X10LIB Reference Manual

1.0

Generated by Doxygen 1.3.9.1

Tue May 20 16:02:00 2008

Contents

Chapter 1

X10LIB Data Structure Index

1.1 X10LIB Data Structures

Here are the data structures with brief descriptions:

Chapter 2

X10LIB File Index

2.1 X10LIB File List

Here is a list of all files with brief descriptions:

Chapter 3

X10LIB Data Structure Documentation

3.1 __x10_async_descr_t Struct Reference

Data Fields

- **__x10_async_type_t _async_type**
- **x10_async_closure_t ∗ closure**
- union {
  __x10_normal_async_descr_t _normal_async_descr
  __x10_global_async_descr_t _global_async_descr
  } **u**

3.1.1 Field Documentation

3.1.1.1 __x10_async_type_t __x10_async_descr_t::_async_type

3.1.1.2 __x10_global_async_descr_t __x10_async_descr_t::_global_async_-descr

3.1.1.3 __x10_normal_async_descr_t __x10_async_descr_t::_normal_-async_descr

3.1.1.4 x10_async_closure_t∗ __x10_async_descr_t::closure

3.1.1.5 union { ... } __x10_async_descr_t::u

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

- **async.cc**

3.2 __x10_global_async_descr_t Struct Reference

Data Fields

- size_t **cl_size**

3.2.1 Field Documentation

3.2.1.1 size_t __x10_global_async_descr_t::cl_size

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

- **async.cc**

3.3 __x10_normal_async_descr_t Struct Reference

Data Fields

- **x10_finish_record_t finish_record**
- **x10_place_t parent**
- size_t **cl_size**

3.3.1 Field Documentation

3.3.1.1 size_t __x10_normal_async_descr_t::cl_size

3.3.1.2 x10_finish_record_t __x10_normal_async_descr_t::finish_record

3.3.1.3 x10_place_t __x10_normal_async_descr_t::parent

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

- **async.cc**

3.4 AsyncQueue Struct Reference

`#include <queue.h>`

Data Fields

- **x10_async_queue_el_t _head**
- **x10_async_queue_el_t _tail**

3.4.1 Field Documentation

3.4.1.1 x10_async_queue_el_t AsyncQueue::_head

3.4.1.2 x10_async_queue_el_t AsyncQueue::_tail

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

- **queue.h**

3.5 AsyncQueueEl Struct Reference

`#include <queue.h>`

Data Fields

- void * **_el**
- **AsyncQueueEl** * **_next**
- **AsyncQueueEl** * **_prev**

3.5.1 Field Documentation

3.5.1.1 void* AsyncQueueEl::_el

3.5.1.2 struct AsyncQueueEl* AsyncQueueEl::_next

3.5.1.3 struct AsyncQueueEl* AsyncQueueEl::_prev

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

- **queue.h**

3.6 finish_compl_message_t Struct Reference

Data Fields

- **tuple** ∗ **tuples**
- int **num_tuples**
- int **finish_id**

3.6.1 Field Documentation

3.6.1.1 int finish_compl_message_t::finish_id

3.6.1.2 int finish_compl_message_t::num_tuples

3.6.1.3 tuple∗ finish_compl_message_t::tuples

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

- **finish.cc**

3.7 finish_message_t Struct Reference

Data Fields

- __upcrt_AMHeaderHandler_t **header**
- int **headerlen**
- int **finish_id**
- int **usize**

3.7.1 Field Documentation

3.7.1.1 int finish_message_t::finish_id

3.7.1.2 __upcrt_AMHeaderHandler_t finish_message_t::header

3.7.1.3 int finish_message_t::headerlen

3.7.1.4 int finish_message_t::usize

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

- **finish.cc**

3.8 tuple Struct Reference

Data Fields

- int **count**
- int **place**

3.8.1 Field Documentation

3.8.1.1 int tuple::count

3.8.1.2 int tuple::place

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

- **finish.cc**

3.9 x10_async_closure_t Struct Reference

`#include <x10_types.h>`

Data Fields

- **x10_async_handler_t handler**

3.9.1 Field Documentation

3.9.1.1 x10_async_handler_t x10_async_closure_t::handler

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

- **x10_types.h**

3.10 x10_clock_t Struct Reference

`#include <x10_types.h>`

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

- **x10_types.h**

3.11 x10_comm_handle_t Struct Reference

```
#include <x10_types.h>
```

Data Fields

- void * **rts_handle**
- void * **header_buf**

3.11.1 Field Documentation

3.11.1.1 void* x10_comm_handle_t::header_buf

3.11.1.2 void* x10_comm_handle_t::rts_handle

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

- **x10_types.h**

3.12 x10_finish_record_t Struct Reference

`#include <x10_types.h>`

Data Fields

- int **finish_id**
- **x10_place_t finish_root**

3.12.1 Field Documentation

3.12.1.1 int x10_finish_record_t::finish_id

3.12.1.2 x10_place_t x10_finish_record_t::finish_root

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

- **x10_types.h**

3.13 x10_proxy_t Struct Reference

`#include <x10_types.h>`

Data Fields

- **x10_place_t loc**
- void * **addr**

3.13.1 Field Documentation

3.13.1.1 void* x10_proxy_t::addr

3.13.1.2 x10_place_t x10_proxy_t::loc

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

- **x10_types.h**

Chapter 4

X10LIB File Documentation

4.1 async.cc File Reference

```
#include <assert.h>
#include "rts_messaging.h"
#include "queue.h"
#include "x10.h"
```

Data Structures

- struct **__x10_normal_async_descr_t**
- struct **__x10_global_async_descr_t**
- struct **__x10_async_descr_t**

Enumerations

- enum **__x10_async_type_t** { **NORMAL_ASYNC**, **GLOBAL_ASYNC**, **CLOCKED_NORMAL_ASYNC**, **CLOCKED_GLOBAL_ASYNC** }

Functions

- EXTERN void **__x10_callback_asyncswitch** (**x10_async_closure_t** ∗closure, **x10_finish_record_t** ∗frecord, **x10_clock_t** ∗clocks, int num_clocks)
- void **__x10_finish_bookeeping_outgoing** (const **x10_finish_record_t** ∗finish_-record, **x10_place_t** tgt)
- void **__x10_finish_bookeeping_incoming** (**x10_finish_record_t** ∗finish_record)
- void **__x10_async_dispatch** (**__x10_async_descr_t** ∗)
- void **__x10_flush** ()
- void **__x10_async_queue_add** (void ∗async_descr)

  AM handlers (internal).

- __xlupc_local_addr_t **__x10_normal_async_handler** (const __upcrt_-AMHeader_t ∗header, __upcrt_AMComplHandler_t ∗∗comp_h, void ∗∗arg)

- __xlupc_local_addr_t **__x10_global_async_handler** (const __upcrt_-AMHeader_t *header, __upcrt_AMComplHandler_t **comp_h, void **arg)
- void **__x10_async_init** ()
- **x10_comm_handle_t x10_async_spawn** (const **x10_place_t** tgt, const **x10_-async_closure_t** *closure, const size_t cl_size, const **x10_finish_record_t** *frecord, const **x10_clock_t** *clocks, const int num_clocks)

  asyncs spawn an async on given target (NON-BLOCKING). x10lib assumes SPMD programming model; code is replicated everywhere

- **x10_err_t x10_async_spawn_wait** (**x10_comm_handle_t** req)

  wait for the async_spawn to complete locally (BLOCKING)

- **x10_err_t x10_probe** ()

  check for any asyncs in the internal async queue and execute them. This method should be used on the receiver side to make progress (NON-BLOCKING)

Variables

- **x10_place_t __x10_here**
- unsigned int **__x10_numplaces**
- **x10_finish_record_t __x10_global_frecord** = {0, 0}
- **x10_async_queue_t __x10_async_queue**

4.1.1 Enumeration Type Documentation

4.1.1.1 enum __x10_async_type_t

Enumeration values:

NORMAL_ASYNC

GLOBAL_ASYNC

CLOCKED_NORMAL_ASYNC

CLOCKED_GLOBAL_ASYNC

4.1.2 Function Documentation

4.1.2.1 void __x10_async_dispatch (__x10_async_descr_t *)

4.1.2.2 void __x10_async_init ()

4.1.2.3 void __x10_async_queue_add (void * *async_descr*) `[static]`

AM handlers (internal).

4.1.2.4 EXTERN void __x10_callback_asyncswitch (x10_async_closure_t * *closure*, x10_finish_record_t * *frecord*, x10_clock_t * *clocks*, int *num_clocks*)

4.1.2.5 void __x10_finish_bookeeping_incoming (x10_finish_record_t * *finish_record*)

4.1.2.6 void __x10_finish_bookeeping_outgoing (const x10_finish_record_t * *finish_record*, x10_place_t *tgt*)

4.1.2.7 void __x10_flush ()

4.1.2.8 __xlupc_local_addr_t __x10_global_async_handler (const __upcrt_AMHeader_t * *header*, __upcrt_AMComplHandler_t ** *comp_h*, void ** *arg*) `[static]`

4.1.2.9 __xlupc_local_addr_t __x10_normal_async_handler (const __upcrt_AMHeader_t * *header*, __upcrt_AMComplHandler_t ** *comp_h*, void ** *arg*) `[static]`

4.1.2.10 x10_comm_handle_t x10_async_spawn (const x10_place_t *tgt*, const x10_async_closure_t * *closure*, const size_t *cl_size*, const x10_finish_record_t * *frecord*, const x10_clock_t * *clocks*, const int *num_clocks*)

asyncs spawn an async on given target (NON-BLOCKING). x10lib assumes SPMD programming model; code is replicated everywhere

Parameters:

tgt target place

closure pointer to async closure (see **x10_types.h**(p. 35))

cl_size size of the async closure

frecord pointer to the finish record (see **x10_types.h**(p. 35))

clocks clock set for the async (see **x10_types.h**(p. 35))

num_clocks number of clocks in the clock set

Returns:

handle to wait for

4.1.2.11 x10_err_t x10_async_spawn_wait (x10_comm_handle_t *handle*)

wait for the async_spawn to complete locally (BLOCKING)

Parameters:

handle handle returned by x10_async_spawn (see **x10_types.h**(p. 35))

Returns:

returns an error or success

4.1.2.12 x10_err_t x10_probe ()

check for any asyncs in the internal async queue and execute them. This method should be used on the receiver side to make progress (NON-BLOCKING)

4.1.3 Variable Documentation

4.1.3.1 x10_async_queue_t __x10_async_queue

4.1.3.2 x10_finish_record_t __x10_global_frecord = {0, 0} `[static]`

4.1.3.3 x10_place_t __x10_here

4.1.3.4 unsigned int __x10_numplaces

4.2 finish.cc File Reference

```
#include <assert.h>
#include "rts_messaging.h"
#include "x10.h"
#include "x10_types.h"
```

Data Structures

- struct **finish_message_t**
- struct **tuple**
- struct **finish_compl_message_t**

Defines

- #define **X10_MAX_FINISH_ID** 100
- #define **X10_MAX_PLACES** 1024

Functions

- **tuple** * **construct_tuples** (int *size, int finish_id)
- void **__x10_finish_init** ()
- void **__x10_finish_compl_handler** (void *arg)
- __xlupc_local_addr_t **__x10_finish_handler** (const __upcrt_AMHeader_t *header, __upcrt_AMComplHandler_t **comp_h, void **arg)
- int **__x10_is_place_quiescent** (const **x10_finish_record_t** *frecord)
- void **__x10_propagate_credits** (const **x10_finish_record_t** *frecord)
- **x10_err_t x10_finish_child** (const **x10_finish_record_t** *frecord, void *ex_buf, int ex_buf_size)

  notify the "root" that I have finished (called by children activity only)

- **x10_err_t x10_finish_begin** (**x10_finish_record_t** *frecord, void *multi_ex_buf, int *ex_offsets, int max_ex_buf_size, int max_num_exceptions)

  finish start the finish_scope (called by root activity only)

- **x10_err_t x10_finish_begin_global** (**x10_finish_record_t** *frecord, void *multi_ex_buf, int *ex_offsets, int max_ex_buf_size, int max_num_exceptions)
- **x10_err_t x10_finish_end** (const **x10_finish_record_t** *frecord, int *num_-exceptions)

  end the finish_scope (called by root activity only). Waits for global termination of all the activities (BLOCKING)

- void **__x10_finish_bookeeping_outgoing** (const **x10_finish_record_t** *frecord, **x10_place_t** place)
- void **__x10_finish_bookeeping_incoming** (**x10_finish_record_t** *frecord)

Variables

- **x10_place_t __x10_here**
- unsigned int **__x10_numplaces**
- int **__x10_finish_counter** = 1
- int **__x10_async_counts** [X10_MAX_FINISH_ID][X10_MAX_PLACES]
- int **__x10_async_spawned** [X10_MAX_FINISH_ID]

4.2.1 Define Documentation

4.2.1.1 #define X10_MAX_FINISH_ID 100

4.2.1.2 #define X10_MAX_PLACES 1024

4.2.2 Function Documentation

4.2.2.1 void __x10_finish_bookeeping_incoming (x10_finish_record_t * *frecord*)

4.2.2.2 void __x10_finish_bookeeping_outgoing (const x10_finish_record_t * *frecord*, x10_place_t *place*)

4.2.2.3 void __x10_finish_compl_handler (void * *arg*) `[static]`

4.2.2.4 __xlupc_local_addr_t __x10_finish_handler (const __upcrt_AMHeader_t * *header*, __upcrt_AMComplHandler_t ** *comp_h*, void ** *arg*) `[static]`

4.2.2.5 void __x10_finish_init ()

4.2.2.6 int __x10_is_place_quiescent (const x10_finish_record_t * *frecord*)

4.2.2.7 void __x10_propagate_credits (const x10_finish_record_t * *frecord*)

4.2.2.8 tuple * construct_tuples (int * *size*, int *finish_id*) `[static]`

4.2.2.9 x10_err_t x10_finish_begin (x10_finish_record_t * *frecord*, void * *mult_ex_buf*, int * *ex_offsets*, int *max_ex_buf_size*, int *max_num_exceptions*)

finish start the finish_scope (called by root activity only)

Parameters:

frecord the finish record

multi_ex_buf buffer for the resulting multi_exceptions (if any)

ex_offsets offsets array for individual exceptions

max_ex_buf_size maximum size of the multi_ex_buf

max_num_exceptions maximum number of individual exceptions

4.2.2.10 x10_err_t x10_finish_begin_global (x10_finish_record_t * *frecord*, void * *multi_ex_buf*, int * *ex_offsets*, int *max_ex_buf_size*, int *max_num_exceptions*)

4.2.2.11 x10_err_t x10_finish_child (const x10_finish_record_t * *frecord*, void * *ex_buf*, int *ex_buf_size*)

notify the "root" that I have finished (called by children activity only)

Parameters:

frecord finish record

ex_buf exception buffer

ex_buf_size size of the exception buffer

4.2.2.12 x10_err_t x10_finish_end (const x10_finish_record_t * *finish_record*, int * *num_exceptions*)

end the finish_scope (called by root activity only). Waits for global termination of all the activities (BLOCKING)

Parameters:

finish_record pointer to finish_record

num_exceptions total number of exceptions

4.2.3 Variable Documentation

4.2.3.1 int __x10_async_counts[X10_MAX_FINISH_ID][X10_MAX_-PLACES]

4.2.3.2 int __x10_async_spawned[X10_MAX_FINISH_ID]

4.2.3.3 int __x10_finish_counter = 1 `[static]`

4.2.3.4 x10_place_t __x10_here

4.2.3.5 unsigned int __x10_numplaces

4.3 init.cc File Reference

```
#include <assert.h>
#include "rts_messaging.h"
#include "x10.h"
```

Functions

- void **__x10_finish_init** ()
- void **__x10_async_init** ()
- __xlupc_local_addr_t **__x10_termination_handler** (const __upcrt_AMHeader_t *header, __upcrt_AMComplHandler_t **comp_h, void **arg)
- **x10_err_t x10_init** ()

  init/finalize

- **x10_err_t x10_finalize** ()
- **x10_err_t x10_infinite_poll** ()

Variables

- **x10_place_t __x10_here**
- unsigned int **__x10_numplaces**
- int **__x10_terminate_program** = 0

4.3.1 Function Documentation

4.3.1.1 void __x10_async_init ()

4.3.1.2 void __x10_finish_init ()

4.3.1.3 __xlupc_local_addr_t __x10_termination_handler (const __upcrt_AMHeader_t * *header*, __upcrt_AMComplHandler_t ** *comp_h*, void ** *arg*) [static]

4.3.1.4 x10_err_t x10_finalize ()

4.3.1.5 x10_err_t x10_infinite_poll ()

Performs x10_probe infinitely, until a termination message is received (BLOCKING)

4.3.1.6 x10_err_t x10_init ()

init/finalize

4.3.2 Variable Documentation

4.3.2.1 x10_place_t __x10_here

4.3.2.2 unsigned int __x10_numplaces

4.3.2.3 int __x10_terminate_program = 0

4.4 queue.cc File Reference

`#include <assert.h>`

`#include <stdio.h>`

`#include <stdlib.h>`

`#include "queue.h"`

Functions

- **x10_async_queue_t CreateQueue** ()
- void **DeleteQueue** (**x10_async_queue_t** q)
- void **PushQueue** (**x10_async_queue_t** q, void ∗element)
- **x10_async_queue_el_t PopQueue** (**x10_async_queue_t** q)
- void **RemoveQueue** (**x10_async_queue_t** q, **x10_async_queue_el_t** el)

4.4.1 Function Documentation

4.4.1.1 x10_async_queue_t CreateQueue ()

Implementation file for X10Lib's **AsyncQueue**(p. 8) interface. ∗

4.4.1.2 void DeleteQueue (x10_async_queue_t *q*)

4.4.1.3 x10_async_queue_el_t PopQueue (x10_async_queue_t *q*)

4.4.1.4 void PushQueue (x10_async_queue_t *q*, void ∗ *element*)

4.4.1.5 void RemoveQueue (x10_async_queue_t *q*, x10_async_queue_el_t *el*)

4.5 queue.h File Reference

Data Structures

- struct **AsyncQueueEl**
- struct **AsyncQueue**

Typedefs

- typedef **AsyncQueueEl** ∗ **x10_async_queue_el_t**
- typedef **AsyncQueue** ∗ **x10_async_queue_t**

Functions

- **x10_async_queue_t CreateQueue** ()
- void **DeleteQueue** (**x10_async_queue_t**)
- void **PushQueue** (**x10_async_queue_t**, void ∗)
- **x10_async_queue_el_t PopQueue** (**x10_async_queue_t**)
- void **RemoveQueue** (**x10_async_queue_t**, **x10_async_queue_el_t**)

4.5.1 Typedef Documentation

4.5.1.1 typedef struct AsyncQueueEl∗ x10_async_queue_el_t

4.5.1.2 typedef struct AsyncQueue∗ x10_async_queue_t

4.5.2 Function Documentation

4.5.2.1 x10_async_queue_t CreateQueue ()

Implementation file for X10Lib's **AsyncQueue**(p. 8) interface. ∗

4.5.2.2 void DeleteQueue (x10_async_queue_t)

4.5.2.3 x10_async_queue_el_t PopQueue (x10_async_queue_t)

4.5.2.4 void PushQueue (x10_async_queue_t, void ∗)

4.5.2.5 void RemoveQueue (x10_async_queue_t, x10_async_queue_el_t)

4.6 refs.cc File Reference

`#include <stdlib.h>`

`#include "x10.h"`

Functions

- bool **x10_is_localref** (void *ref)
  check if the reference is local
- **x10_remote_ref_t x10_serialize_ref** (void *ref)
  remote reference serialize a reference (local or remote)
- void * **x10_deserialize_ref** (**x10_remote_ref_t** ref)
  deserialize a remote reference
- int **x10_get_loc** (void *ref)

4.6.1 Function Documentation

4.6.1.1 void* x10_deserialize_ref (x10_remote_ref_t *ref*)

deserialize a remote reference

4.6.1.2 int x10_get_loc (void * *ref*)

\ brief get the location of a reference

4.6.1.3 bool x10_is_localref (void * *ref*)

check if the reference is local

4.6.1.4 x10_remote_ref_t x10_serialize_ref (void * *ref*)

remote reference serialize a reference (local or remote)

4.7 x10.h File Reference

```
#include <stdio.h>
#include "x10_types.h"
```

Functions

- EXTERN **x10_err_t x10_init** ()

  init/finalize

- EXTERN **x10_err_t x10_finalize** ()
- EXTERN **x10_comm_handle_t x10_async_spawn** (const **x10_place_t** tgt, const **x10_async_closure_t** ∗closure, const size_t cl_size, const **x10_finish_record_t** ∗frecord, const **x10_clock_t** ∗clocks, const int num_clocks)

  asyncs spawn an async on given target (NON-BLOCKING). x10lib assumes SPMD programming model; code is replicated everywhere

- EXTERN **x10_err_t x10_async_spawn_wait** (**x10_comm_handle_t** handle)

  wait for the async_spawn to complete locally (BLOCKING)

- EXTERN **x10_err_t x10_probe** ()

  check for any asyncs in the internal async queue and execute them. This method should be used on the receiver side to make progress (NON-BLOCKING)

- EXTERN **x10_err_t x10_infinite_poll** ()
- EXTERN **x10_err_t x10_finish_begin** (**x10_finish_record_t** ∗frecord, void ∗mult_ex_buf, int ∗ex_offsets, int max_ex_buf_size, int max_num_exceptions)

  finish start the finish_scope (called by root activity only)

- EXTERN **x10_err_t x10_finish_begin_global** (**x10_finish_record_t** ∗frecord, void ∗mult_ex_buf, int ∗ex_offsets, int max_ex_buf_size, int max_num_exceptions)
- EXTERN **x10_err_t x10_finish_end** (const **x10_finish_record_t** ∗finish_record, int ∗num_exceptions)

  end the finish_scope (called by root activity only). Waits for global termination of all the activities (BLOCKING)

- EXTERN **x10_err_t x10_finish_child** (const **x10_finish_record_t** ∗frecord, void ∗ex_buf, int ex_buf_size)

  notify the "root" that I have finished (called by children activity only)

- EXTERN **x10_err_t x10_clock_init** (**x10_clock_t** ∗c)

  clocks initialize a clock c (see **x10_types.h**(p. 35) *for* **x10_clock_t**(p. 14)*)*

- EXTERN **x10_err_t x10_clock_free** (**x10_clock_t** ∗c)
- EXTERN **x10_err_t x10_clock_resume** (**x10_clock_t** ∗c)

  perform a resume operation on clock c

- EXTERN **x10_err_t x10_clock_drop** (**x10_clock_t** ∗c)

  drop a clock c

- EXTERN **x10_err_t x10_next** (**x10_clock_t** *c)

  perform a next operation

- EXTERN **x10_err_t x10_next_all** ()
- EXTERN **x10_remote_ref_t x10_serialize_ref** (void *ref)

  remote reference serialize a reference (local or remote)

- EXTERN void * **x10_deserialize_ref** (**x10_remote_ref_t** ref)

  deserialize a remote reference

- EXTERN int **x10_get_loc** (void *ref)
- EXTERN bool **x10_is_localref** (void *ref)

  check if the reference is local

Variables

- **x10_place_t __x10_here**
- unsigned int **__x10_numplaces**

4.7.1 Function Documentation

4.7.1.1 EXTERN x10_comm_handle_t x10_async_spawn (const x10_place_t *tgt*, const x10_async_closure_t * *closure*, const size_t *cl_size*, const x10_finish_record_t * *frecord*, const x10_clock_t * *clocks*, const int *num_clocks*)

asyncs spawn an async on given target (NON-BLOCKING). x10lib assumes SPMD programming model; code is replicated everywhere

Parameters:

tgt target place

closure pointer to async closure (see **x10_types.h**(p. 35))

cl_size size of the async closure

frecord pointer to the finish record (see **x10_types.h**(p. 35))

clocks clock set for the async (see **x10_types.h**(p. 35))

num_clocks number of clocks in the clock set

Returns:

handle to wait for

4.7.1.2 EXTERN x10_err_t x10_async_spawn_wait (x10_comm_handle_t *handle*)

wait for the async_spawn to complete locally (BLOCKING)

Parameters:

handle handle returned by x10_async_spawn (see **x10_types.h**(p. 35))

Returns:

returns an error or success

4.7.1.3 EXTERN x10_err_t x10_clock_drop (x10_clock_t * *c*)

drop a clock c

4.7.1.4 EXTERN x10_err_t x10_clock_free (x10_clock_t * *c*)

4.7.1.5 EXTERN x10_err_t x10_clock_init (x10_clock_t * *c*)

clocks initialize a clock c (see **x10_types.h**(p. 35) for **x10_clock_t**(p. 14))

4.7.1.6 EXTERN x10_err_t x10_clock_resume (x10_clock_t * *c*)

perform a resume operation on clock c

4.7.1.7 EXTERN void* x10_deserialize_ref (x10_remote_ref_t *ref*)

deserialize a remote reference

4.7.1.8 EXTERN x10_err_t x10_finalize ()

4.7.1.9 EXTERN x10_err_t x10_finish_begin (x10_finish_record_t * *frecord*, void * *mult_ex_buf*, int * *ex_offsets*, int *max_ex_buf_size*, int *max_num_exceptions*)

finish start the finish_scope (called by root activity only)

Parameters:

frecord the finish record

multi_ex_buf buffer for the resulting multi_exceptions (if any)

ex_offsets offsets array for individual exceptions

max_ex_buf_size maximum size of the multi_ex_buf

max_num_exceptions maximum number of individual exceptions

4.7.1.10 EXTERN x10_err_t x10_finish_begin_global (x10_finish_record_t * *frecord*, void * *mult_ex_buf*, int * *ex_offsets*, int *max_ex_buf_size*, int *max_num_exceptions*)

4.7.1.11 EXTERN x10_err_t x10_finish_child (const x10_finish_record_t * *frecord*, void * *ex_buf*, int *ex_buf_size*)

notify the "root" that I have finished (called by children activity only)

Parameters:

frecord finish record

ex_buf exception buffer

ex_buf_size size of the exception buffer

4.7.1.12 EXTERN x10_err_t x10_finish_end (const x10_finish_record_t * *finish_record*, int * *num_exceptions*)

end the finish_scope (called by root activity only). Waits for global termination of all the activities (BLOCKING)

Parameters:

finish_record pointer to finish_record

num_exceptions total number of exceptions

4.7.1.13 EXTERN int x10_get_loc (void * *ref*)

\ brief get the location of a reference

4.7.1.14 EXTERN x10_err_t x10_infinite_poll ()

Performs x10_probe infinitely, until a termination message is received (BLOCKING)

4.7.1.15 EXTERN x10_err_t x10_init ()

init/finalize

4.7.1.16 EXTERN bool x10_is_localref (void * *ref*)

check if the reference is local

4.7.1.17 EXTERN x10_err_t x10_next (x10_clock_t * *c*)

perform a next operation

4.7.1.18 EXTERN x10_err_t x10_next_all ()

4.7.1.19 EXTERN x10_err_t x10_probe ()

check for any asyncs in the internal async queue and execute them. This method should be used on the receiver side to make progress (NON-BLOCKING)

4.7.1.20 EXTERN x10_remote_ref_t x10_serialize_ref (void * *ref*)

remote reference serialize a reference (local or remote)

4.7.2 Variable Documentation

4.7.2.1 x10_place_t __x10_here

4.7.2.2 unsigned int __x10_numplaces

4.8 x10_types.h File Reference

`#include <stdio.h>`

Data Structures

- struct **x10_finish_record_t**
- struct **x10_async_closure_t**
- struct **x10_comm_handle_t**
- struct **x10_clock_t**
- struct **x10_proxy_t**

Defines

- #define **EXTERN**
- #define **bool** char

Typedefs

- typedef unsigned **x10_place_t**
- typedef unsigned **x10_async_handler_t**
- typedef **x10_proxy_t x10_remote_ref_t**
- typedef unsigned **x10_condition_t**

Enumerations

- enum **x10_err_t** { **X10_OK**, **X10_NOT_OK** }

4.8.1 Define Documentation

4.8.1.1 #define bool char

4.8.1.2 #define EXTERN

4.8.2 Typedef Documentation

4.8.2.1 typedef unsigned x10_async_handler_t

4.8.2.2 typedef unsigned x10_condition_t

4.8.2.3 typedef unsigned x10_place_t

4.8.2.4 typedef x10_proxy_t x10_remote_ref_t

4.8.3 Enumeration Type Documentation

4.8.3.1 enum x10_err_t

Enumeration values:

X10_OK

X10_NOT_OK

Index