

TI4\_SE2

Generated by Doxygen 1.7.6.1

Wed Apr 27 2016 09:35:30



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>9</b>
3.1	Class List . . . . .	9
<b>4</b>	<b>Namespace Documentation</b>	<b>15</b>
4.1	lib::SingletonConcurrency Namespace Reference . . . . .	15
4.1.1	Detailed Description . . . . .	15
<b>5</b>	<b>Class Documentation</b>	<b>17</b>
5.1	hw::AcknowledgePacket Class Reference . . . . .	17
5.2	hw::Actuator Class Reference . . . . .	17
5.2.1	Detailed Description . . . . .	18
5.3	lib::And< T1, T2 > Struct Template Reference . . . . .	18
5.4	lib::ListAnd< List >::AndFn< T1, T2 > Struct Template Reference . . . . .	19
5.5	lib::Apply< F, T > Struct Template Reference . . . . .	19
5.6	lib::Apply< F, Nil > Struct Template Reference . . . . .	19
5.7	lib::Array< T, N > Class Template Reference . . . . .	19
5.8	lib::Array< T, 0 > Class Template Reference . . . . .	20
5.9	lib::log::BaseFilter Struct Reference . . . . .	20
5.10	lib::log::BaseFormatter Struct Reference . . . . .	21
5.11	lib::log::BaseHandler Struct Reference . . . . .	21
5.12	lib::BasicFunctor Struct Reference . . . . .	21

5.13	<a href="#">lib::BasicFunctorImpl&lt; F &gt; Class Template Reference</a>	22
5.13.1	<a href="#">Detailed Description</a>	22
5.14	<a href="#">lib::Bool&lt; V &gt; Struct Template Reference</a>	22
5.15	<a href="#">lib::Caar&lt; T &gt; Struct Template Reference</a>	22
5.16	<a href="#">lib::Cadr&lt; T &gt; Struct Template Reference</a>	23
5.17	<a href="#">lib::Car&lt; T &gt; Struct Template Reference</a>	23
5.18	<a href="#">lib::Cdar&lt; T &gt; Struct Template Reference</a>	23
5.19	<a href="#">lib::Cddr&lt; T &gt; Struct Template Reference</a>	23
5.20	<a href="#">lib::Cdr&lt; T &gt; Struct Template Reference</a>	23
5.21	<a href="#">lib::qnx::Channel Class Reference</a>	24
5.22	<a href="#">lib::TryCall_apply&lt; T, E, D &gt;::Check&lt; typename, &gt; Struct Template Reference</a>	24
5.23	<a href="#">lib::TryCall_apply&lt; T, E, void &gt;::Check&lt; typename, &gt; Struct Template Reference</a>	24
5.24	<a href="#">lib::CleanupUtility Class Reference</a>	24
5.24.1	<a href="#">Detailed Description</a>	25
5.25	<a href="#">lib::CreateTransitionDependencyList&lt; List &gt;::CollectDependencies&lt; E &gt; Struct Template Reference</a>	25
5.26	<a href="#">lib::Condition Class Reference</a>	25
5.27	<a href="#">hw::Connection Class Reference</a>	26
5.28	<a href="#">lib::qnx::Connection Class Reference</a>	26
5.29	<a href="#">lib::Cons&lt; H, T &gt; Struct Template Reference</a>	26
5.30	<a href="#">lib::ConsFn&lt; T1, T2 &gt; Struct Template Reference</a>	26
5.31	<a href="#">lib::ConstructFSMLineage&lt; T &gt; Struct Template Reference</a>	27
5.32	<a href="#">lib::ConstructFSMLineage&lt; Cons&lt; T, Nil &gt; &gt; Struct Template Reference</a>	27
5.33	<a href="#">lib::Contains&lt; List, T &gt; Struct Template Reference</a>	27
5.34	<a href="#">lib::Contains&lt; Nil, T &gt; Struct Template Reference</a>	27
5.35	<a href="#">lib::CreateStateList&lt; List &gt; Struct Template Reference</a>	27
5.36	<a href="#">lib::CreateTransitionDependencyList&lt; List &gt; Struct Template Reference</a>	28
5.37	<a href="#">lib::CreateTransitionMap&lt; List &gt; Struct Template Reference</a>	28
5.38	<a href="#">lib::FSMMaker&lt; I, D, T &gt;::CreateTransitionTree&lt; TT &gt; Struct Template Reference</a>	28
5.39	<a href="#">hw::DataPacket Class Reference</a>	29
5.40	<a href="#">lib::Decay&lt; T &gt; Struct Template Reference</a>	29
5.41	<a href="#">lib::Decay&lt; const T &gt; Struct Template Reference</a>	29

5.42 lib::Decay< const volatile T > Struct Template Reference . . . . .	30
5.43 lib::Decay< T & > Struct Template Reference . . . . .	30
5.44 lib::Decay< volatile T > Struct Template Reference . . . . .	30
5.45 lib::log::DefaultFormatter Class Reference . . . . .	30
5.45.1 Detailed Description . . . . .	31
5.46 hw::Motor::Direction Struct Reference . . . . .	31
5.47 lib::RingBufferConcurrency::MultiThreaded< T >::EmptyLock Class - Reference . . . . .	31
5.48 lib::RingBufferConcurrency::SingleThreaded< T >::EmptyLock Struct - Reference . . . . .	31
5.49 hw::ErrorPacket Class Reference . . . . .	31
5.50 lib::RingBufferConcurrency::SingleThreaded< T >::FillLock Struct - Reference . . . . .	32
5.51 lib::RingBufferConcurrency::MultiThreaded< T >::FillLock Class - Reference . . . . .	32
5.52 lib::log::Filter< F > Struct Template Reference . . . . .	32
5.52.1 Detailed Description . . . . .	33
5.53 lib::Filter< F, List > Struct Template Reference . . . . .	33
5.54 lib::Filter< F, Nil > Struct Template Reference . . . . .	33
5.55 lib::Flatten< T > Struct Template Reference . . . . .	34
5.56 lib::Flatten< Cons< H, T > > Struct Template Reference . . . . .	34
5.57 lib::log::Formatter< F > Struct Template Reference . . . . .	34
5.58 lib::Frequency Class Reference . . . . .	34
5.58.1 Detailed Description . . . . .	35
5.59 lib::FSM< ID, I, D, Lineage > Struct Template Reference . . . . .	35
5.60 lib::FSM< ID, I, void, Lineage > Struct Template Reference . . . . .	35
5.61 lib::FSMBase< D > Struct Template Reference . . . . .	35
5.62 lib::FSMBase< void > Struct Template Reference . . . . .	36
5.63 lib::FSMMaker< I, D, T > Struct Template Reference . . . . .	37
5.64 lib::FtorWrapper< T > Class Template Reference . . . . .	37
5.64.1 Detailed Description . . . . .	38
5.65 lib::CreateTransitionDependencyList< List >::GetDependency< T > - Struct Template Reference . . . . .	38
5.66 lib::GetElem< IDX, List > Struct Template Reference . . . . .	38
5.67 lib::GetElem< 0, List > Struct Template Reference . . . . .	38

5.68	<code>lib::CreateStateList&lt; List &gt;::GetStateFromTransition&lt; T &gt; Struct - Template Reference</code>	38
5.69	<code>lib::GetValue&lt; Map, Key &gt; Struct Template Reference</code>	39
5.70	<code>lib::log::Handler&lt; F &gt; Class Template Reference</code>	39
5.70.1	Detailed Description	39
5.71	<code>hw::HWAccessImpl Class Reference</code>	40
5.71.1	Detailed Description	40
5.72	<code>lib::Identity&lt; T &gt; Struct Template Reference</code>	41
5.73	<code>lib::If&lt; false, T1, T2 &gt; Struct Template Reference</code>	41
5.74	<code>lib::If&lt; true, T1, T2 &gt; Struct Template Reference</code>	41
5.75	<code>lib::InheritLineage&lt; T &gt; Struct Template Reference</code>	41
5.76	<code>lib::InheritLineage&lt; Nil &gt; Struct Template Reference</code>	41
5.77	<code>lib::Int&lt; I &gt; Struct Template Reference</code>	41
5.78	<code>lib::CreateTransitionDependencyList&lt; List &gt;::CollectDependencies&lt; E &gt;::IsCorrectEvent&lt; T &gt; Struct Template Reference</code>	42
5.79	<code>lib::IsList&lt; T &gt; Struct Template Reference</code>	42
5.80	<code>lib::IsList&lt; Cons&lt; T1, T2 &gt; &gt; Struct Template Reference</code>	42
5.81	<code>lib::IsSame&lt; T1, T2 &gt; Struct Template Reference</code>	42
5.82	<code>lib::IsSame&lt; T, T &gt; Struct Template Reference</code>	43
5.83	<code>lib::IsSuperType&lt; Sub, Super &gt; Struct Template Reference</code>	43
5.84	<code>lib::Join&lt; List, Appendage &gt; Struct Template Reference</code>	43
5.85	<code>lib::Join&lt; Nil, Appendage &gt; Struct Template Reference</code>	43
5.86	<code>hw::LED Class Reference</code>	44
5.86.1	Detailed Description	45
5.87	<code>lib::ListAnd&lt; List &gt; Struct Template Reference</code>	45
5.88	<code>lib::ListOr&lt; List &gt; Struct Template Reference</code>	45
5.89	<code>lib::ListToMap&lt; List &gt; Struct Template Reference</code>	46
5.90	<code>lib::ListToMapImpl&lt; IDX, List &gt; Struct Template Reference</code>	46
5.91	<code>lib::ListToMapImpl&lt; IDX, Nil &gt; Struct Template Reference</code>	46
5.92	<code>lib::SingletonConcurrency::SingleThreaded&lt; T &gt;::Lock Struct Reference</code>	46
5.93	<code>lib::SingletonConcurrency::MultiThreaded&lt; T &gt;::Lock Struct Reference</code>	46
5.94	<code>hw::HWAccessImpl::Lock Struct Reference</code>	47
5.95	<code>lib::Lock&lt; T, E, R &gt; Class Template Reference</code>	47
5.96	<code>lib::LockableClass&lt; T, M &gt;::Lock Struct Reference</code>	47

5.97 lib::LockableObject< T, M >::Lock Struct Reference . . . . .	47
5.98 lib::LockableClass< T, M > Class Template Reference . . . . .	47
5.99 lib::LockableObject< T, M > Class Template Reference . . . . .	48
5.100lib::test::TestManager::Log Struct Reference . . . . .	48
5.101lib::log::Logger Class Reference . . . . .	48
5.101.1 Detailed Description . . . . .	49
5.102lib::log::LogLevel Class Reference . . . . .	49
5.103lib::log::LogManagerImpl Class Reference . . . . .	50
5.103.1 Detailed Description . . . . .	50
5.104lib::log::LogRecord Class Reference . . . . .	50
5.105lib::MakeList_0 Struct Reference . . . . .	51
5.106lib::MakeList_1< A > Struct Template Reference . . . . .	51
5.107lib::MakeList_2< A, B > Struct Template Reference . . . . .	51
5.108lib::MakeList_3< A, B, C > Struct Template Reference . . . . .	51
5.109lib::MakeList_4< A, B, C, D > Struct Template Reference . . . . .	52
5.110lib::MakeList_5< A, B, C, D, E > Struct Template Reference . . . . .	52
5.111lib::MakeList_6< A, B, C, D, E, F > Struct Template Reference . . . . .	52
5.112lib::MakeList_7< A, B, C, D, E, F, G > Struct Template Reference . . . . .	52
5.113lib::MakeList_8< A, B, C, D, E, F, G, H > Struct Template Reference . . . . .	53
5.114lib::MakeList_9< A, B, C, D, E, F, G, H, I > Struct Template Reference . . . . .	53
5.115lib::Merge< F, T, List > Struct Template Reference . . . . .	53
5.116lib::Merge< F, T, Nil > Struct Template Reference . . . . .	53
5.117hw::Motor Class Reference . . . . .	53
5.117.1 Detailed Description . . . . .	54
5.117.2 Member Function Documentation . . . . .	55
5.117.2.1 controlBelt . . . . .	55
5.117.2.2 controlSwitch . . . . .	55
5.118lib::SingletonConcurrency::MultiThreaded< T > Struct Template - Reference . . . . .	55
5.119lib::RingBufferConcurrency::MultiThreaded< T > Class Template - Reference . . . . .	55
5.120lib::Mutex Class Reference . . . . .	56
5.121lib::Nil Struct Reference . . . . .	56
5.122lib::IsSuperType< Sub, Super >::No Struct Reference . . . . .	56

5.123lib::Not< T > Struct Template Reference . . . . .	56
5.124hw::OKPacket Class Reference . . . . .	57
5.125lib::Or< T1, T2 > Struct Template Reference . . . . .	57
5.126lib::ListOr< List >::OrFn< T1, T2 > Struct Template Reference . . . . .	57
5.127hw::Packet Class Reference . . . . .	58
5.128lib::qnx::Receiver Class Reference . . . . .	58
5.129lib::test::TestManager::Registrar Struct Reference . . . . .	58
5.130lib::Reverse< List > Struct Template Reference . . . . .	59
5.131lib::ReverseCons< Cell > Struct Template Reference . . . . .	59
5.132lib::ReverselImpl< Done, ToDo > Struct Template Reference . . . . .	59
5.133lib::ReverselImpl< Done, Nil > Struct Template Reference . . . . .	59
5.134lib::RingBuffer< T, N, ThreadingPolicy > Class Template Reference . . . . .	59
5.135lib::Rule< O, E, D, A > Struct Template Reference . . . . .	60
5.136lib::test::TestManager::Selector Struct Reference . . . . .	60
5.137lib::Semaphore Class Reference . . . . .	60
5.138lib::Setify< List > Struct Template Reference . . . . .	60
5.139lib::SetifyImpl< Done, ToDo > Struct Template Reference . . . . .	61
5.140lib::SetifyImpl< Done, Nil > Struct Template Reference . . . . .	61
5.141lib::RingBufferConcurrency::SingleThreaded< T > Class Template - Reference . . . . .	61
5.142lib::SingletonConcurrency::SingleThreaded< T > Struct Template - Reference . . . . .	61
5.143lib::Singleton< T, TM, P > Class Template Reference . . . . .	62
5.143.1 Detailed Description . . . . .	62
5.143.2 Member Function Documentation . . . . .	62
5.143.2.1 instance . . . . .	62
5.144lib::SmartPtr< T > Class Template Reference . . . . .	62
5.144.1 Detailed Description . . . . .	63
5.145hw::Motor::Speed Struct Reference . . . . .	63
5.146hw::Motor::State Struct Reference . . . . .	64
5.147lib::log::StreamHandler Class Reference . . . . .	64
5.147.1 Detailed Description . . . . .	64
5.148lib::test::TestManager Class Reference . . . . .	64
5.149lib::Thread Class Reference . . . . .	65



5.149.1 Detailed Description . . . . .	66
5.149.2 Constructor & Destructor Documentation . . . . .	66
5.149.2.1 Thread . . . . .	66
5.149.2.2 Thread . . . . .	66
5.149.2.3 ~Thread . . . . .	66
5.149.3 Member Function Documentation . . . . .	66
5.149.3.1 join . . . . .	66
5.149.3.2 joinable . . . . .	66
5.149.3.3 operator= . . . . .	67
5.149.3.4 run . . . . .	67
5.150lib::ThreadManagerImpl Class Reference . . . . .	67
5.151lib::Time Class Reference . . . . .	67
5.151.1 Detailed Description . . . . .	68
5.152lib::Timer Class Reference . . . . .	68
5.152.1 Detailed Description . . . . .	69
5.152.2 Member Function Documentation . . . . .	69
5.152.2.1 active . . . . .	69
5.152.2.2 deactivateAll . . . . .	70
5.152.2.3 delta . . . . .	70
5.152.2.4 elapsed . . . . .	70
5.152.2.5 sync . . . . .	70
5.152.2.6 timestamp . . . . .	70
5.153lib::TimerPoolImpl Class Reference . . . . .	70
5.154lib::CreateTransitionMap< List >::Transform< T > Struct Template - Reference . . . . .	71
5.155lib::TransImpl< E, D, L, S, T > Struct Template Reference . . . . .	71
5.156lib::TransImpl< E, D, Nil, S, T > Struct Template Reference . . . . .	72
5.157lib::TransImpl< E, void, L, S, T > Struct Template Reference . . . . .	72
5.158lib::TransImpl< E, void, Nil, S, T > Struct Template Reference . . . . .	73
5.159lib::Transition< O, E, D > Struct Template Reference . . . . .	73
5.160lib::TryCall_apply< T, E, D > Struct Template Reference . . . . .	74
5.161lib::TryCall_apply< T, E, void > Struct Template Reference . . . . .	74
5.162lib::test::UnitTest Class Reference . . . . .	74
5.163lib::Value< T, I > Struct Template Reference . . . . .	75

5.164lib::Valueldentity< Bool< I > > Struct Template Reference . . . . .	75
5.165lib::Valueldentity< Int< I > > Struct Template Reference . . . . .	75
5.166lib::Valueldentity< Value< T, I > > Struct Template Reference . . . . .	75
5.167lib::IsSuperType< Sub, Super >::Yes Struct Reference . . . . .	75

# Chapter 1

## Namespace Index

### 1.1 Namespace List

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

<a href="#">lib::SingletonConcurrency</a>	
<a href="#">Contains</a> threading models of the <a href="#">Singleton</a> template	15



## Chapter 2

# Class Index

### 2.1 Class Hierarchy

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

lib::And< T1, T2 > . . . . .	18
lib::ListAnd< List >::AndFn< T1, T2 > . . . . .	19
lib::Apply< F, T > . . . . .	19
lib::Apply< F, Nil > . . . . .	19
lib::Array< T, N > . . . . .	19
lib::Array< T, 0 > . . . . .	20
lib::log::BaseFilter . . . . .	20
lib::log::Filter< F > . . . . .	32
lib::log::BaseFormatter . . . . .	21
lib::log::Formatter< F > . . . . .	34
lib::log::BaseHandler . . . . .	21
lib::log::Handler< F > . . . . .	39
lib::BasicFunctor . . . . .	21
lib::BasicFunctorImpl< F > . . . . .	22
lib::Caar< T > . . . . .	22
lib::Cadr< T > . . . . .	23
lib::Car< T > . . . . .	23
lib::Cdar< T > . . . . .	23
lib::Cddr< T > . . . . .	23
lib::Cdr< T > . . . . .	23
lib::qnx::Channel . . . . .	24
lib::TryCall_apply< T, E, D >::Check< typename, > . . . . .	24
lib::TryCall_apply< T, E, void >::Check< typename, > . . . . .	24
lib::CreateTransitionDependencyList< List >::CollectDependencies< E > . . . . .	25
lib::Condition . . . . .	25
hw::Connection . . . . .	26
lib::qnx::Connection . . . . .	26
lib::Cons< H, T > . . . . .	26

lib::ConsFn< T1, T2 > . . . . .	26
lib::ConstructFSMLineage< T > . . . . .	27
lib::ConstructFSMLineage< Cons< T, Nil > > . . . . .	27
lib::Contains< List, T > . . . . .	27
lib::Contains< Nil, T > . . . . .	27
lib::CreateStateList< List > . . . . .	27
lib::CreateTransitionDependencyList< List > . . . . .	28
lib::CreateTransitionMap< List > . . . . .	28
lib::FSMMaker< I, D, T >::CreateTransitionTree< TT > . . . . .	28
lib::Decay< T > . . . . .	29
lib::Decay< const T > . . . . .	29
lib::Decay< const volatile T > . . . . .	30
lib::Decay< T & > . . . . .	30
lib::Decay< volatile T > . . . . .	30
hw::Motor::Direction . . . . .	31
lib::RingBufferConcurrency::MultiThreaded< T >::EmptyLock . . . . .	31
lib::RingBufferConcurrency::SingleThreaded< T >::EmptyLock . . . . .	31
lib::RingBufferConcurrency::SingleThreaded< T >::FillLock . . . . .	32
lib::RingBufferConcurrency::MultiThreaded< T >::FillLock . . . . .	32
lib::Filter< F, List > . . . . .	33
lib::Filter< F, Nil > . . . . .	33
lib::Flatten< T > . . . . .	34
lib::Flatten< Cons< H, T > > . . . . .	34
lib::Frequency . . . . .	34
lib::FSM< ID, I, D, Lineage > . . . . .	35
lib::FSM< ID, I, void, Lineage > . . . . .	35
lib::FSMBase< D > . . . . .	35
lib::TransImpl< E, D, Nil, S, T > . . . . .	72
lib::FSMBase< void > . . . . .	36
lib::TransImpl< E, void, Nil, S, T > . . . . .	73
lib::FSMMaker< I, D, T > . . . . .	37
lib::FtorWrapper< T > . . . . .	37
lib::CreateTransitionDependencyList< List >::GetDependency< T > . . . . .	38
lib::GetElem< IDX, List > . . . . .	38
lib::GetElem< 0, List > . . . . .	38
lib::CreateStateList< List >::GetStateFromTransition< T > . . . . .	38
lib::GetValue< Map, Key > . . . . .	39
lib::Identity< T > . . . . .	41
lib::If< false, T1, T2 > . . . . .	41
lib::If< true, T1, T2 > . . . . .	41
lib::InheritLineage< T > . . . . .	41
lib::InheritLineage< Nil > . . . . .	41
lib::IsList< T > . . . . .	42
lib::IsList< Cons< T1, T2 > > . . . . .	42
lib::IsSame< T1, T2 > . . . . .	42
lib::IsSame< DO(Car< T >), E > . . . . .	42
lib::CreateTransitionDependencyList< List >::CollectDependencies< E >::IsCorrectEvent< T > . . . . .	42
lib::IsSame< T, T > . . . . .	43

lib::IsSuperType< Sub, Super > . . . . .	43
lib::Join< List, Appendage > . . . . .	43
lib::Join< Nil, Appendage > . . . . .	43
lib::ListAnd< List > . . . . .	45
lib::ListOr< List > . . . . .	45
lib::ListToMap< List > . . . . .	46
lib::ListToMapImpl< IDX, List > . . . . .	46
lib::ListToMapImpl< IDX, Nil > . . . . .	46
lib::SingletonConcurrency::SingleThreaded< T >::Lock . . . . .	46
lib::SingletonConcurrency::MultiThreaded< T >::Lock . . . . .	46
hw::HWAccessImpl::Lock . . . . .	47
lib::Lock< T, E, R > . . . . .	47
lib::LockableClass< T, M >::Lock . . . . .	47
lib::LockableObject< T, M >::Lock . . . . .	47
lib::LockableClass< T, M > . . . . .	47
lib::LockableClass< HWAccessImpl > . . . . .	47
hw::HWAccessImpl . . . . .	40
lib::LockableClass< T > . . . . .	47
lib::SingletonConcurrency::MultiThreaded< T > . . . . .	55
lib::LockableObject< T, M > . . . . .	48
lib::LockableObject< Actuator > . . . . .	48
hw::Actuator . . . . .	17
lib::LockableObject< CleanupUtility > . . . . .	48
lib::CleanupUtility . . . . .	24
lib::LockableObject< DefaultFormatter > . . . . .	48
lib::log::DefaultFormatter . . . . .	30
lib::LockableObject< LED > . . . . .	48
hw::LED . . . . .	44
lib::LockableObject< Logger > . . . . .	48
lib::log::Logger . . . . .	48
lib::LockableObject< LogManagerImpl > . . . . .	48
lib::log::LogManagerImpl . . . . .	50
lib::LockableObject< Motor > . . . . .	48
hw::Motor . . . . .	53
lib::LockableObject< SmartPtr< T > > . . . . .	48
lib::SmartPtr< T > . . . . .	62
lib::LockableObject< StreamHandler > . . . . .	48
lib::log::StreamHandler . . . . .	64
lib::LockableObject< ThreadManagerImpl > . . . . .	48
lib::ThreadManagerImpl . . . . .	67
lib::test::TestManager::Log . . . . .	48
lib::log::LogLevel . . . . .	49
lib::log::LogRecord . . . . .	50
lib::MakeList_0 . . . . .	51
lib::MakeList_1< A > . . . . .	51
lib::MakeList_2< A, B > . . . . .	51

lib::MakeList_3< A, B, C > . . . . .	51
lib::MakeList_4< A, B, C, D > . . . . .	52
lib::MakeList_5< A, B, C, D, E > . . . . .	52
lib::MakeList_6< A, B, C, D, E, F > . . . . .	52
lib::MakeList_7< A, B, C, D, E, F, G > . . . . .	52
lib::MakeList_8< A, B, C, D, E, F, G, H > . . . . .	53
lib::MakeList_9< A, B, C, D, E, F, G, H, I > . . . . .	53
lib::Merge< F, T, List > . . . . .	53
lib::Merge< F, T, Nil > . . . . .	53
lib::RingBufferConcurrency::MultiThreaded< T > . . . . .	55
lib::Mutex . . . . .	56
lib::Nil . . . . .	56
lib::IsSuperType< Sub, Super >::No . . . . .	56
lib::Not< T > . . . . .	56
lib::Or< T1, T2 > . . . . .	57
lib::ListOr< List >::OrFn< T1, T2 > . . . . .	57
hw::Packet . . . . .	58
hw::AcknowledgePacket . . . . .	17
hw::DataPacket . . . . .	29
hw::ErrorPacket . . . . .	31
hw::OKPacket . . . . .	57
lib::qnx::Receiver . . . . .	58
lib::test::TestManager::Registrar . . . . .	58
lib::Reverse< List > . . . . .	59
lib::ReverseCons< Cell > . . . . .	59
lib::ReverseImpl< Done, ToDo > . . . . .	59
lib::ReverseImpl< Done, Nil > . . . . .	59
lib::RingBuffer< T, N, ThreadingPolicy > . . . . .	59
lib::test::TestManager::Selector . . . . .	60
lib::Semaphore . . . . .	60
lib::Setify< List > . . . . .	60
lib::SetifyImpl< Done, ToDo > . . . . .	61
lib::SetifyImpl< Done, Nil > . . . . .	61
lib::RingBufferConcurrency::SingleThreaded< T > . . . . .	61
lib::SingletonConcurrency::SingleThreaded< T > . . . . .	61
lib::Singleton< T, TM, P > . . . . .	62
hw::Motor::Speed . . . . .	63
hw::Motor::State . . . . .	64
lib::test::TestManager . . . . .	64
lib::Thread . . . . .	65
lib::Time . . . . .	67
lib::Timer . . . . .	68
lib::TimerPoolImpl . . . . .	70
lib::CreateTransitionMap< List >::Transform< T > . . . . .	71
lib::TransImpl< E, D, L, S, T > . . . . .	71
lib::TransImpl< E, void, DO(Cdr< L >), S, T > . . . . .	71
lib::TransImpl< E, void, L, S, T > . . . . .	72
lib::Transition< O, E, D > . . . . .	73
lib::Rule< O, E, D, A > . . . . .	60



lib::TryCall_apply< T, E, D > . . . . .	74
lib::TryCall_apply< T, E, void > . . . . .	74
lib::test::UnitTest . . . . .	74
lib::Value< T, I > . . . . .	75
lib::Value< bool, V > . . . . .	75
lib::Bool< V > . . . . .	22
lib::Value< int, I > . . . . .	75
lib::Int< I > . . . . .	41
lib::ValueIdentity< Bool< I > > . . . . .	75
lib::ValueIdentity< Int< I > > . . . . .	75
lib::ValueIdentity< Value< T, I > > . . . . .	75
lib::IsSuperType< Sub, Super >::Yes . . . . .	75



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">hw::AcknowledgePacket</a>	17
<a href="#">hw::Actuator</a>	
Singular access point to actuators	17
<a href="#">lib::And&lt; T1, T2 &gt;</a>	18
<a href="#">lib::ListAnd&lt; List &gt;::AndFn&lt; T1, T2 &gt;</a>	19
<a href="#">lib::Apply&lt; F, T &gt;</a>	19
<a href="#">lib::Apply&lt; F, Nil &gt;</a>	19
<a href="#">lib::Array&lt; T, N &gt;</a>	19
<a href="#">lib::Array&lt; T, 0 &gt;</a>	20
<a href="#">lib::log::BaseFilter</a>	20
<a href="#">lib::log::BaseFormatter</a>	21
<a href="#">lib::log::BaseHandler</a>	21
<a href="#">lib::BasicFunctor</a>	21
<a href="#">lib::BasicFunctorImpl&lt; F &gt;</a>	
Basic functor encapsulating anything callable that takes no arguments	22
<a href="#">lib::Bool&lt; V &gt;</a>	22
<a href="#">lib::Caar&lt; T &gt;</a>	22
<a href="#">lib::Cadr&lt; T &gt;</a>	23
<a href="#">lib::Car&lt; T &gt;</a>	23
<a href="#">lib::Cdar&lt; T &gt;</a>	23
<a href="#">lib::Cddr&lt; T &gt;</a>	23
<a href="#">lib::Cdr&lt; T &gt;</a>	23
<a href="#">lib::qnx::Channel</a>	24
<a href="#">lib::TryCall_apply&lt; T, E, D &gt;::Check&lt; typename, &gt;</a>	24
<a href="#">lib::TryCall_apply&lt; T, E, void &gt;::Check&lt; typename, &gt;</a>	24
<a href="#">lib::CleanupUtility</a>	
Utility for controlling the lifetime of static objects, i.e	24
<a href="#">lib::CreateTransitionDependencyList&lt; List &gt;::CollectDependencies&lt; E &gt;</a>	25

<a href="#">lib::Condition</a>	25
<a href="#">hw::Connection</a>	26
<a href="#">lib::qnx::Connection</a>	26
<a href="#">lib::Cons&lt; H, T &gt;</a>	26
<a href="#">lib::ConsFn&lt; T1, T2 &gt;</a>	26
<a href="#">lib::ConstructFSMLineage&lt; T &gt;</a>	27
<a href="#">lib::ConstructFSMLineage&lt; Cons&lt; T, Nil &gt; &gt;</a>	27
<a href="#">lib::Contains&lt; List, T &gt;</a>	27
<a href="#">lib::Contains&lt; Nil, T &gt;</a>	27
<a href="#">lib::CreateStateList&lt; List &gt;</a>	27
<a href="#">lib::CreateTransitionDependencyList&lt; List &gt;</a>	28
<a href="#">lib::CreateTransitionMap&lt; List &gt;</a>	28
<a href="#">lib::FSMMaker&lt; I, D, T &gt;::CreateTransitionTree&lt; TT &gt;</a>	28
<a href="#">hw::DataPacket</a>	29
<a href="#">lib::Decay&lt; T &gt;</a>	29
<a href="#">lib::Decay&lt; const T &gt;</a>	29
<a href="#">lib::Decay&lt; const volatile T &gt;</a>	30
<a href="#">lib::Decay&lt; T &amp; &gt;</a>	30
<a href="#">lib::Decay&lt; volatile T &gt;</a>	30
<a href="#">lib::log::DefaultFormatter</a>	
Default formatter that lists all information of the passed <a href="#">LogRecord</a>	30
<a href="#">hw::Motor::Direction</a>	31
<a href="#">lib::RingBufferConcurrency::MultiThreaded&lt; T &gt;::EmptyLock</a>	31
<a href="#">lib::RingBufferConcurrency::SingleThreaded&lt; T &gt;::EmptyLock</a>	31
<a href="#">hw::ErrorPacket</a>	31
<a href="#">lib::RingBufferConcurrency::SingleThreaded&lt; T &gt;::FillLock</a>	32
<a href="#">lib::RingBufferConcurrency::MultiThreaded&lt; T &gt;::FillLock</a>	32
<a href="#">lib::log::Filter&lt; F &gt;</a>	
Filter template that accepts functors	32
<a href="#">lib::Filter&lt; F, List &gt;</a>	33
<a href="#">lib::Filter&lt; F, Nil &gt;</a>	33
<a href="#">lib::Flatten&lt; T &gt;</a>	34
<a href="#">lib::Flatten&lt; Cons&lt; H, T &gt; &gt;</a>	34
<a href="#">lib::log::Formatter&lt; F &gt;</a>	34
<a href="#">lib::Frequency</a>	
Convenience class that allows calculation of a signal's period length through its frequency	34
<a href="#">lib::FSM&lt; ID, I, D, Lineage &gt;</a>	35
<a href="#">lib::FSM&lt; ID, I, void, Lineage &gt;</a>	35
<a href="#">lib::FSMBase&lt; D &gt;</a>	35
<a href="#">lib::FSMBase&lt; void &gt;</a>	36
<a href="#">lib::FSMMaker&lt; I, D, T &gt;</a>	37
<a href="#">lib::FtorWrapper&lt; T &gt;</a>	
A functor that calls an object's member function	37
<a href="#">lib::CreateTransitionDependencyList&lt; List &gt;::GetDependency&lt; T &gt;</a>	38
<a href="#">lib::GetElem&lt; IDX, List &gt;</a>	38
<a href="#">lib::GetElem&lt; 0, List &gt;</a>	38
<a href="#">lib::CreateStateList&lt; List &gt;::GetStateFromTransition&lt; T &gt;</a>	38
<a href="#">lib::GetValue&lt; Map, Key &gt;</a>	39

<a href="#">lib::log::Handler&lt; F &gt;</a>	
Handler template that holds a functor	39
<a href="#">hw::HWAcessImpl</a>	
Interface for direct hardware access	40
<a href="#">lib::Identity&lt; T &gt;</a>	41
<a href="#">lib::If&lt; false, T1, T2 &gt;</a>	41
<a href="#">lib::If&lt; true, T1, T2 &gt;</a>	41
<a href="#">lib::InheritLineage&lt; T &gt;</a>	41
<a href="#">lib::InheritLineage&lt; Nil &gt;</a>	41
<a href="#">lib::Int&lt; I &gt;</a>	41
<a href="#">lib::CreateTransitionDependencyList&lt; List &gt;::CollectDependencies&lt; E &gt;:-</a>	
IsCorrectEvent< T >	42
<a href="#">lib::IsList&lt; T &gt;</a>	42
<a href="#">lib::IsList&lt; Cons&lt; T1, T2 &gt; &gt;</a>	42
<a href="#">lib::IsSame&lt; T1, T2 &gt;</a>	42
<a href="#">lib::IsSame&lt; T, T &gt;</a>	43
<a href="#">lib::IsSuperType&lt; Sub, Super &gt;</a>	43
<a href="#">lib::Join&lt; List, Appendage &gt;</a>	43
<a href="#">lib::Join&lt; Nil, Appendage &gt;</a>	43
<a href="#">hw::LED</a>	
Allows access to LEDs	44
<a href="#">lib::ListAnd&lt; List &gt;</a>	45
<a href="#">lib::ListOr&lt; List &gt;</a>	45
<a href="#">lib::ListToMap&lt; List &gt;</a>	46
<a href="#">lib::ListToMapImpl&lt; IDX, List &gt;</a>	46
<a href="#">lib::ListToMapImpl&lt; IDX, Nil &gt;</a>	46
<a href="#">lib::SingletonConcurrency::SingleThreaded&lt; T &gt;::Lock</a>	46
<a href="#">lib::SingletonConcurrency::MultiThreaded&lt; T &gt;::Lock</a>	46
<a href="#">hw::HWAcessImpl::Lock</a>	47
<a href="#">lib::Lock&lt; T, E, R &gt;</a>	47
<a href="#">lib::LockableClass&lt; T, M &gt;::Lock</a>	47
<a href="#">lib::LockableObject&lt; T, M &gt;::Lock</a>	47
<a href="#">lib::LockableClass&lt; T, M &gt;</a>	47
<a href="#">lib::LockableObject&lt; T, M &gt;</a>	48
<a href="#">lib::test::TestManager::Log</a>	48
<a href="#">lib::log::Logger</a>	
Logger class	48
<a href="#">lib::log::LogLevel</a>	49
<a href="#">lib::log::LogManagerImpl</a>	
LogManager Singleton, grants access to <a href="#">Logger</a> instances	50
<a href="#">lib::log::LogRecord</a>	50
<a href="#">lib::MakeList_0</a>	51
<a href="#">lib::MakeList_1&lt; A &gt;</a>	51
<a href="#">lib::MakeList_2&lt; A, B &gt;</a>	51
<a href="#">lib::MakeList_3&lt; A, B, C &gt;</a>	51
<a href="#">lib::MakeList_4&lt; A, B, C, D &gt;</a>	52
<a href="#">lib::MakeList_5&lt; A, B, C, D, E &gt;</a>	52
<a href="#">lib::MakeList_6&lt; A, B, C, D, E, F &gt;</a>	52
<a href="#">lib::MakeList_7&lt; A, B, C, D, E, F, G &gt;</a>	52
<a href="#">lib::MakeList_8&lt; A, B, C, D, E, F, G, H &gt;</a>	53

<a href="#">lib::MakeList_9&lt; A, B, C, D, E, F, G, H, I &gt;</a>	53
<a href="#">lib::Merge&lt; F, T, List &gt;</a>	53
<a href="#">lib::Merge&lt; F, T, Nil &gt;</a>	53
<a href="#">hw::Motor</a>	
Client interface for controlling the conveyor belt and electromagnetic switch	53
<a href="#">lib::SingletonConcurrency::MultiThreaded&lt; T &gt;</a>	55
<a href="#">lib::RingBufferConcurrency::MultiThreaded&lt; T &gt;</a>	55
<a href="#">lib::Mutex</a>	56
<a href="#">lib::Nil</a>	56
<a href="#">lib::IsSuperType&lt; Sub, Super &gt;::No</a>	56
<a href="#">lib::Not&lt; T &gt;</a>	56
<a href="#">hw::OKPacket</a>	57
<a href="#">lib::Or&lt; T1, T2 &gt;</a>	57
<a href="#">lib::ListOr&lt; List &gt;::OrFn&lt; T1, T2 &gt;</a>	57
<a href="#">hw::Packet</a>	58
<a href="#">lib::qnx::Receiver</a>	58
<a href="#">lib::test::TestManager::Registrar</a>	58
<a href="#">lib::Reverse&lt; List &gt;</a>	59
<a href="#">lib::ReverseCons&lt; Cell &gt;</a>	59
<a href="#">lib::ReverseImpl&lt; Done, ToDo &gt;</a>	59
<a href="#">lib::ReverseImpl&lt; Done, Nil &gt;</a>	59
<a href="#">lib::RingBuffer&lt; T, N, ThreadingPolicy &gt;</a>	59
<a href="#">lib::Rule&lt; O, E, D, A &gt;</a>	60
<a href="#">lib::test::TestManager::Selector</a>	60
<a href="#">lib::Semaphore</a>	60
<a href="#">lib::Setify&lt; List &gt;</a>	60
<a href="#">lib::SetifyImpl&lt; Done, ToDo &gt;</a>	61
<a href="#">lib::SetifyImpl&lt; Done, Nil &gt;</a>	61
<a href="#">lib::RingBufferConcurrency::SingleThreaded&lt; T &gt;</a>	61
<a href="#">lib::SingletonConcurrency::SingleThreaded&lt; T &gt;</a>	61
<a href="#">lib::Singleton&lt; T, TM, P &gt;</a>	
Template for convenient <a href="#">Singleton</a> creation	62
<a href="#">lib::SmartPtr&lt; T &gt;</a>	
Smart pointer class for automatic life time management	62
<a href="#">hw::Motor::Speed</a>	63
<a href="#">hw::Motor::State</a>	64
<a href="#">lib::log::StreamHandler</a>	
Handler compatible functor that writes its <a href="#">LogRecord</a> to an std- ::stream instance	64
<a href="#">lib::test::TestManager</a>	64
<a href="#">lib::Thread</a>	
Encapsulates the most important features of a thread	65
<a href="#">lib::ThreadManagerImpl</a>	67
<a href="#">lib::Time</a>	
Data class representing a timeframe with microsecond accuracy	67
<a href="#">lib::Timer</a>	
Timer that allows scheduling of functors	68
<a href="#">lib::TimerPoolImpl</a>	70
<a href="#">lib::CreateTransitionMap&lt; List &gt;::Transform&lt; T &gt;</a>	71

lib::TransImpl< E, D, L, S, T > . . . . .	71
lib::TransImpl< E, D, Nil, S, T > . . . . .	72
lib::TransImpl< E, void, L, S, T > . . . . .	72
lib::TransImpl< E, void, Nil, S, T > . . . . .	73
lib::Transition< O, E, D > . . . . .	73
lib::TryCall_apply< T, E, D > . . . . .	74
lib::TryCall_apply< T, E, void > . . . . .	74
lib::test::UnitTest . . . . .	74
lib::Value< T, I > . . . . .	75
lib::ValueIdentity< Bool< I > > . . . . .	75
lib::ValueIdentity< Int< I > > . . . . .	75
lib::ValueIdentity< Value< T, I > > . . . . .	75
lib::IsSuperType< Sub, Super >::Yes . . . . .	75





## 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.

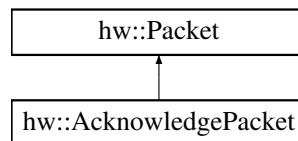


## Chapter 5

# Class Documentation

### 5.1 hw::AcknowledgePacket Class Reference

Inheritance diagram for hw::AcknowledgePacket:



#### Public Member Functions

- **AcknowledgePacket** (uint32\_t s)
- uint32\_t **size** () const
- const uint8\_t \* **data** () const
- uint8\_t **id** () const
- uint32\_t **hash** () const

#### Static Public Member Functions

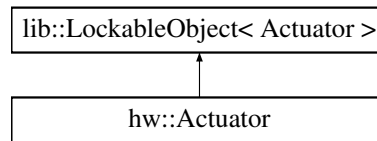
- static [Packet\\_ptr](#) **assemble** (const uint8\_t \*d, uint32\_t s)

### 5.2 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

#### 5.2.1 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.

### 5.3 [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 >
```

## 5.4 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 >
```

## 5.5 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 >
```

## 5.6 lib::Apply< F, Nil > Struct Template Reference

### Public Types

- typedef [Nil](#) **Type**

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

## 5.7 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

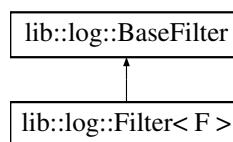
```
template<typename T, std::size_t N> class lib::Array< T, N >
```

### 5.8 lib::Array< T, 0 > Class Template Reference

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

### 5.9 lib::log::BaseFilter Struct Reference

Inheritance diagram for lib::log::BaseFilter:

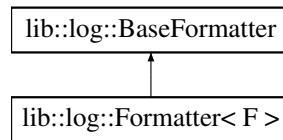


### Public Member Functions

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

## 5.10 lib::log::BaseFormatter Struct Reference

Inheritance diagram for lib::log::BaseFormatter:

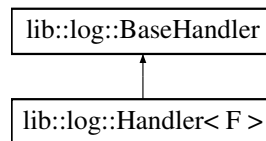


### Public Member Functions

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

## 5.11 lib::log::BaseHandler Struct Reference

Inheritance diagram for lib::log::BaseHandler:

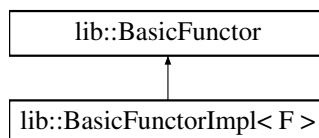


### Public Member Functions

- **BaseHandler** ([Formatter\\_ptr](#) f)
- void **handle** (const [LogRecord](#) &lr)

## 5.12 lib::BasicFunctor Struct Reference

Inheritance diagram for lib::BasicFunctor:



### Public Member Functions

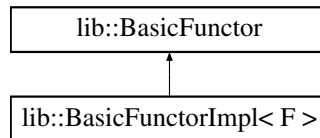
- virtual void **operator()** ()=0

### 5.13 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()** ()

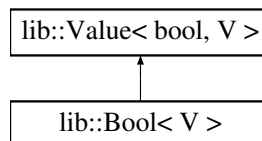
#### 5.13.1 Detailed Description

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

Basic functor encapsulating anything callable that takes no arguments.

### 5.14 lib::Bool< V > Struct Template Reference

Inheritance diagram for lib::Bool< V >:



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

### 5.15 lib::Caar< T > Struct Template Reference

#### Public Member Functions

- typedef **DO** ([Car](#)< DO([Car](#)< T >)>)> Type



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

## 5.16 lib::Cadr< T > Struct Template Reference

### Public Member Functions

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

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

## 5.17 lib::Car< T > Struct Template Reference

### Public Types

- typedef T::Head **Type**

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

## 5.18 lib::Cdar< T > Struct Template Reference

### Public Member Functions

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

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

## 5.19 lib::Cddr< T > Struct Template Reference

### Public Member Functions

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

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

## 5.20 lib::Cdr< T > Struct Template Reference

### Public Types

- typedef T::Tail **Type**

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

## 5.21 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**

## 5.22 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, >
```

## 5.23 lib::TryCall\_apply< T, E, void >::Check< typename, > Struct Template Reference

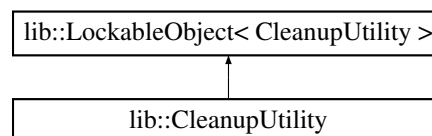
```
template<typename T, typename E>template<typename, void(*) (const E &)> struct lib::TryCall-
_apply< T, E, void >::Check< typename, >
```

## 5.24 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

### 5.24.1 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.

## 5.25 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 >
```

## 5.26 lib::Condition Class Reference

### Public Member Functions

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

## 5.27 hw::Connection Class Reference

### Public Member Functions

- **Connection** (const std::string &, bool)
- void **write** (const void \*d, size\_t)
- [Packet\\_ptr](#) **read** ()
- void **close** ()

## 5.28 lib::qnx::Connection Class Reference

### Public Member Functions

- void **send** (Packet\_ptr) const

### Friends

- class **Channel**

## 5.29 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 >
```

## 5.30 lib::ConsFn< T1, T2 > Struct Template Reference

### Public Types

- typedef [Cons](#)< T1, T2 > **Type**

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

### 5.31 lib::ConstructFSMLineage< T > Struct Template Reference

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

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

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

### 5.33 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 >
```

### 5.34 lib::Contains< Nil, T > Struct Template Reference

#### Static Public Attributes

- static const bool **value** = false

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

### 5.35 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 >
```

### 5.36 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 >
```

### 5.37 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 >
```

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

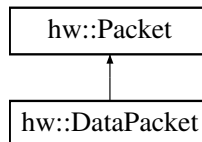
#### Public Types

- 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 >
```

## 5.39 hw::DataPacket Class Reference

Inheritance diagram for hw::DataPacket:



### Public Member Functions

- template<typename T >  
**DataPacket** (const T &v)
- **DataPacket** (const void \*d, uint32\_t s)
- void **set** (void \*, size\_t)
- uint32\_t **size** () const
- const uint8\_t \* **data** () const
- uint8\_t **id** () const
- uint32\_t **hash** () const

### Static Public Member Functions

- static [Packet\\_ptr](#) **assemble** (const uint8\_t \*d, uint32\_t s)

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

### Public Types

- typedef T **Type**

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

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

### Public Types

- typedef [Decay](#)< T >::Type **Type**

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

## 5.42 lib::Decay< const volatile T > Struct Template Reference

### Public Types

- typedef [Decay](#)< T >::Type **Type**

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

## 5.43 lib::Decay< T & > Struct Template Reference

### Public Types

- typedef [Decay](#)< T >::Type **Type**

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

## 5.44 lib::Decay< volatile T > Struct Template Reference

### Public Types

- typedef [Decay](#)< T >::Type **Type**

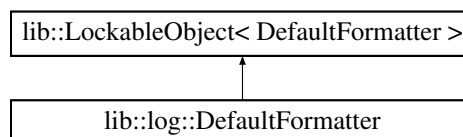
```
template<typename T> struct lib::Decay< volatile T >
```

## 5.45 lib::log::DefaultFormatter Class Reference

Default formatter that lists all information of the passed [LogRecord](#).

```
#include <DefaultFormat.h>
```

Inheritance diagram for lib::log::DefaultFormatter:



### Public Member Functions

- `std::string operator()` (const [LogRecord](#) &)



### Static Public Member Functions

- static std::string **toDate** (uint64\_t)

#### 5.45.1 Detailed Description

Default formatter that lists all information of the passed [LogRecord](#).

It generates string as follows: "**thread-ID** [ **LogLevel**]  
@**filename**:**line** '**message**' "

## 5.46 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

## 5.47 lib::RingBufferConcurrency::MultiThreaded< T >::Empty-Lock Class Reference

### Public Member Functions

- **EmptyLock** ([MultiThreaded](#)< T > \*t)

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

## 5.48 lib::RingBufferConcurrency::SingleThreaded< T >::Empty-Lock Struct Reference

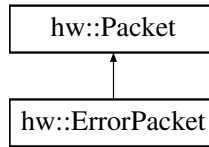
### Public Member Functions

- **EmptyLock** ([SingleThreaded](#)< T > \*)

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

## 5.49 hw::ErrorPacket Class Reference

Inheritance diagram for hw::ErrorPacket:



### Public Member Functions

- uint32\_t **size** () const
- const uint8\_t \* **data** () const
- uint8\_t **id** () const
- uint32\_t **hash** () const

### Static Public Member Functions

- static [Packet\\_ptr](#) **assemble** (const uint8\_t \*d, uint32\_t s)

## 5.50 lib::RingBufferConcurrency::SingleThreaded< T >::FillLock Struct Reference

### Public Member Functions

- **FillLock** ([SingleThreaded](#)< T > \*)

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

## 5.51 lib::RingBufferConcurrency::MultiThreaded< T >::FillLock - Class Reference

### Public Member Functions

- **FillLock** ([MultiThreaded](#)< T > \*t)

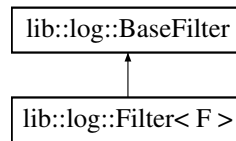
```
template<typename T> class lib::RingBufferConcurrency::MultiThreaded< T >::FillLock
```

## 5.52 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)

#### 5.52.1 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.

## 5.53 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 >

## 5.54 lib::Filter< F, Nil > Struct Template Reference

### Public Types

- typedef [Nil](#) **Type**

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

### 5.55 lib::Flatten< T > Struct Template Reference

#### Public Types

- typedef T **Type**

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

### 5.56 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**

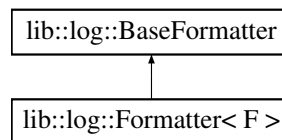
#### Public Member Functions

- typedef **DO** ([Flatten](#)< T >) Rest

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

### 5.57 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 >
```

### 5.58 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)

#### 5.58.1 Detailed Description

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

## 5.59 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
>
```

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

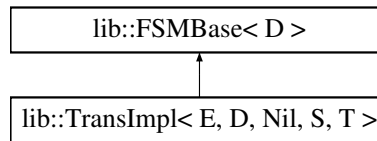
### Public Types

- typedef TryCall\_enter< I, void > **EnterFunction**

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

## 5.61 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)

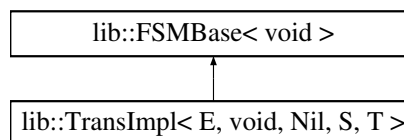
### Public Attributes

- int **state\_**
- D **data\_**

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

## 5.62 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 >
```

## 5.63 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 >
```

## 5.64 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()** ()

### 5.64.1 Detailed Description

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

A functor that calls an object's member function.

## 5.65 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 >
```

## 5.66 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 >
```

## 5.67 lib::GetElem< 0, List > Struct Template Reference

### Public Member Functions

- typedef **DO** ([Car](#)< List >) Type

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

## 5.68 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 >
```

## 5.69 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**

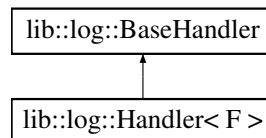
```
template<typename Map, typename Key> struct lib::GetValue< Map, Key >
```

## 5.70 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)

### 5.70.1 Detailed Description

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

[Handler](#) template that holds a functor.

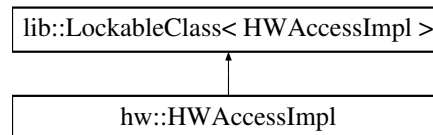
All accepted LogRecords of a [Logger](#) instance are passed to the [Logger](#)'s handlers. There they are run through a formatter; and the formatters output is passed to the functor.

## 5.71 hw::HWAcessImpl Class Reference

Interface for direct hardware access.

```
#include <HWAcess.h>
```

Inheritance diagram for hw::HWAcessImpl:



### Classes

- struct [Lock](#)

### Public Types

- typedef [lib::LockableClass](#) < [HWAcessImpl](#) > **Super**
- typedef [lib::Singleton](#) < [HWAcessImpl](#), [lib::SingletonConcurrency::Multi-Threaded](#) > **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

#### 5.71.1 Detailed Description

Interface for direct hardware access.

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

## 5.72 lib::Identity< T > Struct Template Reference

### Public Types

- typedef T **Type**

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

## 5.73 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 >
```

## 5.74 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 >
```

## 5.75 lib::InheritLineage< T > Struct Template Reference

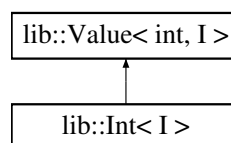
```
template<typename T> struct lib::InheritLineage< T >
```

## 5.76 lib::InheritLineage< Nil > Struct Template Reference

```
template<> struct lib::InheritLineage< Nil >
```

## 5.77 lib::Int< I > Struct Template Reference

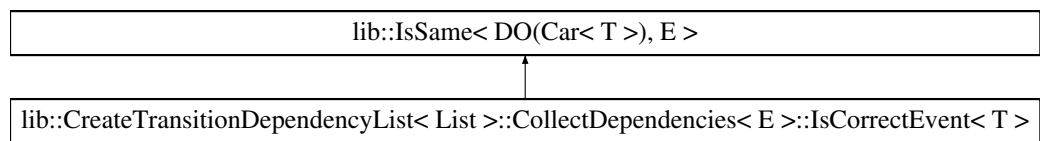
Inheritance diagram for lib::Int< I >:



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

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

Inheritance diagram for 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 >
```

### 5.79 lib::IsList< T > Struct Template Reference

#### Static Public Attributes

- static const bool **value** = false

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

### 5.80 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 > >
```

### 5.81 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 >
```

## 5.82 lib::IsSame< T, T > Struct Template Reference

### Static Public Attributes

- static const bool **value** = true

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

## 5.83 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 >
```

## 5.84 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 >
```

## 5.85 lib::Join< Nil, Appendage > Struct Template Reference

## Public Types

- typedef Appendage **Type**

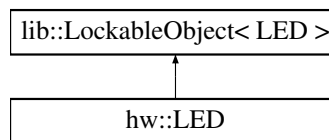
```
template<typename Appendage> struct lib::Join< Nil, Appendage >
```

## 5.86 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**

### 5.86.1 Detailed Description

Allows access to LEDs.

Offers an interface for accessing LEDs on the hw unit. Implements blinking functionality via [lib::Timer](#).

## 5.87 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 >
```

## 5.88 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 >
```

## 5.89 lib::ListToMap< List > Struct Template Reference

### Public Member Functions

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

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

## 5.90 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 >
```

## 5.91 lib::ListToMapImpl< IDX, Nil > Struct Template Reference

### Public Types

- typedef [Nil](#) **Type**

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

## 5.92 lib::SingletonConcurrency::SingleThreaded< T >::Lock - Struct Reference

### Public Member Functions

- **Lock** ([Mutex](#) \*)

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

## 5.93 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
```

## 5.94 hw::HWAAccessImpl::Lock Struct Reference

## 5.95 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 >
```

## 5.96 lib::LockableClass< T, M >::Lock Struct Reference

### Public Member Functions

- **Lock** (T \*)

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

## 5.97 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
```

## 5.98 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 >
```

## 5.99 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 >
```

## 5.100 lib::test::TestManager::Log Struct Reference

## Public Member Functions

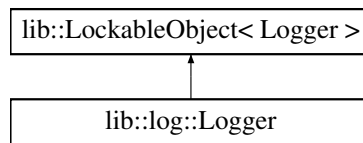
- virtual std::string **read** ()=0
- virtual bool **isEmpty** () const =0

## 5.101 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 >**

### 5.101.1 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.

## 5.102 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

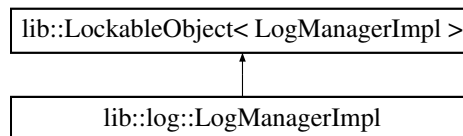
- static const [LogLevel](#) **INFO**
- static const [LogLevel](#) **WARNING**
- static const [LogLevel](#) **ERROR**
- static const [LogLevel](#) **CRITICAL**

## 5.103 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 &)

#### 5.103.1 Detailed Description

LogManager [Singleton](#), grants access to [Logger](#) instances.

Creates and exposes [Logger](#) instances by alphanumerical id.

Offers a "root log" for convenience.

## 5.104 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

## 5.105 lib::MakeList\_0 Struct Reference

## Public Types

- typedef [Nil](#) **Type**

## 5.106 lib::MakeList\_1&lt; A &gt; Struct Template Reference

## Public Types

- typedef [Cons](#)< A, MAKELIST\_0 > **Type**

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

## 5.107 lib::MakeList\_2&lt; A, B &gt; Struct Template Reference

## Public Types

- typedef [Cons](#)< A, MAKELIST\_1(B)> **Type**

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

## 5.108 lib::MakeList\_3&lt; A, B, C &gt; 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 >
```

### 5.109 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 >
```

### 5.110 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 >
```

### 5.111 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 >
```

### 5.112 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 >
```

## 5.113 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 >
```

## 5.114 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 >
```

## 5.115 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 >
```

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

### Public Types

- typedef T **Type**

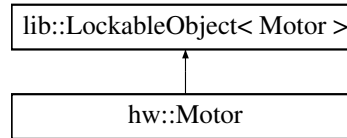
```
template<template< typename, typename > class F, typename T> struct lib::Merge< F, T, Nil >
```

## 5.117 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**

### 5.117.1 Detailed Description

Client interface for controlling the conveyor belt and electromagnetic switch.



## 5.117.2 Member Function Documentation

### 5.117.2.1 void hw::Motor::controlBelt ( pid\_t *dir*, pid\_t *speed* )

Controls conveyor belt.

#### Parameters

<i>dir</i>	<a href="#">Direction</a> the conveyor belt is supposed to move in.
<i>speed</i>	<a href="#">Speed</a> of the conveyor belt.

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

### 5.117.2.2 void hw::Motor::controlSwitch ( pid\_t *state* )

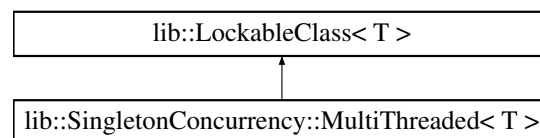
Controls electromagnetic switch.

#### Parameters

<i>state</i>	<a href="#">State</a> the switch is supposed to be in.
--------------	--

## 5.118 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 >
```

## 5.119 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 >
```

## 5.120 lib::Mutex Class Reference

### Public Member Functions

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

## 5.121 lib::Nil Struct Reference

### Public Types

- typedef [Nil](#) **Type**

## 5.122 lib::IsSuperType< Sub, Super >::No Struct Reference

### Public Attributes

- char **v** [2]

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

## 5.123 lib::Not< T > Struct Template Reference

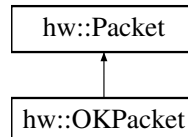
### Static Public Attributes

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

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

## 5.124 hw::OKPacket Class Reference

Inheritance diagram for hw::OKPacket:



### Public Member Functions

- **OKPacket** (uint32\_t s)
- uint32\_t **size** () const
- const uint8\_t \* **data** () const
- uint8\_t **id** () const
- uint32\_t **hash** () const
- uint32\_t **status** () const

### Static Public Member Functions

- static [Packet\\_ptr](#) **assemble** (const uint8\_t \*d, uint32\_t s)

## 5.125 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 >
```

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

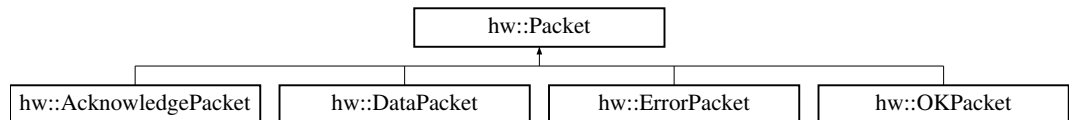
### Public Types

- typedef [Or](#)< T1, T2 > **Type**

```
template<typename List>template<typename T1, typename T2> struct lib::ListOr< List >::Or-
Fn< T1, T2 >
```

## 5.127 hw::Packet Class Reference

Inheritance diagram for hw::Packet:



### Public Member Functions

- virtual uint32\_t **size** () const =0
- virtual const uint8\_t \* **data** () const =0
- virtual uint8\_t **id** () const =0
- virtual uint32\_t **hash** () const =0

### Static Public Attributes

- static const uint8\_t **DATA\_ID** = 0
- static const uint8\_t **OK\_ID** = 1
- static const uint8\_t **ERROR\_ID** = 2
- static const uint8\_t **ACKNOWLEDGE\_ID** = 3

## 5.128 lib::qnx::Receiver Class Reference

### Public Member Functions

- Packet\_ptr **receive** () const

### Friends

- class **Channel**

## 5.129 lib::test::TestManager::Registrar Struct Reference

### Public Member Functions

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

## 5.130 lib::Reverse< List > Struct Template Reference

### Public Member Functions

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

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

## 5.131 lib::ReverseCons< Cell > Struct Template Reference

### Public Types

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

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

## 5.132 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 >
```

## 5.133 lib::ReverseImpl< Done, Nil > Struct Template Reference

### Public Types

- typedef Done **Type**

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

## 5.134 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

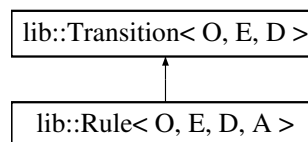
### Static Public Attributes

- 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 >
```

## 5.135 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
>
```

## 5.136 lib::test::TestManager::Selector Struct Reference

### Public Member Functions

- **Selector** (const std::string &unit\_id)

## 5.137 lib::Semaphore Class Reference

### Public Member Functions

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

## 5.138 lib::Setify< List > Struct Template Reference

### Public Member Functions

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

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

## 5.139 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 >
```

## 5.140 lib::SetifyImpl< Done, Nil > Struct Template Reference

### Public Member Functions

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

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

## 5.141 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 >
```

## 5.142 lib::SingletonConcurrency::SingleThreaded< T > Struct - Template Reference

## Classes

- struct [Lock](#)

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

### 5.143 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.*

#### 5.143.1 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.

#### 5.143.2 Member Function Documentation

5.143.2.1 `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.

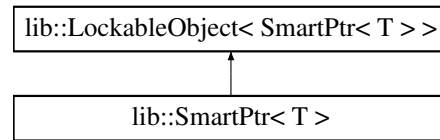
### 5.144 lib::SmartPtr< T > Class Template Reference

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

#### 5.144.1 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.

## 5.145 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

## 5.146 hw::Motor::State Struct Reference

### Static Public Attributes

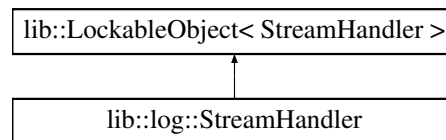
- static const pid\_t **OPEN** = 0
- static const pid\_t **CLOSE** = 1

## 5.147 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 &)

#### 5.147.1 Detailed Description

[Handler](#) compatible functor that writes its [LogRecord](#) to an std::stream instance.

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

## 5.148 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** ()

## 5.149 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  
*Whether 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 \*)

### 5.149.1 Detailed Description

Encapsulates the most important features of a thread.

### 5.149.2 Constructor & Destructor Documentation

#### 5.149.2.1 `lib::Thread::Thread ( void )`

Default constructor.

Initializes inert [Thread](#)

#### 5.149.2.2 `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.

#### 5.149.2.3 `lib::Thread::~~Thread ( void )`

Destructor.

##### Warning

terminates if this [Thread](#) is still joinable

### 5.149.3 Member Function Documentation

#### 5.149.3.1 `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

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

Whether or not the [Thread](#) is joinable.

##### Warning

[Thread](#) cannot be destroyed while joinable

## 5.149.3.3 Thread &amp; lib::Thread::operator= ( Thread &amp; t )

Assignment operator.

Moves content to this [Thread](#) object.

**Warning**

terminates if this [Thread](#) is already joinable

## 5.149.3.4 void lib::Thread::run ( void ) [protected]

This is called from the new [Thread](#).

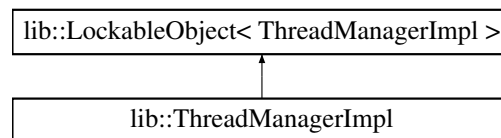
It executes the user's functor.

**Warning**

terminates if functor throws an exception

## 5.150 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** ()

## 5.151 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

### 5.151.1 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.

## 5.152 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.*

### 5.152.1 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.

### 5.152.2 Member Function Documentation

#### 5.152.2.1 bool lib::Timer::active ( void ) const

Is timer currently waiting for ftor execution.

#### 5.152.2.2 void lib::Timer::deactivateAll ( void ) [static]

Deactivate all timers.

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

#### 5.152.2.3 Time lib::Timer::delta ( void )

Amount of time elapsed since last reset.

Resets timer.

#### 5.152.2.4 Time lib::Timer::elapsed ( void )

Amount of time elapsed since last reset.

Doesn't reset timer.

#### 5.152.2.5 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 (`{1}{t}$f`)

#### 5.152.2.6 Timer::ts\_t lib::Timer::timestamp ( void ) [static]

Returns current system time in nanoseconds since Jan.

1st 1970.

## 5.153 lib::TimerPoolImpl Class Reference

### Public Types

- typedef [Singleton](#)< [TimerPoolImpl](#) > **SingletonInst**

### Friends

- class **Timer**



## 5.154 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 >
```

## 5.155 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)

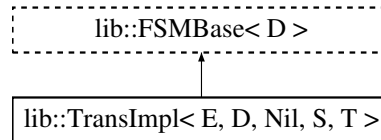
### Static Public Attributes

- 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 >
```

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

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



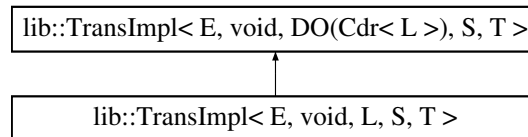
#### Public Member Functions

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

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

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

Inheritance diagram for 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)

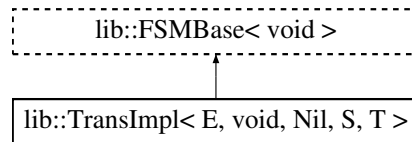
**Static Public Attributes**

- 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 >
```

## 5.158 lib::TransImpl< E, void, Nil, S, T > Struct Template - Reference

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

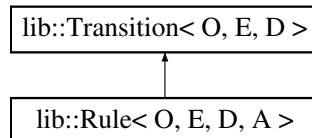
**Public Member Functions**

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

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

## 5.159 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 >
```

## 5.160 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 >
```

## 5.161 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 >
```

## 5.162 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)

## 5.163 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 >
```

## 5.164 lib::ValueIdentity< Bool< I > > Struct Template Reference

### Static Public Attributes

- static const int **value** = I

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

## 5.165 lib::ValueIdentity< Int< I > > Struct Template Reference

### Static Public Attributes

- static const int **value** = I

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

## 5.166 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 > >
```

## 5.167 lib::IsSuperType< Sub, Super >::Yes Struct Reference

### Public Attributes

- char **v** [1]

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