Detailed Description

These functions manages the configurable scheduler (just with Argobots).

### 3.8.2 Function Documentation

**3.8.2.1** void [glt\\_scheduleduler\\_create\\_basic](#) ( GLT\_sched\_predef predef, int num\_threads, int \* threads\_id, GLT\_sched\_config config, GLT\_sched \* newsched )

Creates a new scheduler with predefined behaviour.

[glt\\_scheduleduler\\_create\\_basic\(\)](#) creates a new scheduler for some threads with a predefined behaviour.

#### Parameters

in	<i>def</i>	Handle of the target c\ GLT_sched_predef.
in	<i>num_threads</i>	number of GLT_thread affected by this scheduler.
in	<i>threads_id</i>	pointer to an array of c\ GLT_threads_id.
in	<i>config</i>	Handle of the target c\ GLT_sched_config.

out	<i>newsched</i>	Handle of new c\ GLT_sched.
-----	-----------------	-----------------------------

### 3.8.2.2 void glt\_scheduler\_config\_free ( GLT\_sched\_config \* config )

Destroys the scheduler configuration.

[glt\\_scheduler\\_config\\_free\(\)](#) deletes the scheduler configuration.

Parameters

in	<i>config</i>	Handle of the target c\ GLT_sched_config.
----	---------------	---

### 3.8.2.3 void glt\_scheduler\_create ( GLT\_sched\_def \* def, int num\_threads, int \* threads\_id, GLT\_sched\_config config, GLT\_sched \* newsched )

Creates a new scheduler.

[glt\\_scheduler\\_create\(\)](#) creates a new scheduler for some threads.

Parameters

in	<i>def</i>	Handle of the target c\ GLT_sched_def.
in	<i>num_threads</i>	number of GLT_thread affected by this scheduler.
in	<i>threads_id</i>	pointer to an array of c\ GLT_threads_id.
in	<i>config</i>	Handle of the target c\ GLT_sched_config.
out	<i>newsched</i>	Handle of new c\ GLT_sched.

### 3.8.2.4 void glt\_scheduler\_exit ( GLT\_sched sched )

Stops a scheduler.

[glt\\_scheduler\\_exit\(\)](#) Stops a scheduler.

Parameters

in	<i>sched</i>	Handle of the target c\ GLT_sched.
----	--------------	------------------------------------

### 3.8.2.5 void glt\_scheduler\_finish ( GLT\_sched sched )

Finalizes a scheduler.

[glt\\_scheduler\\_finish\(\)](#) finalizes a scheduler.

Parameters

in	<i>sched</i>	Handle of the target c\ GLT_sched.
----	--------------	------------------------------------

### 3.8.2.6 void glt\_scheduler\_free ( GLT\_sched \* sched )

Destroys a scheduler.

[glt\\_scheduler\\_free\(\)](#) destroys a scheduler.

## Parameters

in	<i>sched</i>	Handle of the target c\ GLT_sched.
----	--------------	------------------------------------

3.8.2.7 void glt\_scheduler\_get\_data ( GLT\_sched *sched*, void \*\* *data* )

gets data from a scheduler.

`glt_scheduler_get_data()` gets data from a scheduler.

## Parameters

in	<i>sched</i>	Handle of the target c\ GLT_sched.
out	<i>data</i>	obtained.

3.8.2.8 void glt\_scheduler\_get\_size ( GLT\_sched *sched*, size\_t \* *size* )

gets the current size from the scheduler.

`glt_scheduler_get_size()` gets size from a scheduler.

## Parameters

in	<i>sched</i>	Handle of the target c\ GLT_sched.
out	<i>size</i>	obtained.

3.8.2.9 void glt\_scheduler\_get\_total\_size ( GLT\_sched *sched*, size\_t \* *size* )

gets the total size from the scheduler.

`glt_scheduler_get_total_size()` gets the total size from a scheduler.

## Parameters

in	<i>sched</i>	Handle of the target c\ GLT_sched.
out	<i>size</i>	obtained.

3.8.2.10 void glt\_scheduler\_has\_to\_stop ( GLT\_sched *sched*, GLT\_bool \* *stop* )

Requires to a scheduler to stop.

`glt_scheduler_has_to_stop()` Requires a scheduler to stop.

## Parameters

in	<i>sched</i>	Handle of the target c\ GLT_sched.
out	<i>stop</i>	shows the answer of the scheduler.

3.8.2.11 void glt\_scheduler\_set\_data ( GLT\_sched *sched*, void \* *data* )

Sets new data to a scheduler.

`glt_scheduler_set_data()` Sets data to a scheduler.

**Parameters**

in	<i>sched</i>	Handle of the target c\ GLT_sched.
in	<i>data</i>	to be set.

## 3.9 Key functions

### Functions

- void [glt\\_key\\_create](#) (void(\*destructor)(void \*value), GLT\_key \*newkey)  
*Creates a key.*
- void [glt\\_key\\_free](#) (GLT\_key \*key)  
*Destroys a key.*
- void [glt\\_key\\_set](#) (GLT\_key key, void \*value)  
*Sets new value to a key.*
- void [glt\\_key\\_get](#) (GLT\_key key, void \*\*value)  
*Gets value from a key.*

### 3.9.1 Detailed Description

These functions manage the GLT keys for the ULTs.

### 3.9.2 Function Documentation

#### 3.9.2.1 void [glt\\_key\\_create](#) ( void(\*) (void \*value) *destructor*, GLT\_key \* *newkey* )

Creates a key.

[glt\\_key\\_create\(\)](#) creates a key.

##### Parameters

in	<i>destructor</i>	Handle to newly created GLT_ult.
out	<i>newkey</i>	Handle to newly created GLT_key.

#### 3.9.2.2 void [glt\\_key\\_free](#) ( GLT\_key \* *key* )

Destroys a key.

[glt\\_key\\_free\(\)](#) destroys a key for ULTs.

##### Parameters

in	<i>key</i>	Handle to the target GLT_key.
----	------------	-------------------------------

#### 3.9.2.3 void [glt\\_key\\_get](#) ( GLT\_key *key*, void \*\* *value* )

Gets value from a key.

[glt\\_key\\_get\(\)](#) Gets value from a key.

##### Parameters

in	<i>key</i>	Handle of the target c\ GLT_key.
out	<i>value</i>	obtained value.

#### 3.9.2.4 void [glt\\_key\\_set](#) ( GLT\_key *key*, void \* *value* )

Sets new value to a key.

[glt\\_key\\_set\(\)](#) Sets value to a key.

**Parameters**

<i>in</i>	<i>key</i>	Handle of the target c\ GLT_key.
<i>in</i>	<i>value</i>	to be set.



# Index

- `__attribute__`
    - Library functions, [5](#)
- Barrier functions, [7](#)
  - `glt_barrier_create`, [7](#)
  - `glt_barrier_free`, [7](#)
  - `glt_barrier_wait`, [7](#)
- Condition functions, [8](#)
  - `glt_cond_broadcast`, [8](#)
  - `glt_cond_create`, [8](#)
  - `glt_cond_free`, [8](#)
  - `glt_cond_signal`, [8](#)
  - `glt_cond_wait`, [9](#)
- `glt_barrier_create`
  - Barrier functions, [7](#)
- `glt_barrier_free`
  - Barrier functions, [7](#)
- `glt_barrier_wait`
  - Barrier functions, [7](#)
- `glt_cond_broadcast`
  - Condition functions, [8](#)
- `glt_cond_create`
  - Condition functions, [8](#)
- `glt_cond_free`
  - Condition functions, [8](#)
- `glt_cond_signal`
  - Condition functions, [8](#)
- `glt_cond_wait`
  - Condition functions, [9](#)
- `glt_finalize`
  - Library functions, [5](#)
- `glt_get_num_threads`
  - Util functions, [20](#)
- `glt_get_thread_num`
  - Util functions, [20](#)
- `glt_get_wtime`
  - Timer functions, [18](#)
- `glt_init`
  - Library functions, [5](#)
- `glt_key_create`
  - Key functions, [25](#)
- `glt_key_free`
  - Key functions, [25](#)
- `glt_key_get`
  - Key functions, [25](#)
- `glt_key_set`
  - Key functions, [25](#)
- `glt_mutex_create`
  - Mutex functions, [10](#)
- `glt_mutex_free`
  - Mutex functions, [10](#)
- `glt_mutex_lock`
  - Mutex functions, [10](#)
- `glt_mutex_trylock`
  - Mutex functions, [10](#)
- `glt_mutex_unlock`
  - Mutex functions, [11](#)
- `glt_scheduler_create_basic`
  - Scheduler functions, [21](#)
- `glt_scheduler_config_free`
  - Scheduler functions, [22](#)
- `glt_scheduler_create`
  - Scheduler functions, [22](#)
- `glt_scheduler_exit`
  - Scheduler functions, [22](#)
- `glt_scheduler_finish`
  - Scheduler functions, [22](#)
- `glt_scheduler_free`
  - Scheduler functions, [22](#)
- `glt_scheduler_get_data`
  - Scheduler functions, [23](#)
- `glt_scheduler_get_size`
  - Scheduler functions, [23](#)
- `glt_scheduler_get_total_size`
  - Scheduler functions, [23](#)
- `glt_scheduler_has_to_stop`
  - Scheduler functions, [23](#)
- `glt_scheduler_set_data`
  - Scheduler functions, [23](#)
- `glt_tasklet_cancel`
  - Work-units functions, [12](#)
- `glt_tasklet_create`
  - Work-units functions, [13](#)
- `glt_tasklet_create_to`
  - Work-units functions, [13](#)
- `glt_tasklet_join`
  - Work-units functions, [13](#)
- `glt_tasklet_malloc`
  - Work-units functions, [13](#)
- `glt_tasklet_self`
  - Work-units functions, [13](#)
- `glt_timer_create`
  - Timer functions, [18](#)
- `glt_timer_free`
  - Timer functions, [18](#)
- `glt_timer_get_secs`
  - Timer functions, [19](#)

- glt\_timer\_start
  - Timer functions, [19](#)
- glt\_timer\_stop
  - Timer functions, [19](#)
- glt\_ult\_cancel
  - Work-units functions, [15](#)
- glt\_ult\_create
  - Work-units functions, [15](#)
- glt\_ult\_create\_to
  - Work-units functions, [15](#)
- glt\_ult\_exit
  - Work-units functions, [15](#)
- glt\_ult\_get\_id
  - Work-units functions, [15](#)
- glt\_ult\_join
  - Work-units functions, [16](#)
- glt\_ult\_malloc
  - Work-units functions, [16](#)
- glt\_ult\_migrate\_self\_to
  - Work-units functions, [16](#)
- glt\_ult\_self
  - Work-units functions, [16](#)
- glt\_workunit\_get\_thread\_id
  - Work-units functions, [16](#)
- glt\_yield
  - Work-units functions, [17](#)
- glt\_yield\_to
  - Work-units functions, [17](#)
- Key functions, [25](#)
  - glt\_key\_create, [25](#)
  - glt\_key\_free, [25](#)
  - glt\_key\_get, [25](#)
  - glt\_key\_set, [25](#)
- Library functions, [5](#)
  - \_\_attribute\_\_, [5](#)
  - glt\_finalize, [5](#)
  - glt\_init, [5](#)
- Mutex functions, [10](#)
  - glt\_mutex\_create, [10](#)
  - glt\_mutex\_free, [10](#)
  - glt\_mutex\_lock, [10](#)
  - glt\_mutex\_trylock, [10](#)
  - glt\_mutex\_unlock, [11](#)
- Scheduler functions, [21](#)
  - glt\_schededuler\_create\_basic, [21](#)
  - glt\_scheduler\_config\_free, [22](#)
  - glt\_scheduler\_create, [22](#)
  - glt\_scheduler\_exit, [22](#)
  - glt\_scheduler\_finish, [22](#)
  - glt\_scheduler\_free, [22](#)
  - glt\_scheduler\_get\_data, [23](#)
  - glt\_scheduler\_get\_size, [23](#)
  - glt\_scheduler\_get\_total\_size, [23](#)
  - glt\_scheduler\_has\_to\_stop, [23](#)
  - glt\_scheduler\_set\_data, [23](#)
- Timer functions, [18](#)
  - glt\_get\_wtime, [18](#)
  - glt\_timer\_create, [18](#)
  - glt\_timer\_free, [18](#)
  - glt\_timer\_get\_secs, [19](#)
  - glt\_timer\_start, [19](#)
  - glt\_timer\_stop, [19](#)
- Util functions, [20](#)
  - glt\_get\_num\_threads, [20](#)
  - glt\_get\_thread\_num, [20](#)
- Work-units functions, [12](#)
  - glt\_tasklet\_cancel, [12](#)
  - glt\_tasklet\_create, [13](#)
  - glt\_tasklet\_create\_to, [13](#)
  - glt\_tasklet\_join, [13](#)
  - glt\_tasklet\_malloc, [13](#)
  - glt\_tasklet\_self, [13](#)
  - glt\_ult\_cancel, [15](#)
  - glt\_ult\_create, [15](#)
  - glt\_ult\_create\_to, [15](#)
  - glt\_ult\_exit, [15](#)
  - glt\_ult\_get\_id, [15](#)
  - glt\_ult\_join, [16](#)
  - glt\_ult\_malloc, [16](#)
  - glt\_ult\_migrate\_self\_to, [16](#)
  - glt\_ult\_self, [16](#)
  - glt\_workunit\_get\_thread\_id, [16](#)
  - glt\_yield, [17](#)
  - glt\_yield\_to, [17](#)