TI4_SE2

Generated by Doxygen 1.7.6.1

Wed Apr 27 2016 09:41:16

Contents

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Class Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Namespace Documentation

4.1 lib::SingletonConcurrency Namespace Reference

Contains threading models of the Singleton template.

Classes

- struct SingleThreaded
- struct MultiThreaded

4.1.1 Detailed Description

Contains threading models of the Singleton template.

Chapter 5

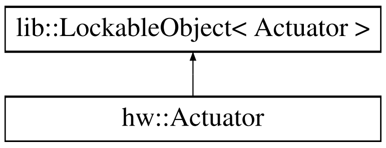
Class Documentation

5.1 hw::Actuator Class Reference

Singular access point to actuators.

```
#include <Actuator.h>
```

Inheritance diagram for hw::Actuator:



Public Types

- typedef lib::LockableObject < Actuator > **Super**
- typedef lib::Singleton < Actuator, lib::SingletonConcurrency::MultiThreaded > **SingletonInst**
- typedef Super::Lock **Lock**
- typedef lib::qnx::Channel **Channel**
- typedef lib::Thread **Thread**

Public Member Functions

- const Channel & **getChannel** () const

Static Public Attributes

- static const uint8_t **LED_ACTIVATE** = 0x00

- static const uint8_t **MOTOR_BELT** = 0x01
- static const uint8_t **MOTOR_SWITCH** = 0x02
- static const int **CCMD** = 3

5.1.1 Detailed Description

Singular access point to actuators.

The Actuator class encapsulates access to all actuators of the attached hw unit, including LEDs, the conveyor belt and the electromagnetic switch. The dispatcher runs in its own thread and communicates via lib::qnx::Channel. It is a singleton via lib::Singleton template.

5.2 lib::And< T1, T2 > Struct Template Reference

Static Public Attributes

- static const bool **value** = T1::value && T2::value

template<typename T1, typename T2> struct lib::And< T1, T2 >

5.3 lib::ListAnd< List >::AndFn< T1, T2 > Struct Template - Reference

Public Types

- typedef And< T1, T2 > **Type**

template<typename List>template<typename T1, typename T2> struct lib::ListAnd< List >::- AndFn< T1, T2 >

5.4 lib::Apply< F, T > Struct Template Reference

Public Types

- typedef Cons< DO(F< DO(Car< T >)>), DO(Apply< F, DO(Cdr< T >)>)> **Type**

template<template< typename > class F, typename T> struct lib::Apply< F, T >

5.5 lib::Apply< F, Nil > Struct Template Reference

Public Types

- typedef Nil **Type**

template<template< typename > class F> struct lib::Apply< F, Nil >

5.6 lib::Array< T, N > Class Template Reference

Public Types

- typedef T **value_type**
- typedef std::size_t **size_type**
- typedef std::ptrdiff_t **difference_type**
- typedef value_type & **reference**
- typedef const value_type & **const_reference**
- typedef value_type ∗ **pointer**
- typedef const value_type ∗ **const_pointer**
- typedef pointer **iterator**
- typedef const_pointer **const_iterator**

Public Member Functions

- reference **at** (size_type i)
- const_reference **at** (size_type i) const
- reference **operator[ ]** (size_type i)
- const_reference **operator[ ]** (size_type i) const
- reference **front** ()
- const_reference **front** () const
- reference **back** ()
- const_reference **back** () const
- pointer **data** ()
- const_pointer **data** () const
- iterator **begin** ()
- const_iterator **cbegin** () const
- iterator **end** ()
- const_iterator **cend** ()
- bool **empty** () const
- size_type **size** () const
- size_type **max_size** () const

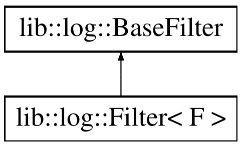
template<typename T, std::size_t N> class lib::Array< T, N >

5.7 lib::Array< T, 0 > Class Template Reference

template<typename T> class lib::Array< T, 0 >

5.8 lib::log::BaseFilter Struct Reference

Inheritance diagram for lib::log::BaseFilter:

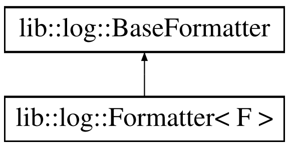


Public Member Functions

- virtual bool **accept** (const LogRecord &)=0

5.9 lib::log::BaseFormatter Struct Reference

Inheritance diagram for lib::log::BaseFormatter:

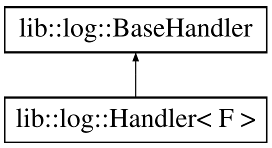


Public Member Functions

- virtual std::string **format** (const LogRecord &)=0

5.10 lib::log::BaseHandler Struct Reference

Inheritance diagram for lib::log::BaseHandler:

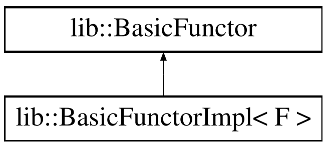


Public Member Functions

- **BaseHandler** (Formatter_ptr f)
- void **handle** (const LogRecord &lr)

5.11 lib::BasicFunctor Struct Reference

Inheritance diagram for lib::BasicFunctor:



Public Member Functions

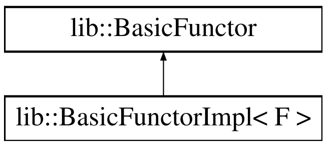
- virtual void **operator()** ()=0

5.12 lib::BasicFunctorImpl< F > Class Template Reference

Basic functor encapsulating anything callable that takes no arguments.

```
#include <FtorWrapper.hpp>
```

Inheritance diagram for lib::BasicFunctorImpl< F >:



Public Member Functions

- **BasicFunctorImpl** (const F &f)
- void **operator()** ()

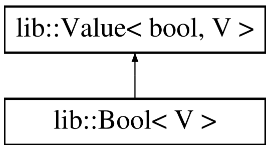
5.12.1 Detailed Description

template<typename F>class lib::BasicFunctorImpl< F >

Basic functor encapsulating anything callable that takes no arguments.

5.13 lib::Bool< V > Struct Template Reference

Inheritance diagram for lib::Bool< V >:



template<bool V> struct lib::Bool< V >

5.14 lib::Caar< T > Struct Template Reference

Public Member Functions

- typedef **DO** (Car< DO(Car< T >)>) Type

template<typename T> struct lib::Caar< T >

5.15 lib::Cadr< T > Struct Template Reference

Public Member Functions

- typedef **DO** (Car< DO(Cdr< T >)>) Type

template<typename T> struct lib::Cadr< T >

5.16 lib::Car< T > Struct Template Reference

Public Types

- typedef T::Head **Type**

template<typename T> struct lib::Car< T >

5.17 lib::Cdar< T > Struct Template Reference

Public Member Functions

- typedef **DO** (Cdr< DO(Car< T >)>) Type

template<typename T> struct lib::Cdar< T >

5.18 lib::Cddr< T > Struct Template Reference

Public Member Functions

- typedef **DO** (Cdr< DO(Cdr< T >)>) Type

template<typename T> struct lib::Cddr< T >

5.19 lib::Cdr< T > Struct Template Reference

Public Types

- typedef T::Tail **Type**

template<typename T> struct lib::Cdr< T >

5.20 lib::qnx::Channel Class Reference

Public Member Functions

- Receiver **open** (int=0)
- Connection **connect** (int=0) const
- bool **isOpen** () const
- void **close** ()

Friends

- class **Receiver**

5.21 lib::TryCall_apply< T, E, D >::Check< typename, > Struct Template Reference

template<typename T, typename E, typename D>template<typename, void(∗)(const E &, D)> struct lib::TryCall_apply< T, E, D >::Check< typename, >

5.22 lib::TryCall_apply< T, E, void >::Check< typename, > Struct Template Reference

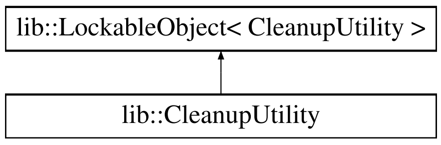
template<typename T, typename E>template<typename, void(∗)(const E &)> struct lib::TryCall_apply< T, E, void >::Check< typename, >

5.23 lib::CleanupUtility Class Reference

Utility for controlling the lifetime of static objects, i.e.

`#include <CleanupUtility.h>`

Inheritance diagram for lib::CleanupUtility:



Classes

- struct **Compare**

Public Member Functions

- void **scheduleAtExit** (atexit_fn f)
- void **scheduleAtExitWithPriority** (atexit_fn, size_t)

Static Public Member Functions

- static CleanupUtility & **instance** ()

Static Public Attributes

- static const size_t **DEFAULT_PRIORITY** = 10

5.23.1 Detailed Description

Utility for controlling the lifetime of static objects, i.e.

\ Singletons. This utility class offers a more fine-grained, priority based version of clib's lifo based ::atexit(void (∗)(void)) function. Functors are executed highest (numerically smallest) priority first.

5.24 lib::CreateTransitionDependencyList< List >::CollectDependencies< E > Struct Template Reference

Classes

- struct IsCorrectEvent

Public Member Functions

- typedef **DO** (Apply< Cadr, DO(Filter< IsCorrectEvent, RawDependencies >)>) Dependencies
- typedef **MAKELIST** (E, Dependencies) Type

template<typename List>template<typename E> struct lib::CreateTransitionDependencyList< List >::CollectDependencies< E >

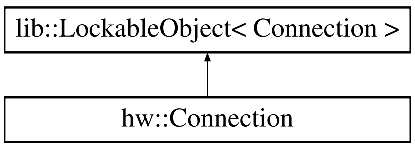
5.25 lib::Condition Class Reference

Public Member Functions

- void **wait** ()
- bool **wait** (timespec ∗)
- void **broadcast** ()
- void **lock** ()
- void **unlock** ()

5.26 hw::Connection Class Reference

Inheritance diagram for hw::Connection:



Classes

- class Impl

Public Member Functions

- **Connection** (const std::string &d, bool a)
- void **close** ()
- bool **connected** () const
- bool **running** () const
- bool **doneWriting** () const
- void **sendData** (lib::Data_ptr)
- lib::Data_ptr **receiveData** ()
- bool **hasData** () const

5.27 lib::qnx::Connection Class Reference

Public Member Functions

- void **send** (Data_ptr) const

Friends

- class **Channel**

5.28 lib::Cons< H, T > Struct Template Reference

Public Types

- typedef H **Head**
- typedef T **Tail**

template<typename H, typename T> struct lib::Cons< H, T >

5.29 lib::ConsFn< T1, T2 > Struct Template Reference

Public Types

- typedef Cons< T1, T2 > **Type**

template<typename T1, typename T2> struct lib::ConsFn< T1, T2 >

5.30 lib::ConstructFSMLineage< T > Struct Template Reference

template<typename T> struct lib::ConstructFSMLineage< T >

5.31 lib::ConstructFSMLineage< Cons< T, Nil > > Struct - Template Reference

template<typename T> struct lib::ConstructFSMLineage< Cons< T, Nil > >

5.32 lib::Contains< List, T > Struct Template Reference

Static Public Attributes

- static const bool **value** = IsSame<DO(Car<List>), T>::value || Contains<DO(-Cdr<List>), T>::value

template<typename List, typename T> struct lib::Contains< List, T >

5.33 lib::Contains< Nil, T > Struct Template Reference

Static Public Attributes

- static const bool **value** = false

template<typename T> struct lib::Contains< Nil, T >

5.34 lib::CreateStateList< List > Struct Template Reference

Classes

- struct GetStateFromTransition

Public Member Functions

- typedef **DO** (Flatten< DO(Apply< GetStateFromTransition, List >)>) StateList
- typedef **DO** (ListToMap< DO(Setify< StateList >)>) StateMap
- typedef **DO** (Apply< ReverseCons, StateMap >) Type

template<typename List> struct lib::CreateStateList< List >

5.35 lib::CreateTransitionDependencyList< List > Struct - Template Reference

Classes

- struct CollectDependencies
- struct GetDependency

Public Member Functions

- typedef **DO** (Apply< GetDependency, List >) RawDependencies
- typedef **DO** (Setify< DO(Apply< Car, RawDependencies >)>) EventList
- typedef **DO** (Apply< CollectDependencies, EventList >) Type

template<typename List> struct lib::CreateTransitionDependencyList< List >

5.36 lib::CreateTransitionMap< List > Struct Template Reference

Classes

- struct Transform

Public Member Functions

- typedef **DO** (Apply< Transform, List >) Type

template<typename List> struct lib::CreateTransitionMap< List >

5.37 lib::FSMMaker< I, D, T >::CreateTransitionTree< TT > - Struct Template Reference

Public Types

- typedef TransImpl< DO(Car< TT >), Data, DO(Cadr< TT >), StateList, - TransitionMap > **Type**

template< typename I, typename D, typename T>template< typename TT> struct lib::FSM-Maker< I, D, T >::CreateTransitionTree< TT >

5.38 lib::Data Class Reference

Public Types

- typedef lib::SmartPtr< Data > **Data_ptr**

Public Member Functions

- void ∗ **data** ()
- const void ∗ **data** () const
- size_t **size** () const

Static Public Member Functions

- static Data_ptr **get** (const void ∗, size_t)
- static Data_ptr **move** (void ∗d, size_t s)
- template< typename T >
  static Data_ptr **get** (const T &t)
- static Data_ptr **empty** (size_t s)

5.39 lib::Decay< T > Struct Template Reference

Public Types

- typedef T **Type**

template< typename T> struct lib::Decay< T >

5.40 lib::Decay< const T > Struct Template Reference

Public Types

- typedef Decay< T >::Type **Type**

template< typename T> struct lib::Decay< const T >

5.41 lib::Decay< const volatile T > Struct Template Reference

Public Types

- typedef Decay< T >::Type **Type**

template<typename T> struct lib::Decay< const volatile T >

5.42 lib::Decay< T & > Struct Template Reference

Public Types

- typedef Decay< T >::Type **Type**

template<typename T> struct lib::Decay< T & >

5.43 lib::Decay< volatile T > Struct Template Reference

Public Types

- typedef Decay< T >::Type **Type**

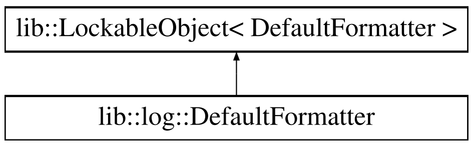
template<typename T> struct lib::Decay< volatile T >

5.44 lib::log::DefaultFormatter Class Reference

Default formatter that lists all information of the passed LogRecord.

`#include <DefaultFormat.h>`

Inheritance diagram for lib::log::DefaultFormatter:


lib::LockableObject< DefaultFormatter >
lib::log::DefaultFormatter

Public Member Functions

- std::string **operator()** (const LogRecord &)

Static Public Member Functions

- static std::string **toDate** (uint64_t)

5.44.1 Detailed Description

Default formatter that lists all information of the passed LogRecord.

It generates string as follows: `"<b>thread-ID</b> [<b>LogLevel</b>] @<b>filename</b>:<b>line</b> '<b>message</b>'"`

5.45 hw::Motor::Direction Struct Reference

Static Public Attributes

- static const pid_t **NONE** = 0x00
- static const pid_t **RIGHT** = 0x01
- static const pid_t **LEFT** = 0x02

5.46 lib::RingBufferConcurrency::MultiThreaded< T >::EmptyLock Class Reference

Public Member Functions

- **EmptyLock** (MultiThreaded< T > ∗t)

template<typename T> class lib::RingBufferConcurrency::MultiThreaded< T >::EmptyLock

5.47 lib::RingBufferConcurrency::SingleThreaded< T >::EmptyLock Struct Reference

Public Member Functions

- **EmptyLock** (SingleThreaded< T > ∗)

template<typename T> struct lib::RingBufferConcurrency::SingleThreaded< T >::EmptyLock

5.48 lib::RingBufferConcurrency::MultiThreaded< T >::FillLock - Class Reference

Public Member Functions

- **FillLock** (MultiThreaded< T > ∗t)

template<typename T> class lib::RingBufferConcurrency::MultiThreaded< T >::FillLock

5.49 lib::RingBufferConcurrency::SingleThreaded< T >::FillLock Struct Reference

Public Member Functions

- **FillLock** (SingleThreaded< T > ∗)

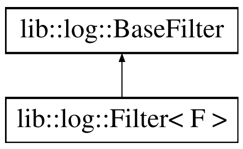
template<typename T> struct lib::RingBufferConcurrency::SingleThreaded< T >::FillLock

5.50 lib::log::Filter< F > Struct Template Reference

Filter template that accepts functors.

`#include <Filter.hpp>`

Inheritance diagram for lib::log::Filter< F >:



Public Member Functions

- **Filter** (F f)
- bool **accept** (const LogRecord &lr)

5.50.1 Detailed Description

template<typename F>struct lib::log::Filter< F >

Filter template that accepts functors.

Any logged LogRecord is passed through all filters of the give Logger instance. If any reject it, it will be discarded.

5.51 lib::Filter< F, List > Struct Template Reference

Public Types

- typedef If< F< DO(Car< List >)>::value, Identity< Cons< DO(Car < List >), Rest > >, Identity < Rest > >::Type **Type**

Public Member Functions

- typedef **DO** (Filter< F, DO(Cdr< List >)>) Rest

template<template< typename > class F, typename List> struct lib::Filter< F, List >

5.52 lib::Filter< F, Nil > Struct Template Reference

Public Types

- typedef Nil **Type**

template<template< typename > class F> struct lib::Filter< F, Nil >

5.53 lib::Flatten< T > Struct Template Reference

Public Types

- typedef T **Type**

template<typename T> struct lib::Flatten< T >

5.54 lib::Flatten< Cons< H, T > > Struct Template Reference

Public Types

- typedef If< IsList< H >::value, Join< DO(Flatten< H >), Rest > , Identity< Cons< H, Rest > > >::Type **Type**

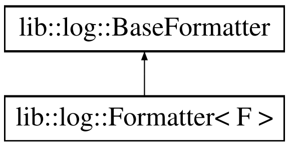
Public Member Functions

- typedef **DO** (Flatten< T >) Rest

template<typename H, typename T> struct lib::Flatten< Cons< H, T > >

5.55 lib::log::Formatter< F > Struct Template Reference

Inheritance diagram for lib::log::Formatter< F >:



Public Member Functions

- **Formatter** (F f)
- std::string **format** (const LogRecord &lr)

template<typename F> struct lib::log::Formatter< F >

5.56 lib::Frequency Class Reference

Convenience class that allows calculation of a signal's period length through its frequency.

```
#include <TimeP.h>
```

Static Public Member Functions

- static Time **Hz** (double v)
- static Time **kHz** (double v)
- static Time **MHz** (double v)

5.56.1 Detailed Description

Convenience class that allows calculation of a signal's period length through its frequency.

5.57 lib::FSM< ID, I, D, Lineage > Struct Template Reference

Public Types

- typedef TryCall_enter< I, D > **EnterFunction**

Public Member Functions

- **FSM** (D d)

template<int ID, typename I, typename D, typename Lineage> struct lib::FSM< ID, I, D, Lineage >

5.58 lib::FSM< ID, I, void, Lineage > Struct Template Reference

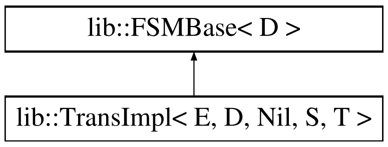
Public Types

- typedef TryCall_enter< I, void > **EnterFunction**

template<int ID, typename I, typename Lineage> struct lib::FSM< ID, I, void, Lineage >

5.59 lib::FSMBase< D > Struct Template Reference

Inheritance diagram for lib::FSMBase< D >:



Public Member Functions

- int **get_state** ()
- D **get_data** ()
- void **set_state** (int state)
- void **set_data** (D d)

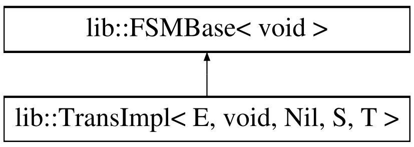
Public Attributes

- int **state_**
- D **data_**

template<typename D> struct lib::FSMBase< D >

5.60 lib::FSMBase< void > Struct Template Reference

Inheritance diagram for lib::FSMBase< void >:



Public Member Functions

- int **get_state** ()
- void **set_state** (int state)

Public Attributes

- int **state_**

template$<>$ struct lib::FSMBase$<$ void $>$

5.61 lib::FSMMaker$<$ I, D, T $>$ Struct Template Reference

Classes

- struct CreateTransitionTree

Public Types

- typedef I **InitialState**
- typedef D **Data**
- typedef ConstructFSMLineage $<$ DO(Apply $<$ CreateTransitionTree, Transitions $>$)$>$ **Lineage**
- typedef FSM$<$ InitialID, InitialState, Data, Lineage $>$ **Type**

Public Member Functions

- typedef **DO** (CreateStateList$<$ T $>$) StateList
- typedef **DO** (CreateTransitionDependencyList$<$ T $>$) Transitions
- typedef **DO** (CreateTransitionMap$<$ T $>$) TransitionMap

Static Public Attributes

- static const int **InitialID** = ValueIdentity$<$DO(GetValue$<$StateList, InitialState$>$)$>$::value

template<typename I, typename D, typename T> struct lib::FSMMaker< I, D, T >

5.62 lib::FtorWrapper< T > Class Template Reference

A functor that calls an object's member function.

`#include <FtorWrapper.hpp>`

Public Member Functions

- **FtorWrapper** (T ∗t, void(T::∗f)(void))
- void **operator()** ()

5.62.1 Detailed Description

template<typename T>class lib::FtorWrapper< T >

A functor that calls an object's member function.

5.63 lib::CreateTransitionDependencyList< List >::GetDependency< T > Struct Template Reference

Public Types

- typedef Cons< typename T::Origin, typename T::Destination > **Tmp**

Public Member Functions

- typedef **MAKELIST** (typename T::Event, Tmp) Type

template<typename List>template<typename T> struct lib::CreateTransitionDependencyList< List >::GetDependency< T >

5.64 lib::GetElem< IDX, List > Struct Template Reference

Public Member Functions

- typedef **DO** (GetElem< IDX-1, DO(Cdr< List >)>) Type

template< int IDX, typename List > struct lib::GetElem< IDX, List >

5.65 lib::GetElem< 0, List > Struct Template Reference

Public Member Functions

- typedef **DO** (Car< List >) Type

template< typename List > struct lib::GetElem< 0, List >

5.66 lib::CreateStateList< List >::GetStateFromTransition< T > Struct Template Reference

Public Member Functions

- typedef **MAKELIST** (typename T::Origin, typename T::Destination) Type

template< typename List >template< typename T > struct lib::CreateStateList< List >::GetState-FromTransition< T >

5.67 lib::GetValue< Map, Key > Struct Template Reference

Public Types

- typedef If< IsSame< DO(Caar < Map >), Key >::value, Identity< DO(Cdar< Map >)>, GetValue< DO(Cdr< Map >), Key > >::Type **Type**

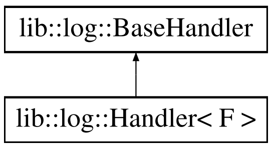
template< typename Map, typename Key > struct lib::GetValue< Map, Key >

5.68 lib::log::Handler< F > Class Template Reference

Handler template that holds a functor.

`#include <Handler.hpp>`

Inheritance diagram for lib::log::Handler< F >:



Public Member Functions

- **Handler** (F f, Formatter_ptr p)

5.68.1 Detailed Description

template<typename F>class lib::log::Handler< F >

Handler template that holds a functor.

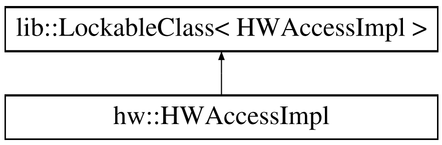
All accepted LogRecords of a Logger instance are passed to the Logger's handlers. There they are run through a formatter; and the formatters output is passed to the functor.

5.69 hw::HWAccessImpl Class Reference

Interface for direct hardware access.

```
#include <HWAccess.h>
```

Inheritance diagram for hw::HWAccessImpl:



Classes

- struct Lock

Public Types

- typedef lib::LockableClass < HWAccessImpl > **Super**
- typedef lib::Singleton < HWAccessImpl, lib::SingletonConcurrency::MultiThreaded > **SingletonInst**
- typedef uint16_t **port_t**
- typedef uint8_t **pin_t**

Public Member Functions

- uint8_t **in** (port_t) const
- void **out** (port_t, pin_t) const
- void **setBits** (port_t, pin_t) const
- void **resetBits** (port_t, pin_t) const

Static Public Attributes

- static const port_t **PORT_A** = 0x300
- static const port_t **PORT_B** = 0x301
- static const port_t **PORT_C** = 0x302

5.69.1 Detailed Description

Interface for direct hardware access.

The HWAccess singleton offers read/write operations to the three ports of the hw unit.

5.70 lib::Identity< T > Struct Template Reference

Public Types

- typedef T **Type**

template<typename T> struct lib::Identity< T >

5.71 lib::If< false, T1, T2 > Struct Template Reference

Public Types

- typedef T2::Type **Type**

template<typename T1, typename T2> struct lib::If< false, T1, T2 >

5.72 lib::If< true, T1, T2 > Struct Template Reference

Public Types

- typedef T1::Type **Type**

template<typename T1, typename T2> struct lib::If< true, T1, T2 >

5.73 hw::Connection::Impl Class Reference

Classes

- struct **DoneRunning**
- struct **Packet**

Friends

- class **Connection**

5.74 lib::InheritLineage< T > Struct Template Reference

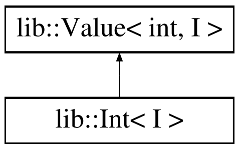
template<typename T> struct lib::InheritLineage< T >

5.75 lib::InheritLineage< Nil > Struct Template Reference

template<> struct lib::InheritLineage< Nil >

5.76 lib::Int< I > Struct Template Reference

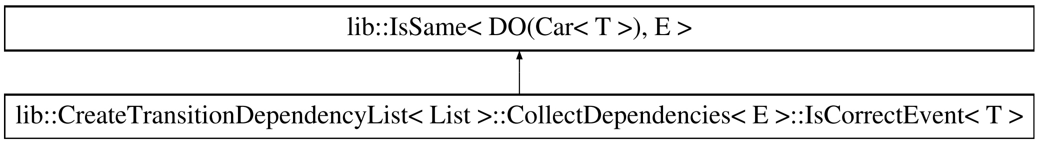
Inheritance diagram for lib::Int< I >:



template<int I> struct lib::Int< I >

5.77 lib::CreateTransitionDependencyList< List >::CollectDependencies< E >::IsCorrectEvent< T > Struct Template Reference

Inheritance diagram for lib::CreateTransitionDependencyList< List >::Collect-Dependencies< E >::IsCorrectEvent< T >:


lib::IsSame< DO(Car< T >), E >
lib::CreateTransitionDependencyList< List >::CollectDependencies< E >::IsCorrectEvent< T >

template<typename List>template<typename E>template<typename T> struct lib::Create-TransitionDependencyList< List >::CollectDependencies< E >::IsCorrectEvent< T >

5.78 lib::IsList< T > Struct Template Reference

Static Public Attributes

- static const bool **value** = false

template<typename T> struct lib::IsList< T >

5.79 lib::IsList< Cons< T1, T2 > > Struct Template Reference

Static Public Attributes

- static const bool **value** = true

template<typename T1, typename T2> struct lib::IsList< Cons< T1, T2 > >

5.80 lib::IsSame< T1, T2 > Struct Template Reference

Static Public Attributes

- static const bool **value** = false

template<typename T1, typename T2> struct lib::IsSame< T1, T2 >

5.81 lib::IsSame< T, T > Struct Template Reference

Static Public Attributes

- static const bool **value** = true

template<typename T> struct lib::IsSame< T, T >

5.82 lib::IsSuperType< Sub, Super > Struct Template Reference

Classes

- struct No
- struct Yes

Static Public Member Functions

- template<typename T >
  static Yes **f** (T ∗)
- template<typename T >
  static No **f** (...)

Static Public Attributes

- static const bool **value** = sizeof(f<Super>(static_cast<Sub ∗>(NULL))) == sizeof(Yes)

template<typename Sub, typename Super> struct lib::IsSuperType< Sub, Super >

5.83 lib::Join< List, Appendage > Struct Template Reference

Public Types

- typedef Cons< DO(Car< List >), DO(Join< DO(Cdr< List >), Appendage >)> **Type**

template<typename List, typename Appendage> struct lib::Join< List, Appendage >

5.84 lib::Join< Nil, Appendage > Struct Template Reference

Public Types

- typedef Appendage **Type**

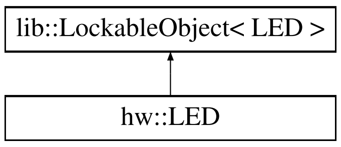
template<typename Appendage> struct lib::Join< Nil, Appendage >

5.85 hw::LED Class Reference

Allows access to LEDs.

```
#include <LED.h>
```

Inheritance diagram for hw::LED:



Public Types

- typedef lib::LockableObject< LED > **Super**
- typedef lib::Singleton< LED, lib::SingletonConcurrency::MultiThreaded > **-SingletonInst**
- typedef Super::Lock **Lock**
- typedef uint32_t **led_t**

Public Member Functions

- void **turnOn** (led_t led)
- void **turnOff** (led_t led)
- void **activate** (led_t, bool)
- void **blink** (led_t, const lib::Time &)

Static Public Attributes

- static const led_t **GREEN** = MXT_PINPORT(HWAccessImpl::PORT_A, 0x20)
- static const led_t **YELLOW** = MXT_PINPORT(HWAccessImpl::PORT_A, 0x40)
- static const led_t **RED** = MXT_PINPORT(HWAccessImpl::PORT_A, 0x80)
- static const led_t **START** = MXT_PINPORT(HWAccessImpl::PORT_C, 0x01)
- static const led_t **RESET** = MXT_PINPORT(HWAccessImpl::PORT_C, 0x02)
- static const led_t **Q1** = MXT_PINPORT(HWAccessImpl::PORT_C, 0x04)
- static const led_t **Q2** = MXT_PINPORT(HWAccessImpl::PORT_C, 0x08)
- static const int **CLED** = 7

Friends

- class **Actuator**

5.85.1 Detailed Description

Allows access to LEDs.

Offers an interface for accessing LEDs on the hw unit. Implements blinking functionality via lib::Timer.

5.86 lib::ListAnd< List > Struct Template Reference

Classes

- struct AndFn

Public Member Functions

- typedef **DO** (Merge< AndFn, True, List >) Type

Static Public Attributes

- static const bool **value** = Type::value

template<typename List> struct lib::ListAnd< List >

5.87 lib::ListOr< List > Struct Template Reference

Classes

- struct OrFn

Public Member Functions

- typedef **DO** (Merge< OrFn, False, List >) Type

Static Public Attributes

- static const bool **value** = Type::value

template<typename List> struct lib::ListOr< List >

5.88 lib::ListToMap< List > Struct Template Reference

Public Member Functions

- typedef **DO** (ListToMapImpl< 0, List >) Type

template<typename List> struct lib::ListToMap< List >

5.89 lib::ListToMapImpl< IDX, List > Struct Template Reference

Public Types

- typedef Cons< Cons< Int< IDX > , DO(Car< List >)>, > **Type** template< int IDX, typename List > struct lib::ListToMapImpl< IDX, List >

5.90 lib::ListToMapImpl< IDX, Nil > Struct Template Reference

Public Types

- typedef Nil **Type**

template< int IDX > struct lib::ListToMapImpl< IDX, Nil >

5.91 lib::SingletonConcurrency::SingleThreaded< T >::Lock - Struct Reference

Public Member Functions

- **Lock** (Mutex ∗)

template< typename T > struct lib::SingletonConcurrency::SingleThreaded< T >::Lock

5.92 lib::SingletonConcurrency::MultiThreaded< T >::Lock Struct Reference

Public Member Functions

- **Lock** (Mutex ∗mtx)

Public Attributes

- Mutex ∗ **mtx_**

template< typename T > struct lib::SingletonConcurrency::MultiThreaded< T >::Lock

5.93 hw::HWAccessImpl::Lock Struct Reference

5.94 lib::Lock< T, E, R > Class Template Reference

Public Types

- typedef T **Mutex**

Public Member Functions

- **Lock** (Mutex &mtx)
- **Lock** (Mutex ∗mtx)

template<typename T, void(T::∗)(void) E = &T::lock, void(T::∗)(void) R = &T::unlock> class lib::-Lock< T, E, R >

5.95 lib::LockableClass< T, M >::Lock Struct Reference

Public Member Functions

- **Lock** (T ∗)

template<typename T, typename M = Mutex> struct lib::LockableClass< T, M >::Lock

5.96 lib::LockableObject< T, M >::Lock Struct Reference

Public Member Functions

- **Lock** (T ∗t)

template<typename T, typename M = Mutex> struct lib::LockableObject< T, M >::Lock

5.97 lib::LockableClass< T, M > Class Template Reference

Classes

- struct Lock

Public Types

- typedef M **Mutex**

template<typename T, typename M = Mutex> class lib::LockableClass< T, M >

5.98 lib::LockableObject< T, M > Class Template Reference

Classes

- struct Lock

Public Types

- typedef M **Mutex**

Friends

- class **Lock**

template$<$typename T, typename M = Mutex$>$ class lib::LockableObject$<$ T, M $>$

5.99 lib::test::TestManager::Log Struct Reference

Public Member Functions

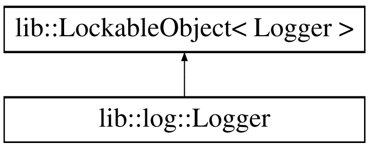
- virtual std::string **read** ()=0
- virtual bool **isEmpty** () const =0

5.100 lib::log::Logger Class Reference

Logger class.

`#include <Logger.h>`

Inheritance diagram for lib::log::Logger:



Public Types

- typedef SmartPtr$<$ Logger $>$ **Logger_ptr**

Public Member Functions

- void **addParent** (Logger_ptr)
- void **removeParent** (Logger_ptr)
- void **addHandler** (Handler_ptr)
- void **removeHandler** (Handler_ptr)
- void **addFilter** (Filter_ptr)
- void **removeFilter** (Filter_ptr)
- void **log** (const LogLevel &, const std::string &, const char $*$, int,...)
- void **log** (const LogRecord &)

Friends

- class **LogManagerImpl**
- class **SmartPtr**< **Logger** >

5.100.1 Detailed Description

Logger class.

Compiles a LogLevel, the file name & line of caller and a custom message into a -LogRecord. This LogRecord is run through all added filters; if any filter rejects it the LogRecord is discarded. Otherwise it is passed to all added handlers and send to all added parent logs

Cannot be instantiated directly; the LogManager utility grants access to Logger instances.

5.101 lib::log::LogLevel Class Reference

Public Member Functions

- int **level** () const
- const char ∗ **label** () const
- bool **operator==** (const LogLevel &ll) const
- bool **operator!=** (const LogLevel &ll) const
- bool **operator**< (const LogLevel &ll) const
- bool **operator**> (const LogLevel &ll) const
- bool **operator**<**=** (const LogLevel &ll) const
- bool **operator**>**=** (const LogLevel &ll) const

Static Public Attributes

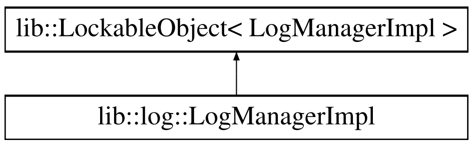
- static const LogLevel **INFO**
- static const LogLevel **WARNING**
- static const LogLevel **ERROR**
- static const LogLevel **CRITICAL**

5.102 lib::log::LogManagerImpl Class Reference

LogManager Singleton, grants access to Logger instances.

`#include <LogManager.h>`

Inheritance diagram for lib::log::LogManagerImpl:



Public Member Functions

- Logger_ptr **rootLog** ()
- Logger_ptr **getLog** (const std::string &)

5.102.1 Detailed Description

LogManager Singleton, grants access to Logger instances.

Creates and exposes Logger instances by alphanumerical id.

Offers a "root log" for convenience.

5.103 lib::log::LogRecord Class Reference

Public Member Functions

- **LogRecord** (LogLevel ll, uint64_t ts, uint16_t tid, const std::string &msg, const char ∗f=NULL, int l=-1)
- const LogLevel & **logLevel** () const
- uint64_t **timestamp** () const
- uint16_t **threadID** () const
- const std::string & **message** () const
- const char ∗ **file** () const
- int **line** () const
- bool **hasFile** () const

5.104 lib::MakeList_0 Struct Reference

Public Types

- typedef Nil **Type**

5.105 lib::MakeList_1< A > Struct Template Reference

Public Types

- typedef Cons< A, MAKELIST_0 > **Type**

template<typename A> struct lib::MakeList_1< A >

5.106 lib::MakeList_2< A, B > Struct Template Reference

Public Types

- typedef Cons< A, MAKELIST_1(B)> **Type**

template<typename A, typename B> struct lib::MakeList_2< A, B >

5.107 lib::MakeList_3< A, B, C > Struct Template Reference

Public Types

- typedef Cons< A, MAKELIST_2(B, C)> **Type**

template<typename A, typename B, typename C> struct lib::MakeList_3< A, B, C >

5.108 lib::MakeList_4< A, B, C, D > Struct Template Reference

Public Types

- typedef Cons< A, MAKELIST_3(B, C, D)> **Type**

template<typename A, typename B, typename C, typename D> struct lib::MakeList_4< A, B, C, D >

5.109 lib::MakeList_5< A, B, C, D, E > Struct Template Reference

Public Types

- typedef Cons< A, MAKELIST_4(B, C, D, E)> **Type**

template<typename A, typename B, typename C, typename D, typename E> struct lib::MakeList-_5< A, B, C, D, E >

5.110 lib::MakeList_6< A, B, C, D, E, F > Struct Template - Reference

Public Types

- typedef Cons< A, MAKELIST_5(B, C, D, E, F)> **Type**

template<typename A, typename B, typename C, typename D, typename E, typename F> struct lib::MakeList_6< A, B, C, D, E, F >

5.111 lib::MakeList_7< A, B, C, D, E, F, G > Struct Template - Reference

Public Types

- typedef Cons< A, MAKELIST_6(B, C, D, E, F, G)> **Type**

template<typename A, typename B, typename C, typename D, typename E, typename F, typename G> struct lib::MakeList_7< A, B, C, D, E, F, G >

5.112 lib::MakeList_8< A, B, C, D, E, F, G, H > Struct Template - Reference

Public Types

- typedef Cons< A, MAKELIST_7(B, C, D, E, F, G, H)> **Type**

template<typename A, typename B, typename C, typename D, typename E, typename F, typename G, typename H> struct lib::MakeList_8< A, B, C, D, E, F, G, H >

5.113 lib::MakeList_9< A, B, C, D, E, F, G, H, I > Struct Template Reference

Public Types

- typedef Cons< A, MAKELIST_8(B, C, D, E, F, G, H, I)> **Type**

template<typename A, typename B, typename C, typename D, typename E, typename F, typename G, typename H, typename I> struct lib::MakeList_9< A, B, C, D, E, F, G, H, I >

5.114 lib::Merge< F, T, List > Struct Template Reference

Public Member Functions

- typedef **DO** (Merge< F, DO(F< T, DO(Car< List >)>), DO(Cdr< List >)>) Type

template<template< typename, typename > class F, typename T, typename List> struct lib::Merge< F, T, List >

5.115 lib::Merge< F, T, Nil > Struct Template Reference

Public Types

- typedef T **Type**

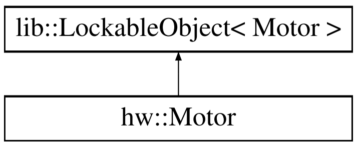
template<template< typename, typename > class F, typename T> struct lib::Merge< F, T, Nil >

5.116 hw::Motor Class Reference

Client interface for controlling the conveyor belt and electromagnetic switch.

`#include <Motor.h>`

Inheritance diagram for hw::Motor:



Classes

- struct Direction
- struct Speed
- struct State

Public Types

- typedef lib::LockableObject < Motor > **Super**
- typedef lib::Singleton< Motor, lib::SingletonConcurrency::MultiThreaded > **SingletonInst**
- typedef Super::Lock **Lock**
- typedef uint8_t **pid_t**

Public Member Functions

- void controlBelt (pid_t dir, pid_t speed)

  Controls conveyor belt.
- void controlSwitch (pid_t state)

  Controls electromagnetic switch.

Static Public Attributes

- static const pid_t **SWITCH** = 0x10

Friends

- class **Actuator**

5.116.1 Detailed Description

Client interface for controlling the conveyor belt and electromagnetic switch.

5.116.2 Member Function Documentation

5.116.2.1 void hw::Motor::controlBelt (pid_t *dir,* pid_t *speed*)

Controls conveyor belt.

Parameters

dir	`Direction` the conveyor belt is supposed to move in.
speed	`Speed` of the conveyor belt.

If `dir == Direction::NONE` *or* `speed == Speed::STOP` the conveyor belt is turned of, but never suppressed.

5.116.2.2 void hw::Motor::controlSwitch (pid_t *state*)

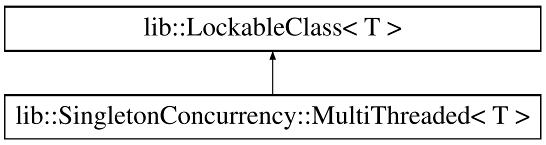
Controls electromagnetic switch.

Parameters

state	`State` the switch is supposed to be in.

5.117 lib::SingletonConcurrency::MultiThreaded< T > Struct - Template Reference

Inheritance diagram for lib::SingletonConcurrency::MultiThreaded< T >:



Classes

- struct Lock

template<typename T> struct lib::SingletonConcurrency::MultiThreaded< T >

5.118 lib::RingBufferConcurrency::MultiThreaded< T > Class - Template Reference

Classes

- class EmptyLock
- class FillLock

Public Member Functions

- std::size_t **size** () const

template<typename T> class lib::RingBufferConcurrency::MultiThreaded< T >

5.119 lib::Mutex Class Reference

Public Member Functions

- void **lock** ()
- void **unlock** ()
- pthread_mutex_t & **raw** ()

5.120 lib::Nil Struct Reference

Public Types

- typedef Nil **Type**

5.121 lib::IsSuperType< Sub, Super >::No Struct Reference

Public Attributes

- char **v** [2]

template<typename Sub, typename Super> struct lib::IsSuperType< Sub, Super >::No

5.122 lib::Not< T > Struct Template Reference

Static Public Attributes

- static const bool **value** = !T::value

template<typename T> struct lib::Not< T >

5.123 lib::Or< T1, T2 > Struct Template Reference

Static Public Attributes

- static const bool **value** = T1::value || T2::value

template<typename T1, typename T2> struct lib::Or< T1, T2 >

5.124 lib::ListOr< List >::OrFn< T1, T2 > Struct Template - Reference

Public Types

- typedef Or< T1, T2 > **Type**

template<typename List>template<typename T1, typename T2> struct lib::ListOr< List >::OrFn< T1, T2 >

5.125 lib::qnx::Receiver Class Reference

Public Member Functions

- Data_ptr **receive** () const

Friends

- class **Channel**

5.126 lib::test::TestManager::Registrar Struct Reference

Public Member Functions

- **Registrar** (const std::string &test_id, testFn test)

5.127 lib::Reverse< List > Struct Template Reference

Public Member Functions

- typedef **DO** (ReverseImpl< Nil, List >) Type

template<typename List> struct lib::Reverse< List >

5.128 lib::ReverseCons< Cell > Struct Template Reference

Public Types

- typedef Cons< DO(Cdr< Cell >), DO(Car< Cell >)> **Type**

template<typename Cell> struct lib::ReverseCons< Cell >

5.129 lib::ReverseImpl< Done, ToDo > Struct Template Reference

Public Types

- typedef ReverseImpl< Cons< DO(Car < ToDo >), Done >, >::Type **Type**

template< typename Done, typename ToDo > struct lib::ReverseImpl< Done, ToDo >

5.130 lib::ReverseImpl< Done, Nil > Struct Template Reference

Public Types

- typedef Done **Type**

template< typename Done > struct lib::ReverseImpl< Done, Nil >

5.131 lib::RingBuffer< T, N, ThreadingPolicy > Class Template - Reference

Public Member Functions

- T & **front** ()
- const T & **front** () const
- void **enqueue** (const T &)
- T **dequeue** ()
- bool **empty** () const
- size_t **max_size** () const

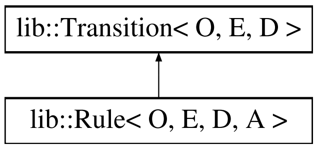
Static Public Attributes

- static const std::size_t **capacity** = N

template< typename T, std::size_t N, template< class > class ThreadingPolicy = RingBuffer-Concurrency::SingleThreaded > class lib::RingBuffer< T, N, ThreadingPolicy >

5.132 lib::Rule< O, E, D, A > Struct Template Reference

Inheritance diagram for lib::Rule< O, E, D, A >:



template< typename O, typename E, typename D, typename A = Nil > struct lib::Rule< O, E, D, A >

5.133 lib::test::TestManager::Selector Struct Reference

Public Member Functions

- **Selector** (const std::string &unit_id)

5.134 lib::Semaphore Class Reference

Public Member Functions

- **Semaphore** (unsigned=0)
- void **up** ()
- void **down** ()
- unsigned **get** () const

5.135 lib::Setify< List > Struct Template Reference

Public Member Functions

- typedef **DO** (SetifyImpl< Nil, List >) Type

template<typename List> struct lib::Setify< List >

5.136 lib::SetifyImpl< Done, ToDo > Struct Template Reference

Public Types

- typedef SetifyImpl< typename If< Contains< Done, DO(Car < ToDo >)>-::value, Identity < Done >, Identity< Cons< DO(Car < ToDo >), Done > > >::Type, > ::Type **Type**

template<typename Done, typename ToDo> struct lib::SetifyImpl< Done, ToDo >

5.137 lib::SetifyImpl< Done, Nil > Struct Template Reference

Public Member Functions

- typedef **DO** (Reverse< Done >) Type

template< typename Done > struct lib::SetifyImpl< Done, Nil >

5.138 lib::SingletonConcurrency::SingleThreaded< T > Struct Template Reference

Classes

- struct Lock

template< typename T > struct lib::SingletonConcurrency::SingleThreaded< T >

5.139 lib::RingBufferConcurrency::SingleThreaded< T > Class Template Reference

Classes

- struct EmptyLock
- struct FillLock

Public Member Functions

- std::size_t **size** () const

template< typename T > class lib::RingBufferConcurrency::SingleThreaded< T >

5.140 lib::Singleton< T, TM, P > Class Template Reference

Template for convenient Singleton creation.

```
#include <Singleton.hpp>
```

Static Public Member Functions

- static T & instance ()

  Access singleton class implementation.

5.140.1 Detailed Description

template< typename T, template< typename > class TM = SingletonConcurrency::SingleThreaded, size_t P = CleanupUtility::DEFAULT_PRIORITY > class lib::Singleton< T, TM, P >

Template for convenient Singleton creation.

Parameters are:

- **T**: Singleton class
- **TM**: Threading model that will be applied to the singletons creation
- **P**: Priority of the singletons lifetime. This template uses the lib::CleanupUtility to manage its life time.

5.140.2 Member Function Documentation

5.140.2.1 template<typename T , template< typename > class TM, size_t P> T & lib::Singleton< T, TM, P >::instance (void) [static]

Access singleton class implementation.

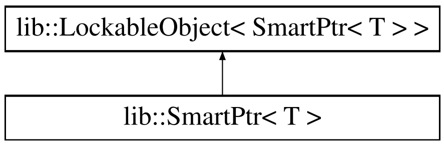
Uses the *double checked locking* pattern for creation synchronization.

5.141 lib::SmartPtr< T > Class Template Reference

Smart pointer class for automatic life time management.

`#include <SmartPtr.hpp>`

Inheritance diagram for lib::SmartPtr< T >:



Public Member Functions

- **SmartPtr** (T *p)
- **SmartPtr** (const SmartPtr< T > &p)
- SmartPtr< T > & **operator=** (const SmartPtr< T > &p)
- void **reset** ()
- void **set** (T *p)
- T * **operator->** ()
- const T * **operator->** () const
- T & **operator*** ()
- const T & **operator*** () const
- template<typename TT >
  TT **to** ()
- **operator bool** () const
- bool **operator==** (const SmartPtr< T > &p) const
- bool **operator!=** (const SmartPtr< T > &p) const

5.141.1 Detailed Description

template<typename T>class lib::SmartPtr< T >

Smart pointer class for automatic life time management.

Supports full object semantics and automatically cleans up when the last SmartPtr instance pointing to its held object is destroyed.

5.142 hw::Motor::Speed Struct Reference

Static Public Attributes

- static const pid_t **FAST** = 0x00
- static const pid_t **SLOW** = 0x04
- static const pid_t **STOP** = 0x08

5.143 hw::Motor::State Struct Reference

Static Public Attributes

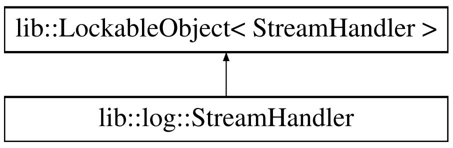
- static const pid_t **OPEN** = 0
- static const pid_t **CLOSE** = 1

5.144 lib::log::StreamHandler Class Reference

Handler compatible functor that writes its LogRecord to an std::stream instance.

`#include <StreamHandler.h>`

Inheritance diagram for lib::log::StreamHandler:



Public Member Functions

- **StreamHandler** (std::ostream ∗os)
- void **operator()** (const std::string &)

5.144.1 Detailed Description

Handler compatible functor that writes its LogRecord to an std::stream instance.

Used in conjuntion with std::cout to write LogRecords to standard output.

5.145 lib::test::TestManager Class Reference

Classes

- struct Log
- struct Registrar
- struct Selector

Public Types

- typedef void(∗ **testFn**)(void)

Public Member Functions

- void **setUnit** (const std::string &)
- void **addTest** (const std::string &, testFn)
- int **run** ()
- Log & **getLog** ()

Static Public Member Functions

- static TestManager & **Instance** ()

5.146 lib::Thread Class Reference

Encapsulates the most important features of a thread.

```
#include <Thread.h>
```

Public Member Functions

- Thread ()

  Default constructor.
- template<typename F >
  Thread (F)

  Constructor taking functor to execute in new thread.
- Thread (Thread &)

Copy constructor; Moves content to this Thread object.

- ∼Thread ()

  Destructor.

- Thread & operator= (Thread &)

  Assignment operator.

- void join ()

  Calls join on the Thread.

- bool joinable () const

  Wether or not the Thread is joinable.

Protected Member Functions

- void run ()

  This is called from the new Thread.

Static Protected Member Functions

- static void $*$ **entryPoint** (void $*$)

5.146.1 Detailed Description

Encapsulates the most important features of a thread.

5.146.2 Constructor & Destructor Documentation

5.146.2.1 lib::Thread::Thread (void)

Default constructor.

Initializes inert Thread

5.146.2.2 template$<$typename F $>$ lib::Thread::Thread (F *f*)

Constructor taking functor to execute in new thread.

Warning

throws std::runtime_error if thread cannot be started.

5.146.2.3 lib::Thread::∼Thread (void)

Destructor.

Warning

terminates if this Thread is still joinable

5.146.3 Member Function Documentation

5.146.3.1 void lib::Thread::join (void)

Calls join on the Thread.

Warning

must be called from the same context as ctor.
throws std::runtime_error if this Thread is not joinable

5.146.3.2 bool lib::Thread::joinable ( ) const `[inline]`

Wether or not the Thread is joinable.

Warning

Thread cannot be destroyed while joinable

5.146.3.3 Thread & lib::Thread::operator= (Thread & *t*)

Assignment operator.

Moves content to this Thread object.

Warning

terminates if this Thread is already joinable

5.146.3.4 void lib::Thread::run (void) `[protected]`

This is called from the new Thread.

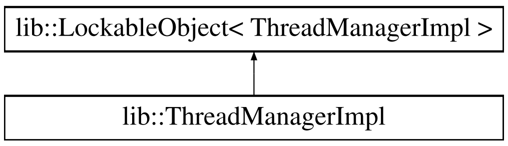
It executes the user's functor.

Warning

terminates if functor throws an exception

5.147 lib::ThreadManagerImpl Class Reference

Inheritance diagram for lib::ThreadManagerImpl:



Public Types

- typedef uint16_t **tid_t**

Public Member Functions

- tid_t **addThread** (pthread_t)
- void **removeThread** (pthread_t)
- tid_t **getThread** (pthread_t)
- tid_t **getCurrent** ()

5.148 lib::Time Class Reference

Data class representing a timeframe with microsecond accuracy.

`#include <TimeP.h>`

Public Types

- typedef uint32_t **us_t**

Public Member Functions

- **Time** (us_t t)
- void **wait** () const
- void **toTimespec** (timespec ∗)
- us_t **raw** () const

Static Public Member Functions

- static Time **h** (us_t v)
- static Time **min** (us_t v)
- static Time **s** (us_t v)

- static Time **ms** (us_t v)
- static Time **us** (us_t v)
- static void **sleep** (us_t)

Static Public Attributes

- static const uint32_t **MS_TO_US** = 1000
- static const uint32_t **S_TO_MS** = 1000
- static const uint32_t **M_TO_S** = 60
- static const uint32_t **H_TO_M** = 60
- static const uint32_t **S_TO_US** = S_TO_MS ∗ MS_TO_US
- static const uint32_t **M_TO_US** = M_TO_S ∗ S_TO_US
- static const uint32_t **H_TO_US** = H_TO_M ∗ M_TO_US

5.148.1 Detailed Description

Data class representing a timeframe with microsecond accuracy.

Allows suspension of current thread via `sleep`. Convenient for intentional delays: - `Time::ms(500).wait()` suspends the currently active thread for 500ms.

5.149 lib::Timer Class Reference

Timer that allows scheduling of functors.

`#include <Timer.h>`

Classes

- struct **Ftor**

Public Types

- typedef uint64_t **ts_t**

Public Member Functions

- void sync (Time t)

  Synchronizes execution.
- void **reset** ()
- Time delta ()

  Amount of time elapsed since last reset.
- Time elapsed ()

Amount of time elapsed since last reset.

- template<typename F >
  void **executeWhen** (Time, F)
- bool active () const

  Is timer currently waiting for ftor execution.
- void **deactivate** ()

Static Public Member Functions

- static void deactivateAll ()

  Deactivate all timers.
- static ts_t timestamp ()

  Returns current system time in nanoseconds since Jan.

5.149.1 Detailed Description

Timer that allows scheduling of functors.

The functors will be executed after a specific amount of time in their own thread. - By resetting the Timer from within the supplied functor a periodic execution can be achieved.

Also allows for synchronisation to a specific time frame.

5.149.2 Member Function Documentation

5.149.2.1 bool lib::Timer::active (void) const

Is timer currently waiting for ftor execution.

5.149.2.2 void lib::Timer::deactivateAll (void) [static]

Deactivate all timers.

Prevents timing issues during the applications termination (i.e. waiting during `join` for a timer).

5.149.2.3 Time lib::Timer::delta (void)

Amount of time elapsed since last reset.

Resets timer.

5.149.2.4 Time lib::Timer::elapsed (void)

Amount of time elapsed since last reset.

Doesn't reset timer.

5.149.2.5 void lib::Timer::sync (Time *t*)

Synchronizes execution.

By suspending the current thread until `t` amount of time has part since the Timer has been started/reset this function (if called within a loop) synchronizes the active threads execution to a specific frequency ($f{1}{t}$f)

5.149.2.6 Timer::ts_t lib::Timer::timestamp (void) `[static]`

Returns current system time in nanoseconds since Jan.

1st 1970.

5.150 lib::TimerPoolImpl Class Reference

Public Types

- typedef Singleton< TimerPoolImpl > **SingletonInst**

Friends

- class **Timer**

5.151 lib::CreateTransitionMap< List >::Transform< T > Struct Template Reference

Public Types

- typedef Cons< Transition < typename T::Origin, typename T::Event, typename T::Destination >, T > **Type**

template<typename List>template<typename T> struct lib::CreateTransitionMap< List >::-Transform< T >

5.152 lib::TransImpl< E, D, L, S, T > Struct Template Reference

Public Types

- typedef TransImpl< E, D, DO(Cdr < L >), S, T > **Super**
- typedef E **Event**
- typedef D **Data**
- typedef S **StateList**
- typedef T **TransitionList**
- typedef TryCall_leave< Origin, Data > **LeaveFunction**
- typedef TryCall_enter < Destination, Data > **EnterFunction**
- typedef TryCall_apply< DO(GetValue < TransitionList, Transition < Origin, -Event, Destination > >), Event, Data > **TransitionFunction**

Public Member Functions

- typedef **DO** (Caar< L >) Origin
- typedef **DO** (Cdar< L >) Destination
- virtual void **process** (const Event &e)

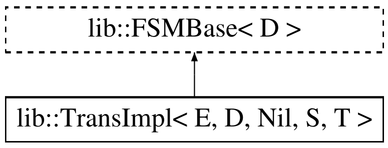
Static Public Attributes

- static const int **OriginID** = ValueIdentity<DO(GetValue<StateList, Origin>)>-::value
- static const int **DestinationID** = ValueIdentity<DO(GetValue<StateList, -Destination>)>::value
- static const bool **IsActualTransition** = !IsSame<Origin, Destination>::value

template<typename E, typename D, typename L, typename S, typename T> struct lib::TransImpl< E, D, L, S, T >

5.153 lib::TransImpl< E, D, Nil, S, T > Struct Template Reference

Inheritance diagram for lib::TransImpl< E, D, Nil, S, T >:



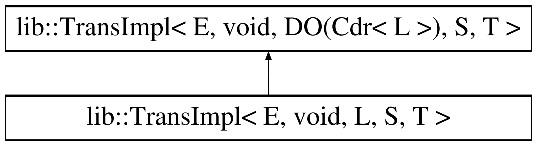
Public Member Functions

- virtual void **process** (const E &e)

template<typename E, typename D, typename S, typename T> struct lib::TransImpl< E, D, Nil, S, T >

5.154 lib::TransImpl< E, void, L, S, T > Struct Template Reference

Inheritance diagram for lib::TransImpl< E, void, L, S, T >:


lib::TransImpl< E, void, DO(Cdr< L >), S, T >
lib::TransImpl< E, void, L, S, T >

Public Types

- typedef TransImpl< E, void, DO(Cdr < L >), S, T > **Super**
- typedef E **Event**
- typedef S **StateList**
- typedef T **TransitionList**
- typedef TryCall_leave< Origin, void > **LeaveFunction**
- typedef TryCall_enter < Destination, void > **EnterFunction**
- typedef TryCall_apply< DO(GetValue < TransitionList, Transition < Origin, -Event, Destination > >), Event, void > **TransitionFunction**

Public Member Functions

- typedef **DO** (Caar< L >) Origin
- typedef **DO** (Cdar< L >) Destination
- virtual void **process** (const Event &e)

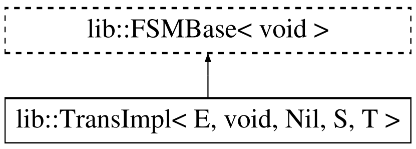
Static Public Attributes

- static const int **OriginID** = ValueIdentity<DO(GetValue<StateList, Origin>)>-::value
- static const int **DestinationID** = ValueIdentity<DO(GetValue<StateList, -Destination>)>::value
- static const bool **IsActualTransition** = !IsSame<Origin, Destination>::value

template<typename E, typename L, typename S, typename T> struct lib::TransImpl< E, void, L, S, T >

5.155 lib::TransImpl< E, void, Nil, S, T > Struct Template Reference

Inheritance diagram for lib::TransImpl< E, void, Nil, S, T >:



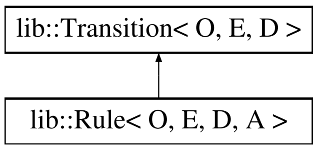
Public Member Functions

- virtual void **process** (const E &e)

template<typename E, typename S, typename T> struct lib::TransImpl< E, void, Nil, S, T >

5.156 lib::Transition< O, E, D > Struct Template Reference

Inheritance diagram for lib::Transition< O, E, D >:



Public Types

- typedef O **Origin**
- typedef E **Event**
- typedef D **Destination**

template<typename O, typename E, typename D> struct lib::Transition< O, E, D >

5.157 lib::TryCall_apply< T, E, D > Struct Template Reference

Classes

- struct Check

Static Public Member Functions

- template<typename TT >
  static void **test** (const E &e, D d, Check< TT,&TT::apply > ∗)
- template<typename >
  static void **test** (const E &e, D d,...)
- static void **call** (const E &e, D d)

template<typename T, typename E, typename D> struct lib::TryCall_apply< T, E, D >

5.158 lib::TryCall_apply< T, E, void > Struct Template Reference

Classes

- struct Check

Static Public Member Functions

- template<typename TT >
  static void **test** (const E &e, Check< TT,&TT::apply > ∗)
- template<typename >
  static void **test** (const E &e,...)
- static void **call** (const E &e)

template<typename T, typename E> struct lib::TryCall_apply< T, E, void >

5.159 lib::test::UnitTest Class Reference

Static Public Member Functions

- static void **assert_true** (bool, const std::string &, int, const char ∗=NULL)
- static void **assert_true** (bool f, const std::string &s, int l, const std::string &m)

5.160 lib::Value< T, I > Struct Template Reference

Static Public Attributes

- static const T **value** = I

template< typename T, T I > struct lib::Value< T, I >

5.161 lib::ValueIdentity< Bool< I > > Struct Template Reference

Static Public Attributes

- static const int **value** = I

template< bool I > struct lib::ValueIdentity< Bool< I > >

5.162 lib::ValueIdentity< Int< I > > Struct Template Reference

Static Public Attributes

- static const int **value** = I

template< int I > struct lib::ValueIdentity< Int< I > >

5.163 lib::ValueIdentity< Value< T, I > > Struct Template - Reference

Static Public Attributes

- static const T **value** = I

template< typename T, T I > struct lib::ValueIdentity< Value< T, I > >

5.164 lib::IsSuperType< Sub, Super >::Yes Struct Reference

Public Attributes

- char **v** [1]

template< typename Sub, typename Super > struct lib::IsSuperType< Sub, Super >::Yes