TI4_SE2

Generated by Doxygen 1.7.6.1

Wed Apr 27 2016 09:41:16

Contents

1	Nam	espace Index	1
	1.1	Namespace List	1
2	Clas	s Index	3
	2.1	Class Hierarchy	3
3	Clas	s Index	9
	3.1	Class List	9
4	Nam	espace Documentation	15
	4.1	lib::SingletonConcurrency Namespace Reference	15
		4.1.1 Detailed Description	15
5	Clas	s Documentation	17
	5.1	hw::Actuator Class Reference	17
		5.1.1 Detailed Description	18
	5.2	lib::And $<$ T1, T2 $>$ Struct Template Reference	18
	5.3	$\label{lib::ListAnd} \mbox{List} > \mbox{::AndFn} < \mbox{T1, T2} > \mbox{Struct Template Reference} . . .$	18
	5.4	$\label{eq:lib::Apply} \mbox{lib::Apply} < \mbox{F, T} > \mbox{Struct Template Reference} \qquad . \qquad . \qquad . \qquad . \qquad .$	18
	5.5	$\label{eq:lib::Apply} \mbox{lib::Apply} < \mbox{F, Nil} > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	18
	5.6	lib::Array< T, N > Class Template Reference	19
	5.7	lib::Array< T, 0 > Class Template Reference	20
	5.8	lib::log::BaseFilter Struct Reference	20
	5.9	lib::log::BaseFormatter Struct Reference	20
	5.10	lib::log::BaseHandler Struct Reference	20
	5.11	lib::BasicFunctor Struct Reference	21
	5.12	lib::BasicFunctorImpl $<$ F $>$ Class Template Reference	21

ii CONTENTS

	5.12.1 Detailed Description	22
5.13	$\label{eq:lib::Bool} \mbox{lib::Bool} < \mbox{V} > \mbox{Struct Template Reference} $	22
5.14	$\label{eq:car_def} \mbox{lib::Caar} < T > \mbox{Struct Template Reference} \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ \ . \ \ \ . \ \ \ \ \ \ \ \ \ \ \ \ \$	22
5.15	$\label{eq:cadr} \mbox{lib::Cadr} < T > \mbox{Struct Template Reference} \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ \ . \ \ . \ \ . \ \ . \ \ . \ \ \ . \ \ \ . \ \ \ \ \ \ \ \ \ \ \ \ \$	22
5.16	$\label{eq:car_def} \mbox{lib::Car} < T > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	22
5.17	$\label{eq:continuous_continuous_continuous} \mbox{lib::Cdar} < T > \mbox{Struct Template Reference} $	23
5.18	$\label{eq:continuous_continuous_continuous} \mbox{lib::Cddr} < T > \mbox{Struct Template Reference} $	23
5.19	lib::Cdr< T $>$ Struct Template Reference	23
5.20	lib::qnx::Channel Class Reference	23
5.21	$\label{lib::TryCall_apply} \begin{subarray}{ll} \$	24
5.22	lib::TryCall_apply< T, E, void >::Check< typename, > Struct Template Reference	24
5.23	lib::CleanupUtility Class Reference	24
	5.23.1 Detailed Description	25
5.24	lib::CreateTransitionDependencyList< List >::CollectDependencies< E > Struct Template Reference	25
5.25	lib::Condition Class Reference	25
5.26	hw::Connection Class Reference	25
5.27	lib::qnx::Connection Class Reference	26
5.28	$\label{lib::Cons} \mbox{Lib::Cons} < \mbox{H, T} > \mbox{Struct Template Reference} $	26
5.29	lib::ConsFn< T1, T2 > Struct Template Reference	26
5.30	$\label{eq:constructFSML} \mbox{lib::ConstructFSMLineage} < T > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	27
5.31	${\it lib::} {\it ConstructFSMLineage} {< cons} {< t, Nil} > {> struct Template Reference}$	27
5.32	$\label{eq:lib::Contains} \mbox{List, T} > \mbox{Struct Template Reference} $	27
5.33	lib::Contains < Nil, T > Struct Template Reference	27
5.34	$\label{lib::CreateStateList} \mbox{List} > \mbox{Struct Template Reference} $	27
5.35	${\it lib::} Create Transition Dependency List < List > Struct\ Template\ Reference$	28
5.36	$\label{eq:continuous_lib::CreateTransitionMap} \textbf{List} > \textbf{Struct Template Reference} $	28
5.37	$\label{lib::FSMMaker} \begin{subarray}{ll} lib::FSMMaker < I, D, T > :: Create Transition Tree < TT > Struct Template \\ Reference & $	28
5.38	lib::Data Class Reference	29
5.39	$\label{eq:lib::Decay} \mbox{lib::Decay} < \mbox{T} > \mbox{Struct Template Reference} \qquad . \qquad . \qquad . \qquad . \qquad . \qquad .$	29
5.40	$\label{eq:lib::Decay} \mbox{const T} > \mbox{Struct Template Reference} \ \ . \ \ \ \ \ . \ \ . \ \ \ . \ \ \ . \ \ \ . \ \ \ \ \ \ \ \ \ \ \ \ \ \ $	29
5.41	lib::Decay< const volatile T $>$ Struct Template Reference	29

CONTENTS iii

F 40	Blow Decree & T. 0. S. Otmost Townslate D. C.	00
	lib::Decay< T & > Struct Template Reference	30
	lib::Decay< volatile T > Struct Template Reference	30
5.44	lib::log::DefaultFormatter Class Reference	30
	5.44.1 Detailed Description	31
	hw::Motor::Direction Struct Reference	31
5.46	lib::RingBufferConcurrency::MultiThreaded< T >::EmptyLock Class - Reference	31
5.47	$\label{lib::RingBufferConcurrency::SingleThreaded} I ib::RingBufferConcurrency::SingleThreaded < T > ::EmptyLock Struct - Reference$	31
5.48	$\label{lib::RingBufferConcurrency::MultiThreaded} \begin{tabular}{ll} T >::FillLock & Class & -Reference & $	31
5.49	$\label{lib::RingBufferConcurrency::SingleThreaded} \begin{tabular}{ll} $T > ::FillLock & Struct - \\ Reference & $	32
5.50	$\label{eq:lib::log::Filter} \mbox{lib::log::Filter} < \mbox{F} > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	32
	5.50.1 Detailed Description	32
5.51	$\label{eq:lib::Filter} \mbox{\sc F, List} > \mbox{\sc Struct Template Reference} \; . \; . \; . \; . \; . \; . \; . \; . \; . \; $	32
5.52	$\label{eq:lib::Filter} \mbox{\sc F, Nil} > \mbox{\sc Struct Template Reference} $	33
5.53	$\label{eq:lib::Flatten} \mbox{lib::Flatten} < T > \mbox{Struct Template Reference} \ \ . \ \ \ . \ \ \ . \ \ . \ \ \ \ \ . \ \ \ . \$	33
5.54	lib::Flatten < Cons < H, T > > Struct Template Reference	33
5.55	$\label{lib::log::Formatter} \mbox{lib::log::Formatter} < \mbox{F} > \mbox{Struct Template Reference} $	34
5.56	lib::Frequency Class Reference	34
	5.56.1 Detailed Description	34
5.57	$\label{eq:lib::FSM} \mbox{ID, I, D, Lineage} > \mbox{Struct Template Reference} $	34
5.58	lib::FSM< ID, I, void, Lineage $>$ Struct Template Reference	35
5.59	$lib:: FSMBase < D > Struct \ Template \ Reference \ \ . \ \ . \ \ . \ \ . \ \ . \ \ .$	35
5.60	$\label{eq:lib::FSMBase} \mbox{lib::FSMBase} < \mbox{void} > \mbox{Struct Template Reference} \ \ . \ \ \ . \ \ \ . \ \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ \$	35
5.61	$\label{eq:lib::FSMMaker} \mbox{lib::FSMMaker} < \mbox{I, D, T} > \mbox{Struct Template Reference} $	36
5.62	$\label{eq:lib::FtorWrapper} \mbox{lib::FtorWrapper} < T > \mbox{Class Template Reference} $	37
	5.62.1 Detailed Description	37
5.63	$\label{lib::CreateTransitionDependencyList} \begin{subarray}{ll} {\tt lib::CreateTransitionDependencyList} < {\tt List} > :: {\tt GetDependency} < {\tt T} > - \\ {\tt Struct Template Reference} & \ldots & \ldots & \ldots & \ldots & \ldots \\ \hline \end{subarray}$	37
5.64	$\label{eq:lib::GetElem} \mbox{lib::GetElem} < \mbox{IDX, List} > \mbox{Struct Template Reference} $	37
5.65	$\label{eq:lib::GetElem} \mbox{lib::GetElem} < 0, \mbox{List} > \mbox{Struct Template Reference} \qquad . \qquad . \qquad . \qquad .$	38
5.66	$\label{lib::CreateStateList} \begin{tabular}{lllllllllllllllllllllllllllllllllll$	38
5.67	lib::GetValue< Map, Key > Struct Template Reference	38

iv CONTENTS

5.68	$\label{eq:lib::log::Handler} \mbox{lib::log::Handler} < \mbox{F} > \mbox{Class Template Reference} \qquad . \qquad . \qquad . \qquad . \qquad .$	38
	5.68.1 Detailed Description	39
5.69	hw::HWAccessImpl Class Reference	39
	5.69.1 Detailed Description	40
5.70	$\label{eq:lib::ldentity} \mbox{lib::ldentity} < T > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	40
5.71	lib::If < false, T1, T2 > Struct Template Reference	40
5.72	$\label{eq:lib::If} \mbox{lib::If} < \mbox{true, T1, T2} > \mbox{Struct Template Reference} \qquad . \qquad . \qquad . \qquad . \\$	40
5.73	hw::Connection::Impl Class Reference	40
5.74	$\label{eq:lib::InheritLineage} \mbox{lib::InheritLineage} < \mbox{T} > \mbox{Struct Template Reference} \ \ . \ \ \ . \ \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \$	41
5.75	$\label{eq:lib::InheritLineage} \mbox{\sc Nil} > \mbox{\sc Struct Template Reference} $	41
5.76	$\label{eq:lib::Int} \mbox{lib::Int} < \mbox{I} > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	41
5.77	$\label{lib::CreateTransitionDependencyList} \begin{tabular}{lllllllllllllllllllllllllllllllllll$	41
5.78	$\label{eq:lib::lsList} \mbox{lib::lsList} < T > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	42
5.79	lib::IsList< Cons< T1, T2 >> Struct Template Reference	42
5.80	lib::IsSame $<$ T1, T2 $>$ Struct Template Reference	42
5.81	$\label{eq:lib::lsSame} \mbox{lib::lsSame} < \mbox{T, T} > \mbox{Struct Template Reference} $	42
5.82	$\label{eq:lib::lsSuperType} \mbox{Sub, Super} > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ .$	42
5.83	lib::Join< List, Appendage > Struct Template Reference	43
5.84	$\label{eq:lib::Join} \mbox{lib::Join} < \mbox{Nil, Appendage} > \mbox{Struct Template Reference} \ \ . \ \ \ . \ \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ \ . \ \ \ \ . \ \ \ . \ \ . \$	43
5.85	hw::LED Class Reference	43
	5.85.1 Detailed Description	44
5.86	$\label{eq:lib::ListAnd} \mbox{List} > \mbox{Struct Template Reference} $	44
5.87	$\label{eq:lib::ListOr} \mbox{List} > \mbox{Struct Template Reference} $	45
5.88	$\label{eq:lib::ListToMap} \mbox{List} > \mbox{Struct Template Reference} \ \ . \ \ \ \ . \ \ \ \ \ . \$	45
5.89	$\label{eq:lib::ListToMapImpl} \textit{LibX}, \textit{List} > \textit{Struct Template Reference} $	45
5.90	$\label{eq:lib::ListToMapImpl} \textit{IDX}, \textit{Nil} > \textit{Struct Template Reference} \dots \dots \dots$	46
5.91	$\label{lib::SingletonConcurrency::SingleThreaded} $$ T>::Lock Struct Reference $$ T>::Lock Struct Ref$	46
5.92	${\it lib::} Singleton Concurrency:: MultiThreaded < T > :: Lock Struct \ Reference$	46
5.93	hw::HWAccessImpl::Lock Struct Reference	46
5.94	$\label{eq:lib::Lock} \mbox{lib::Lock} < \mbox{T, E, R} > \mbox{Class Template Reference} \ldots \ldots $	46
5.95	$\label{lib::LockableClass} \mbox{\sc T, M} > :: \mbox{\sc Lock Struct Reference} $	47
5.96	$\label{lib::LockableObject} \mbox{lib::Lock Struct Reference} \ \ldots \ \ldots \ \ldots$	47
5.97	$\label{lib::LockableClass} \mbox{Lib::LockableClass} < \mbox{T, M} > \mbox{Class Template Reference} $	47

CONTENTS

5.98 lib::LockableObject< T, M > Class Template Reference	17
	18
	18
	19
	19
	19
5.102.1 Detailed Description	50
5.103lib::log::LogRecord Class Reference	50
5.104lib::MakeList_0 Struct Reference	50
5.105lib::MakeList_1 < A > Struct Template Reference	50
5.106lib::MakeList_2< A, B > Struct Template Reference	51
5.107lib::MakeList_3< A, B, C > Struct Template Reference 5	51
5.108lib::MakeList_4< A, B, C, D > Struct Template Reference 5	51
5.109lib::MakeList_5< A, B, C, D, E $>$ Struct Template Reference 5	51
5.110lib::MakeList_6< A, B, C, D, E, F > Struct Template Reference 5	52
5.111 lib::MakeList_7< A, B, C, D, E, F, G $>$ Struct Template Reference 5	52
$5.112 lib:: Make List_8 < A,B,C,D,E,F,G,H > StructTemplateReference . . 5.112 lib:: Make List_8 < A,B,C,D,E,F,G,H > StructTemplateReference $	52
$5.113 lib:: Make List_9 < A,B,C,D,E,F,G,H,I > Struct Template Reference . 5.113 lib:: Make List_9 < A,B,C,D,E,F,G,H,I > Struct Template Reference . 5.113 lib:: Make List_9 < A,B,C,D,E,F,G,H,I > Struct Template Reference . 5.113 lib:: Make List_9 < A,B,C,D,E,F,G,H,I > Struct Template Reference . 5.113 lib:: Make List_9 < A,B,C,D,E,F,G,H,I > Struct Template Reference . 5.113 lib:: Make List_9 < A,B,C,D,E,F,G,H,I > Struct Template Reference . 5.113 lib:: Make List_9 < A,B,C,D,E,F,G,H,D < A,D,D < A,D $	52
5.114lib::Merge < F, T, List > Struct Template Reference	53
5.115lib::Merge < F, T, Nil > Struct Template Reference	53
5.116hw::Motor Class Reference	53
5.116.1 Detailed Description	54
5.116.2 Member Function Documentation	54
5.116.2.1 controlBelt	54
5.116.2.2 controlSwitch	54
5.117lib::SingletonConcurrency::MultiThreaded< T > Struct Template - Reference	55
5.118lib::RingBufferConcurrency::MultiThreaded< T > Class Template - Reference	55
5.119lib::Mutex Class Reference	55
5.120lib::Nil Struct Reference	6
5.121lib::IsSuperType< Sub, Super >::No Struct Reference	6
5.122lib::Not $<$ T $>$ Struct Template Reference	6
5.123lib::Or < T1, T2 > Struct Template Reference	6

vi CONTENTS

5.124lib::ListOr< List >::OrFn< T1, T2 > Struct Template Reference	56
5.125lib::qnx::Receiver Class Reference	57
5.126lib::test::TestManager::Registrar Struct Reference	57
5.127lib::Reverse< List > Struct Template Reference	57
5.128lib::ReverseCons< Cell > Struct Template Reference	57
5.129lib::ReverseImpl< Done, ToDo > Struct Template Reference	57
5.130lib::ReverseImpl< Done, Nil $>$ Struct Template Reference	58
5.131 lib::RingBuffer $<$ T, N, ThreadingPolicy $>$ Class Template Reference	58
5.132lib::Rule < O, E, D, A > Struct Template Reference	58
5.133lib::test::TestManager::Selector Struct Reference	59
5.134lib::Semaphore Class Reference	59
5.135lib::Setify< List > Struct Template Reference	59
5.136lib::SetifyImpl< Done, ToDo > Struct Template Reference	59
5.137lib::SetifyImpl< Done, Nil > Struct Template Reference	59
5.138lib::SingletonConcurrency::SingleThreaded< T > Struct Template - Reference	60
5.139lib::RingBufferConcurrency::SingleThreaded< T > Class Template - Reference	60
5.140lib::Singleton< T, TM, P > Class Template Reference	60
5.140.1 Detailed Description	60
5.140.2 Member Function Documentation	61
5.140.2.1 instance	61
5.141 lib::SmartPtr< T > Class Template Reference	61
5.141.1 Detailed Description	62
5.142hw::Motor::Speed Struct Reference	62
5.143hw::Motor::State Struct Reference	62
5.144lib::log::StreamHandler Class Reference	62
5.144.1 Detailed Description	63
5.145lib::test::TestManager Class Reference	63
5.146lib::Thread Class Reference	
5.146.1 Detailed Description	64
5.146.2 Constructor & Destructor Documentation	64
5.146.2.1 Thread	64
5.146.2.2 Thread	64

CONTENTS vii

5.146.2.3 ~Thread	65
5.146.3 Member Function Documentation	65
5.146.3.1 join	65
5.146.3.2 joinable	65
5.146.3.3 operator=	65
5.146.3.4 run	65
5.147lib::ThreadManagerImpl Class Reference	66
5.148lib::Time Class Reference	66
5.148.1 Detailed Description	67
5.149lib::Timer Class Reference	67
5.149.1 Detailed Description	68
5.149.2 Member Function Documentation	68
5.149.2.1 active	68
5.149.2.2 deactivateAll	68
5.149.2.3 delta	68
5.149.2.4 elapsed	69
5.149.2.5 sync	69
5.149.2.6 timestamp	69
5.150lib::TimerPoolImpl Class Reference	69
5.151lib::CreateTransitionMap< List >::Transform< T > Struct Template - Reference	69
5.152lib::TransImpl< E, D, L, S, T > Struct Template Reference	69
5.153lib::TransImpl< E, D, Nil, S, T > Struct Template Reference	70
5.154lib::TransImpl< E, void, L, S, T > Struct Template Reference	71
5.155lib::TransImpl< E, void, Nil, S, T > Struct Template Reference	72
5.156lib::Transition < O, E, D > Struct Template Reference	72
5.157lib::TryCall_apply< T, E, D > Struct Template Reference	72
5.158lib::TryCall_apply< T, E, void > Struct Template Reference	73
5.159lib::test::UnitTest Class Reference	73
5.160lib::Value< T, I > Struct Template Reference	73
5.161 lib::ValueIdentity $<$ Bool $<$ I $>$ $>$ Struct Template Reference	74
5.162lib::ValueIdentity $<$ Int $<$ I $>>$ Struct Template Reference	74
5.163lib::ValueIdentity $<$ Value $<$ T, I $>>$ Struct Template Reference	74
5.164lib::lsSuperType< Sub, Super >::Yes Struct Reference	74

Chapter 1

Namespace Index

1.1 Name	space	List
----------	-------	------

Here is a list of all documented namespaces with brief descriptions:	
lib::SingletonConcurrency	
Contains threading models of the Singleton template	1.9

Chapter 2

Class Index

2.1 Class Hierarchy

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

lib::And< T1, T2 >
$\label{lib::ListAnd} \mbox{List} > \mbox{::AndFn} < \mbox{T1}, \mbox{T2} > \dots $
lib::Apply $\langle F, T \rangle$
lib::Apply< F, Nil >
lib::Array $<$ T, N $>$
lib::Array $<$ T, 0 $>$
lib::log::BaseFilter
lib::log::Filter< F >
lib::log::BaseFormatter
lib::log::Formatter $<$ F $>$
lib::log::BaseHandler
lib::log::Handler< F >
lib::BasicFunctor
lib::BasicFunctorImpl< F >
lib::Caar< T >
lib::Cadr< T >
lib::Car< T >
lib::Cdar $<$ T $>$
lib::Cddr< T >
lib::Cdr< T >
lib::qnx::Channel
lib::TryCall_apply< T, E, D >::Check< typename, >
lib::TryCall_apply< T, E, void >::Check< typename, >
$\label{lib::CreateTransitionDependencyList} \mbox{List} > \mbox{::CollectDependencies} < E > \ \ . \ \ . \ \ \ \ \ \ \ \ \ \ \ \ \$
lib::Condition
lib::qnx::Connection
lib::Cons< H, T >
lih··ConeFn ∕ T1 T2 ∖

4 Class Index

Ibis:Contains< List, T >	lib::ConstructFSMLineage< T >	27
ibi::Contains < Nil, T > 27 ibi::CoreateStateList < List > 27 ibi::CoreateTransitionDependencyList < List > 28 ibi::CoreateTransitionMap < List > 28 ibi::FSMMaker < I, D, T > ::CreateTransitionTree < TT > 28 ibi::Decay < T > 29 ibi::Decay < T > 29 ibi::Decay < const volatile T > 29 ibi::Decay < T & > 30 ibi::Decay < T & > 30 ibi::Decay < T & > 30 ibi::Decay < V & 30	lib::ConstructFSMLineage < Cons < T, Nil > >	27
ibi::Contains < Nil, T > 27 ibi::CoreateStateList < List > 27 ibi::CoreateTransitionDependencyList < List > 28 ibi::CoreateTransitionMap < List > 28 ibi::FSMMaker < I, D, T > ::CreateTransitionTree < TT > 28 ibi::Decay < T > 29 ibi::Decay < T > 29 ibi::Decay < const volatile T > 29 ibi::Decay < T & > 30 ibi::Decay < T & > 30 ibi::Decay < T & > 30 ibi::Decay < V & 30	lib::Contains $<$ List, T $>$	27
ibi::CreateTransitionDependencyList < List >		
Ibi::CreateTransitionDependencyList < List >		
ibi::CreateTransitionMap< List > 28 ibi::FSMMaker < , D, T >::CreateTransitionTree< TT > 28 ibi::Decay< T > 29 ibi::Decay < T > 29 ibi::Decay < const T > 29 ibi::Decay < const volatile T > 29 ibi::Decay < volatile T > 29 ibi::Decay < volatile T > 30 ibi::BingBufferConcurrency::MultiThreaded < T >::EmptyLock 31 ibi::RingBufferConcurrency::SingleThreaded < T >::EmptyLock 31 ibi::RingBufferConcurrency::SingleThreaded < T >::FillLock 31 ibi::RingBufferConcurrency::SingleThreaded < T >::FillLock 32 ibi::Filter < F, List > 32 ibi::Filter < F, List > 33 ibi::Flatten < T > 33 ibi::Flatten < T > 33 ibi::Flatten < Cons < H, T > 33 ibi::Flatten < Cons < H, T > 33 ibi::FsW < ID, I, D, Lineage > 34 ibi::FSM < ID, I, void, Lineage > 34 ibi::FSM < ID, I, void, Lineage > 35 ibi::TransImpl < E, D, Nil, S, T > 70 ibi::FSMBase < void > 35 ibi::TransImpl < E, void, Nil, S, T > 72 ibi::FsMMaker < I, D, T > 36 ibi::GetElem < O, List > 37 ibi::GetElem < O, List > 38 ibi::GetIentily < T > 40 ibi::If calse, T1, T2 > 40 ibi::IhheritLineage < Ni > 41 ibi::IsheritLineage < Ni > 41 ibi::IsheritLineage < Ni > 42 ibi::Ishame		28
Ibi::FSMMaker < I, D, T >::CreateTransitionTree < TT >		28
Ibi::Decay < T >		28
Ibi::Decay < T > 29 Ibi::Decay < const T > 29 Ibi::Decay < const T > 29 Ibi::Decay < const volatile T > 29 Ibi::Decay < T & > 30 Ibi::Decay < T & > 30 Ibi::Decay < volatile T > 30 30 Ibi::Decay < volatile T > 30 30 Ibi::Decay < volatile T > 30 30 Ibi::RingBufferConcurrency::MultiThreaded < T >::EmptyLock 31 Ibi::RingBufferConcurrency::MultiThreaded < T >::EmptyLock 31 Ibi::RingBufferConcurrency::MultiThreaded < T >::FillLock 31 Ibi::RingBufferConcurrency::MultiThreaded < T >::FillLock 32 Ibi::RingBufferConcurrency::SingleThreaded < T >::FillLock 32 Ibi::RingBufferConcurrency::SingleThreaded < T >::FillLock 32 Ibi::Filter < F, Nil > 33 Ibi::Fequency 34 Ibi::Filter < F, Nil > 33 Ibi::Frequency 34 Ibi::FSM < ID, I, D, Lineage > 35 Ibi::FSM < ID, I, void, Lineage > 35 Ibi::FSMBase < D > 35 Ibi::FSMBase < D > 35 Ibi::FSMBase < D > 35 Ibi::TransImpl < E, void, Nil, S, T > 70 Ibi::FSMBase < ID, T > 36 Ibi::TransImpl < E, void, Nil, S, T > 72 Ibi::FSMMaker < I, D, T > 36 Ibi::TransImpl < E, void, Nil, S, T > 72 Ibi::FSMMaker < I, D, T > 36 Ibi::TransImpl < E, void, Nil, S, T > 72 Ibi::FSMMaker < I, D, T > 36 Ibi::TransImpl < E, void, Nil, S, T > 72 Ibi::FSMMaker < I, D, T > 36 Ibi::GetElem < IDX, List > 37 Ibi::GetElem < IDX, List > 37 Ibi::GetElem < IDX, List > 38 Ibi::GetValue < Map, Key > 38 Ibi::Ibi::Isi < 40 Ibi::Isi < T > 42 Ibi::Isi < T		29
Ibi::Decay < const T > 29 Ibi::Decay < const volatile T > 29 Ibi::Decay < const volatile T > 30 30 Ibi::Decay < volatile T > 30 Ibi::Decay < volatile T > 30 Ibi::RingBufferConcurrency::MultiThreaded < T >::EmptyLock 31 Ibi::RingBufferConcurrency::SingleThreaded < T >::EmptyLock 31 Ibi::RingBufferConcurrency::MultiThreaded < T >::EmptyLock 31 Ibi::RingBufferConcurrency::MultiThreaded < T >::EmptyLock 31 Ibi::RingBufferConcurrency::SingleThreaded < T >::FillLock 32 Ibi::Filter < F, List > 32 Ibi::Filter < F, Nil > 33 Ibi::Flatten < T > 34 Ibi::FSM < ID, I, D, Lineage > 34 Ibi::FSM < ID, I, Void, Lineage > 35 Ibi::FSMBase < D > 35 Ibi::FSMBase < D > 35 Ibi::TransImpl < E, D, Nil, S, T > 70 Ibi::FSMBase < D > 35 Ibi::TransImpl < E, void, Nil, S, T > 72 Ibi::FSMMaker < I, D, T > 36 Ibi::TransImpl < E, void, Nil, S, T > 72 Ibi::GetElem < IDX, List > 37 Ibi::GetElem < IDX, List > 37 Ibi::GetElem < D, List > 38 Ibi::GetElem < D, List >		29
Ibib:Decay < const volatile T >		29
Iib::Decay	•	_
Iib::Decay		_
hw::Motor::Direction 31 lib::RingBufferConcurrency::MultiThreaded		
Iib::RingBufferConcurrency::MultiThreaded		
Iib::RingBufferConcurrency::SingleThreaded		_
Iib::RingBufferConcurrency::MultiThreaded		_
Iib::RingBufferConcurrency::SingleThreaded		_
lib::Filter< F, List > 32 lib::Filter< F, Nil > 33 lib::Flatten< T > 33 lib::Frequency 34 lib::FSM< ID, I, D, Lineage > 34 lib::FSMA ID, I, void, Lineage > 35 lib::FSMBase < D > 35 lib::TransImpl < E, D, Nil, S, T > 70 lib::FSMBase < void > 35 lib::TransImpl < E, void, Nil, S, T > 72 lib::FSMMaker < I, D, T > 36 lib::FstorWrapper < T > 37 lib::GetElem < IDX, List > 37 lib::GetElem < IDX, List > 37 lib::GetElem < IDX, List > 38 lib::CreateStateList < List >::GetStateFromTransition < T > 38 lib::GetValue < Map, Key > 38 lib::Ib 40 lib::If < false, T1, T2 > 40 lib::InheritLineage < T > 41 lib::IsList < Cons < T1, T2 > 42 lib::IsSame < T1, T2 > 42 lib::IsSame < DO(Car < T >), E > 42 lib::CreateTransitionDependencyList < List >::CollectDependencies < E		_
lib::Filter< F, Nil > 33 lib::Flatten< T > 33 lib::Flatten 33 lib::Frequency 34 lib::FSM< ID, I, D, Lineage > 34 lib::FSMS = ID, I, void, Lineage > 35 lib::FSMBase D > 35 lib::TransImpl< E, D, Nil, S, T > 70 lib::FSMBase void > 35 lib::TransImpl< E, void, Nil, S, T > 72 lib::FSMMaker< I, D, T > 36 lib::ForWrapper< T > 37 lib::GetElem< IDX, List > 37 lib::GetElem< IDX, List > 38 lib::GetValue< On List > 38 lib::GetValue< Map, Key > 38 lib::Identity< T > 40 lib::If< false, T1, T2 > 40 hw::Connection:Impl 40 lib::InheritLineage< T > 41 lib::IsList< T > 42 lib::IsSame< T1, T2 > 42 lib::IsSame< DO(Car< T >), E > 42 lib::CreateTransitionDependencyList<		_
lib::Flatten < T > 33 lib::Flatten < Cons < H, T > > 33 lib::Frequency 34 lib::FSM < ID, I, D, Lineage > 34 lib::FSMS = ID, I, void, Lineage > 35 lib::FSMBase < D > 35 lib::TransImpl < E, D, Nil, S, T > 70 lib::FSMBase < void > 35 lib::TransImpl < E, void, Nil, S, T > 72 lib::FSMMaker < I, D, T > 36 lib::ForWrapper < T > 37 lib::GetElem < IDX, List > 37 lib::GetElem < IDX, List > 38 lib::GetElem < O, List > 38 lib::GetValue < Map, Key > 38 lib::Identity < T > 40 lib::If < false, T1, T2 > 40 lib::Ihr < true, T1, T2 > 40 hw::Connection:Impl 40 lib::InheritLineage < T > 41 lib::Ib:List < Cons < T1, T2 > 42 lib::ISsame < DO(Car < T >), E > 42 lib::CreateTransitionDependencyList < List >::CollectDependencies < E		_
lib::Flatten < Cons < H, T >> 33 lib::Frequency 34 lib::FSM < ID, I, D, Lineage > 34 lib::FSM ID, I, void, Lineage > 35 lib::FSMBase < D > 35 lib::TransImpl < E, D, Nil, S, T > 70 lib::FSMBase < void > 35 lib::TransImpl < E, void, Nil, S, T > 72 lib::FSMMaker < I, D, T > 36 lib::FstorWrapper < T > 36 lib::CreateTransitionDependencyList < List >::GetDependency < T > 37 lib::GetElem < IDX, List > 37 lib::GetElem < 0, List > 38 lib::GetValue < Map, Key > 38 lib::Identity < T > 40 lib::If < false, T1, T2 > 40 lib::InheritLineage < T > 41 lib::IsList < Cons < T1, T2 > 42 lib::IsSame < T1, T2 > 42 lib::IsSame < DO(Car < T >), E > 42 lib::CreateTransitionDependencyList < List >::CollectDependencies < E		
Iib::Frequency		
lib::FSM< ID, I, D, Lineage > 34 lib::FSMS ID, I, void, Lineage > 35 lib::FSMBase < D > 35 lib::TransImpl E, D, Nil, S, T > 70 lib::FSMBase < void > 35 lib::TransImpl E, void, Nil, S, T > 72 lib::FSMMaker < I, D, T > 36 lib::FtorWrapper < T > 37 lib::GetElem < IDX, List > 37 lib::GetElem < IDX, List > 37 lib::GetElem < 0, List > 38 lib::GetValue < Map, Key > 38 lib::Identity < T > 40 lib::If < false, T1, T2 > 40 hw::Connection::Impl 40 lib::InheritLineage < T > 41 lib::IsList < T > 42 lib::IsSame < T1, T2 > 42 lib::IsSame < DO(Car < T >), E > 42 lib::CreateTransitionDependencyList < List >::CollectDependencies < E		
lib::FSM ID, I, void, Lineage > 35 lib::FSMBase < D > 35 lib::TransImpl < E, D, Nil, S, T > 70 lib::FSMBase < void > 35 lib::TransImpl < E, void, Nil, S, T > 72 lib::FSMMaker < I, D, T > 36 lib::FtorWrapper < T > 37 lib::Create TransitionDependencyList < List >::GetDependency < T > 37 lib::GetElem < IDX, List > 37 lib::GetElem < 0, List > 38 lib::CreateStateList < List >::GetStateFromTransition < T > 38 lib::GetValue < Map, Key > 38 lib::Identity < T > 40 lib::If < false, T1, T2 > 40 lib::If < true, T1, T2 > 40 lib::InheritLineage < T > 41 lib::IsList < T > 42 lib::IsList < Cons < T1, T2 > 42 lib::IsSame < DO(Car < T >), E > 42 lib::CreateTransitionDependencyList < List >::CollectDependencies < E		
lib::FSMBase < D > 35 lib::TransImpl < E, D, Nil, S, T > 70 lib::FSMBase < void > 35 lib::TransImpl < E, void, Nil, S, T > 72 lib::FSMMaker < I, D, T > 36 lib::FtorWrapper < T > 37 lib::CreateTransitionDependencyList < List >::GetDependency < T > 37 lib::GetElem < IDX, List > 37 lib::GetElem < 0, List > 38 lib::CreateStateList < List >::GetStateFromTransition < T > 38 lib::GetValue < Map, Key > 38 lib::Identity < T > 40 lib::If < false, T1, T2 > 40 lib::If < true, T1, T2 > 40 hw::Connection::Impl 40 lib::InheritLineage < Nil > 41 lib::IsList < T > 42 lib::IsList < Cons < T1, T2 > 42 lib::IsSame < DO(Car < T >), E > 42 lib::CreateTransitionDependencyList < List >::CollectDependencies < E		_
lib::TransImpl < E, D, Nil, S, T > 70 lib::FSMBase < void > 35 lib::TransImpl < E, void, Nil, S, T > 72 lib::FSMMaker < I, D, T > 36 lib::FtorWrapper < T > 37 lib::CreateTransitionDependencyList < List >::GetDependency < T > 37 lib::GetElem < IDX, List > 37 lib::GetElem < 0, List > 38 lib::CreateStateList < List >::GetStateFromTransition < T > 38 lib::GetValue < Map, Key > 38 lib::Identity < T > 40 lib::If < false, T1, T2 > 40 lib::If < true, T1, T2 > 40 hw::Connection::Impl 40 lib::InheritLineage < T > 41 lib::Islsist < T > 42 lib::Islsist < Cons < T1, T2 > 42 lib::IsSame < DO(Car < T >), E > 42 lib::CreateTransitionDependencyList < List >::CollectDependencies < E		
lib::FSMBase < void > 35 lib::TransImpl < E, void, Nil, S, T > 72 lib::FSMMaker < I, D, T > 36 lib::FtorWrapper < T > 37 lib::CreateTransitionDependencyList < List >::GetDependency < T > 37 lib::GetElem < IDX, List > 37 lib::GetElem < 0, List > 38 lib::CreateStateList < List >::GetStateFromTransition < T > 38 lib::GetValue < Map, Key > 38 lib::Identity < T > 40 lib::If < false, T1, T2 > 40 lib::If < true, T1, T2 > 40 hw::Connection::Impl 40 lib::InheritLineage < T > 41 lib::Ib: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		
$\begin{tabular}{ll} lib::TransImpl < E, void, Nil, S, T > & $		
lib::FSMMaker < I, D, T > 36 lib::FtorWrapper < T > 37 lib::Create TransitionDependencyList < List >::GetDependency < T > 37 lib::GetElem < IDX, List > 37 lib::GetElem < 0, List > 38 lib::CreateStateList < List >::GetStateFromTransition < T > 38 lib::GetValue < Map, Key > 38 lib::Identity < T > 40 lib::If < false, T1, T2 > 40 lib::If < true, T1, T2 > 40 hw::Connection::Impl 40 lib::InheritLineage < T > 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		35
$\begin{tabular}{l} lib::FtorWrapper < T > & 37 \\ lib::CreateTransitionDependencyList < List >::GetDependency < T > & 37 \\ lib::GetElem < IDX, List > & 37 \\ lib::GetElem < 0, List > & 38 \\ lib::CreateStateList < List >::GetStateFromTransition < T > & 38 \\ lib::GetValue < Map, Key > & 38 \\ lib::Identity < T > & 40 \\ lib::If < false, T1, T2 > & 40 \\ lib::If < true, T1, T2 > & 40 \\ lib::InheritLineage < T > & 41 \\ lib::InheritLineage < T > & 41 \\ lib::IsList < T > & 42 \\ lib::IsSame < T1, T2 > & 42 \\ lib::IsSame < DO(Car < T >), E > & 42 \\ lib::CreateTransitionDependencyList < List >::CollectDependencies < E \\ \end{tabular}$		
$\label{lib::CreateTransitionDependencyList} \begin{array}{llllllllllllllllllllllllllllllllllll$		
$\begin{tabular}{l} lib::GetElem &< IDX, List > & 38 \\ lib::GetElem &< 0, List > & 38 \\ lib::CreateStateList &< List > ::GetStateFromTransition &< T > & 38 \\ lib::GetValue &< Map, Key > & 38 \\ lib::Identity &< T > & 40 \\ lib::If &< false, T1, T2 > & 40 \\ lib::If &< true, T1, T2 > & 40 \\ lib::InheritLineage &< T > & 40 \\ lib::InheritLineage &< T > & 41 \\ lib::InheritLineage &< Nil > & 41 \\ lib::IsList &< T > & 42 \\ lib::IsSame &< T1, T2 > & 42 \\ lib::IsSame &< DO(Car &< T >), E > & 42 \\ lib::CreateTransitionDependencyList &< List >::CollectDependencies &< E \\ \end{tabular}$	$lib::FtorWrapper < T > \dots \dots$	37
$\begin{tabular}{l} lib::GetElem<0, List>& 38 \\ lib::CreateStateList< List>::GetStateFromTransition& 38 \\ lib::GetValue< Map, Key>& 38 \\ lib::Identity& 40 \\ lib::If< false, T1, T2>& 40 \\ lib::If< true, T1, T2>& 40 \\ lib::If< true, T1, T2>& 40 \\ lib::InheritLineage& 41 \\ lib::InheritLineage& 41 \\ lib::InheritLineage& 42 \\ lib::IsList& 42 \\ lib::IsSame& 42 \\ lib::IsSame), E>& 42 \\ lib::CreateTransitionDependencyList::CollectDependencies$	$\label{lib::CreateTransitionDependencyList} \mbox{List} > \mbox{::GetDependency} < T > \ . \ . \ . \ . \ .$	37
$\begin{tabular}{ll} lib::CreateStateList < List > ::GetStateFromTransition < T > & 38 \\ lib::GetValue < Map, Key > & 38 \\ lib::Identity < T > & 40 \\ lib::If < false, T1, T2 > & 40 \\ lib::If < true, T1, T2 > & 40 \\ lib::If < true, T1, T2 > & 40 \\ lib::InheritLineage < T > & 41 \\ lib::InheritLineage < T > & 41 \\ lib::IsList < T > & 42 \\ lib::IsSame < T1, T2 > & 42 \\ lib::IsSame < DO(Car < T >), E > & 42 \\ lib::CreateTransitionDependencyList < List >::CollectDependencies < E \\ \end{tabular}$	${\sf lib::} {\sf GetElem} {<} {\sf IDX}, {\sf List} {>} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots $	37
$\begin{array}{llllllllllllllllllllllllllllllllllll$	$\label{eq:lib::GetElem} \mbox{lib::GetElem} < \mbox{0, List} > $	38
$\begin{array}{llllllllllllllllllllllllllllllllllll$	$\label{lib::CreateStateList} \mbox{List} > \mbox{::GetStateFromTransition} < T > \label{tib::CreateStateList}$	38
$\begin{array}{llllllllllllllllllllllllllllllllllll$	lib::GetValue < Map, Key >	38
$\begin{array}{llllllllllllllllllllllllllllllllllll$	$lib::Identity < T > \dots \dots$	40
$\begin{array}{llllllllllllllllllllllllllllllllllll$	$\label{eq:lib:interpolation} \mbox{lib::} \mbox{lf} < \mbox{false, T1, T2} > \dots $	40
$\label{lib::InheritLineage} \begin{tabular}{lib::InheritLineage} < T > & 41 \\ \begin{tabular}{lib::InheritLineage} < Nil > & 41 \\ \begin{tabular}{lib::IsList} < T > & 42 \\ \begin{tabular}{lib::IsList} < Cons < T1, T2 > & 42 \\ \begin{tabular}{lib::IsSame} < T1, T2 > & 42 \\ \begin{tabular}{lib::IsSame} < DO(Car < T >), E > & 42 \\ \begin{tabular}{lib::IsCreateTransitionDependencyList} < List >::CollectDependencies < E \\ \end{tabular}$	lib::If $<$ true, T1, T2 $>$	40
lib::InheritLineage Nil >	hw::Connection::Impl	40
$\label{lib::lsList} \begin{array}{llllllllllllllllllllllllllllllllllll$	lib::InheritLineage $<$ T $>$	41
$\label{lib::lsList} \begin{tabular}{l} lib::lsSame < Cons < T1, T2 >> & . & . & . & . & . & . & . & . & . $	lib::InheritLineage < Nil >	41
$\label{lib::lsList} \begin{array}{llllllllllllllllllllllllllllllllllll$	lib::lsList< T >	42
$\label{lib::IsSame} \begin{tabular}{l} lib::IsSame < T1, T2 > $		42
$\label{lib::IsSame} \mbox{Iib::IsSame} < \mbox{DO(Car} < \mbox{T} >), \mbox{E} > \dots $		
lib::CreateTransitionDependencyList< List >::CollectDependencies< E		
>ISOUTTEGLEVETI(< 1 >	>::IsCorrectEvent< T >	41

	42
	42
	43 43
	44
	45
	45
·	45
·	46
lib::SingletonConcurrency::SingleThreaded< T >::Lock	46
3,	46
 	46
	46
	47
	47
	47
	47
	39
	47
· ·	55
	47
	47
	17
lib::LockableObject< CleanupUtility >	47
lib::CleanupUtility	24
lib::LockableObject< Connection >	47
hw::Connection	25
lib::LockableObject< DefaultFormatter >	47
lib::log::DefaultFormatter	30
	47
	43
	47
	48
	47
	49
	+3 47
	53
	47
	61
	47
3	62
${\it lib::} Lockable Object < Thread Manager Impl > \dots $	47
lib::ThreadManagerImpl	66
lib::test::TestManager::Log	48
	49
lib::log::LogRecord	50

6 Class Index

lib::MakeList_0	50
lib::MakeList_1 < A >	50
lib::MakeList_1< A >	51
lib::MakeList_3< A, B, C >	51
	51
lib::MakeList_4< A, B, C, D >	51
lib::MakeList_5< A, B, C, D, E >	-
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 >	52
lib::MakeList_9< A, B, C, D, E, F, G, H, I >	52
lib::Merge < F, T, List >	53
lib::Merge < F, T, Nil >	53
lib::RingBufferConcurrency::MultiThreaded< T >	55
lib::Mutex	55
lib::Nil	56
lib::IsSuperType< Sub, Super >::No	56
lib::Not $<$ T $>$	56
lib::Or $<$ T1, T2 $>$	56
lib::ListOr< List >::OrFn< T1, T2 >	56
lib::qnx::Receiver	57
lib::test::TestManager::Registrar	57
lib::Reverse < List >	57
lib::ReverseCons< Cell >	57
lib::ReverseImpl< Done, ToDo >	57
lib::ReverseImpl< Done, Nil >	58
lib::RingBuffer< T, N, ThreadingPolicy >	58
lib::test::TestManager::Selector	59
lib::Semaphore	59
lib::Setify < List >	59
lib::SetifyImpl< Done, ToDo >	59
lib::SetifyImpl< Done, Nil >	59
lib::SingletonConcurrency::SingleThreaded< T >	60
lib::RingBufferConcurrency::SingleThreaded< T >	60
lib::Singleton < T, TM, P >	60
hw::Motor::Speed	62
hw::Motor::State	62
lib::test::TestManager	63
lib::Thread	63
lib::Time	66
lib::Timer	67
	69
lib::TimerPoolImpl	
•	69
lib::TransImpl < E, D, L, S, T >	69
lib::TransImpl $<$ E, void, DO(Cdr $<$ L $>$), S, T $>$	69
$\label{eq:lib::TransImpl} \mbox{lib::TransImpl} < \mbox{E, void, L, S, T} > \hdots \$	71
lib::Transition $<$ O, E, D $>$	72
lib::Rule< O, E, D, A >	58
lib::TryCall_apply< T, E, D >	72
lib::TryCall_apply< T, E, void >	
	, 0

 $\label{eq:lib::ValueIdentity} \begin{tabular}{ll} $ $1 > 1 \\ $1 > 1 \\ $2 > 1 \\ $3 > 1 \\ 3

8 Class Index

Chapter 3

Class Index

3.1 Class List

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

hw::Actuator	
Singular access point to actuators	17
lib::And< T1, T2 >	18
lib::ListAnd< List >::AndFn< T1, T2 >	18
lib::Apply $< F, T > \dots$	18
$lib::Apply < F, Nil > \dots $	18
lib::Array $<$ T, N $>$	19
lib::Array< T, 0 >	20
lib::log::BaseFilter	20
lib::log::BaseFormatter	20
lib::log::BaseHandler	20
lib::BasicFunctor	21
lib::BasicFunctorImpl< F >	
Basic functor encapsulating anything callable that takes no argu-	
ments	21
lib::Bool< V >	
	22
$lib::Bool < V > \dots \dots \dots \dots \dots \dots \dots \dots \dots $	22
$\label{eq:lib::Bool} \begin{array}{lllllllllllllllllllllllllllllllllll$	22 22 22
$\label{lib::Bool} \begin{split} & \text{lib::Bool} < \text{V} > & & & \\ & \text{lib::Caar} < \text{T} > & & & \\ & \text{lib::Cadr} < \text{T} > & & & \\ \end{split}$	22 22 22 22
$\begin{split} & \text{lib::Bool} < \text{V} > \\ & \text{lib::Caar} < \text{T} > \\ & \text{lib::Cadr} < \text{T} > \\ & \text{lib::Car} < \text{T} > \\ & \text{lib::Car} < \text{Car} > \\ & \text{lib::Car} < \text{Car}$	22 22 22 22 23
lib::Bool< V > lib::Caar< T > lib::Car< T > lib::Cdar< T > lib::Cdar< T >	22 22 22 23 23
lib::Bool< V > lib::Caar< T > lib::Caf< T > lib::Cdar< T > lib::Cddr< T > lib::Cddr< T >	22 22 22 23 23 23
lib::Bool< V > lib::Caar< T > lib::Cadr< T > lib::Cdar< T > lib::Cddr< T > lib::Cdr< T >	22 22 22 23 23 23 23 23
lib::Bool< V > lib::Caar< T > lib::Cadr< T > lib::Cdar< T > lib::Cddr< T > lib::Cdr< T > lib::Cdr< T > lib::qnx::Channel	22 22 22 23 23 23 23 24
lib::Bool < V > lib::Caar < T > lib::Cadr < T > lib::Cdar < T > lib::Cddr < T > lib::Cdr < T > lib::Cdr < T > lib::Tgran::Channel lib::TryCall_apply < T, E, D >::Check < typename, >	22 22 22 23 23 23 23 24
iib::Bool < V >	22 22 22 23 23 23 24 24
iib::Bool	22 22 22 23 23 23 23 24 24

10 Class Index

hw::Connection	25
lib::qnx::Connection	26
lib::Cons $<$ H, T $>$	26
lib::ConsFn< T1, T2 >	
$\label{lib::ConstructFSMLineage} \mbox{lib::ConstructFSMLineage} < \mbox{T} > $	
lib::ConstructFSMLineage < Cons < T, Nil >>	
lib::Contains < List, T >	
$\label{eq:lib::Contains} \mbox{lib::Contains} < \mbox{Nil}, \mbox{T} > \dots \dots \mbo$	27
lib::CreateStateList >	27
lib::CreateTransitionDependencyList< List >	28
lib::CreateTransitionMap< List >	28
lib::FSMMaker< I, D, T >::CreateTransitionTree< TT >	28
lib::Data	29
lib::Decay< T >	29
lib::Decay< const T >	29
lib::Decay< const volatile T >	29
lib::Decay< T & >	
lib::Decay< volatile T >	
lib::log::DefaultFormatter	
Default formatter that lists all information of the passed LogRecord .	30
hw::Motor::Direction	31
$lib:: RingBufferConcurrency:: MultiThreaded < T > :: EmptyLock \\ \ \ . \ \ \ \ . \ \ \ \ \ . \ \ \ \ \ . \ \ \ \ \ \ \ \ \ \ \ \ \$	31
$lib:: RingBufferConcurrency:: SingleThreaded < T > :: EmptyLock \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	31
lib::RingBufferConcurrency::MultiThreaded< T >::FillLock	31
lib::RingBufferConcurrency::SingleThreaded< T >::FillLock	32
lib::log::Filter< F >	
Filter template that accepts functors	32
${\sf lib::Filter} < {\sf F, List} > \ldots \ldots \ldots \ldots \ldots \ldots$	32
$\mbox{lib::Filter} < \mbox{F, Nil} > \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \$	33
$lib::Flatten < T > \dots \dots$	33
$\label{eq:lib::Flatten} \mbox{lib::Flatten} < \mbox{Cons} < \mbox{H}, \mbox{T} >> \ \ \dots \ \ \ \dots \ \ \dots \ \ \dots \ \ \ \$	33
$\label{lib::log::Formatter} \mbox{lib::log::Formatter} < \mbox{F} > \ \ \ldots \ \ \ldots \ \ \ldots \ \ \ \ldots \ \ \ \ldots \ \ \ \ .$	34
lib::Frequency	
Convenience class that allows calculation of a signal's period length	
through its frequency	34
$\label{eq:lib::FSM} \mbox{ID, I, D, Lineage} > \dots $	34
$\label{eq:lib::FSM} \mbox{ID, I, void, Lineage} > \ \ \ldots \ \ \ldots \ \ \ \ldots \ \ \ .$	35
lib::FSMBase < D >	35
${\sf lib::FSMBase} {<} {\sf void} {>} \dots $	35
lib:: FSMMaker < I, D, T > 	36
lib::FtorWrapper< T >	
A functor that calls an object's member function	37
$\label{lib::CreateTransitionDependencyList} \mbox{List} > \mbox{::GetDependency} < \mbox{T} > .$	37
$\label{lib::GetElem} \mbox{lib::GetElem} < \mbox{IDX, List} > \dots \dots \m$	37
$\mbox{lib::GetElem} < 0, \mbox{List} > \ \dots \$	38
$\label{lib::CreateStateList} \mbox{List} > \mbox{::GetStateFromTransition} < \mbox{T} > $	38
$\label{lib::GetValue} \mbox{lib::GetValue} < \mbox{Map, Key} > \ \dots \$	38
lib::log::Handler< F >	
Handler template that holds a functor	38

3.1 Class List

hw::HWAccessImpl Interface for direct hardware access
lib::Identity< T >
lib::If < false, T1, T2 >
lib::If< true, T1, T2 >
hw::Connection::Impl
lib::InheritLineage< T >
lib::InheritLineage< Nil >
lib::Int< I >
lib::CreateTransitionDependencyList< List >::CollectDependencies< E >::-
IsCorrectEvent< T >
lib::lsList< T >
lib::lsList< Cons< T1, T2 >>
lib::lsSame < T1, T2 >
lib::lsSame $< T, T > \dots $ 42
lib::lsSuperType < Sub, Super >
lib::Join < List, Appendage >
lib::Join < Nil, Appendage >
hw::LED
Allows access to LEDs
lib::ListAnd< List >
lib::ListOr< List >
lib::ListToMap < List >
lib::ListToMapImpl< IDX, List >
lib::ListToMapImpl< IDX, Nil >
lib::SingletonConcurrency::SingleThreaded< T >::Lock
lib::SingletonConcurrency::MultiThreaded< T >::Lock
hw::HWAccessImpl::Lock
lib::Lock< T, E, R >
lib::LockableClass< T, M >::Lock
lib::LockableObject< T, M >::Lock
lib::LockableClass $<$ T, M $>$
lib::LockableObject< T, M >
lib::test::TestManager::Log
lib::log::Logger
Logger class
lib::log::LogLevel
lib::log::LogManagerImpl
LogManager Singleton, grants access to Logger instances 49
lib::log::LogRecord
lib::MakeList_0
lib::MakeList_1 < A >
lib::MakeList_2< A, B >
lib::MakeList_3< A, B, C >
lib::MakeList_4< A, B, C, D >
lib::MakeList_5< A, B, C, D, E >
lib::MakeList_6< A, B, C, D, E, F >
lib::MakeList_7< A, B, C, D, E, F, G >
lib::MakeList_8< A, B, C, D, E, F, G, H >
lib::MakeList 9< A, B, C, D, E, F, G, H, I >

12 Class Index

lib::Merge < F, T, List >	53
lib::Merge < F, T, Nil >	53
hw::Motor	
Client interface for controlling the conveyor belt and electromagnetic	
switch	53
lib::SingletonConcurrency::MultiThreaded< T >	55
lib::RingBufferConcurrency::MultiThreaded< T >	55
lib::Mutex	55
lib::Nil	56
lib::lsSuperType< Sub, Super >::No	56
lib::Not < T >	56
lib::Or< T1, T2 >	56
lib::ListOr< List >::OrFn< T1, T2 >	56
lib::qnx::Receiver	57
lib::test::TestManager::Registrar	57
lib::Reverse < List >	57
lib::ReverseCons< Cell >	57
lib::ReverseImpl< Done, ToDo >	57
lib::ReverseImpl< Done, Nil >	58
lib::RingBuffer< T, N, ThreadingPolicy >	58
lib::Rule < O, E, D, A >	58
lib::test::TestManager::Selector	59
lib::Semaphore	59
lib::Setify< List >	59
lib::SetifyImpl< Done, ToDo >	59
lib::SetifyImpl< Done, Nil >	59
lib::SingletonConcurrency::SingleThreaded< T >	60
lib::RingBufferConcurrency::SingleThreaded <t></t>	60
lib::Singleton< T, TM, P >	
Template for convenient Singleton creation	60
lib::SmartPtr< T >	
Smart pointer class for automatic life time management	61
hw::Motor::Speed	62
hw::Motor::State	62
lib::log::StreamHandler	
Handler compatible functor that writes its LogRecord to an std-	
::stream instance	62
lib::test::TestManager	63
lib::Thread	
Encapsulates the most important features of a thread	63
lib::ThreadManagerImpl	66
lib::Time	
Data class representing a timeframe with microsecond accuracy	66
lib::Timer	
Timer that allows scheduling of functors	67
lib::TimerPoolImpl	69
$\label{lib::CreateTransitionMap} \mbox{List} > \mbox{::Transform} < \mbox{T} > \mbox{. } \mbox{. } \mbox{. } \mbox{. } \mbox{. } \mbox{.} .$	69
lib::TransImpl $<$ E, D, L, S, T $>$	69
$\label{eq:lib::TransImpl} \mbox{lib::TransImpl} < \mbox{E, D, Nil, S, T} > \dots \dots \dots \dots \dots \dots \dots \dots \dots $	70
lib::TransImpl $<$ E, void, L, S, T $>$	71

3.1 Class List

lib::TransImpl $<$ E, void, NiI, S, T $>$	 									72
lib::Transition $<$ O, E, D $>$	 									72
lib::TryCall_apply< T, E, D $>$	 									72
lib::TryCall_apply $<$ T, E, void $>$	 									73
lib::test::UnitTest	 									73
lib::Value $<$ T, I $>$	 									73
lib::ValueIdentity< Bool< I $>>$	 									74
lib::ValueIdentity< Int< I $>>$	 									74
lib::ValueIdentity $<$ Value $<$ T, I $>>$	 									74
lib::IsSuperType< Sub_Super >::Yes										74

14 Class Index

Chapter 4

Namespace Documentation

4.1 lib::SingletonConcurrency Namespace Reference

Contains threading models of the Singleton template.

Classes

- struct SingleThreaded
- struct MultiThreaded

4.1.1 Detailed Description

Contains threading models of the Singleton template.

Chapter 5

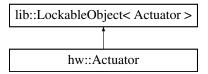
Class Documentation

5.1 hw::Actuator Class Reference

Singular access point to actuators.

#include <Actuator.h>

Inheritance diagram for hw::Actuator:



Public Types

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

Public Member Functions

• const Channel & getChannel () const

Static Public Attributes

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

- static const uint8 t MOTOR BELT = 0x01
- static const uint8_t MOTOR_SWITCH = 0x02
- static const int CCMD = 3

5.1.1 Detailed Description

Singular access point to actuators.

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

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

Static Public Attributes

• static const bool value = T1::value && T2::value

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

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

Public Types

typedef And< T1, T2 > Type

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

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

Public Types

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

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

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

Public Types

• typedef Nil Type

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

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

Public Types

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

Public Member Functions

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

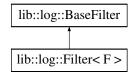
template<typename T, std::size $_{-}$ t N> class lib::Array< T, N>

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

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

5.8 lib::log::BaseFilter Struct Reference

Inheritance diagram for lib::log::BaseFilter:

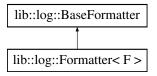


Public Member Functions

• virtual bool accept (const LogRecord &)=0

5.9 lib::log::BaseFormatter Struct Reference

Inheritance diagram for lib::log::BaseFormatter:

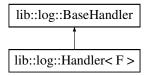


Public Member Functions

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

5.10 lib::log::BaseHandler Struct Reference

Inheritance diagram for lib::log::BaseHandler:

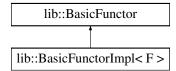


Public Member Functions

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

5.11 lib::BasicFunctor Struct Reference

Inheritance diagram for lib::BasicFunctor:



Public Member Functions

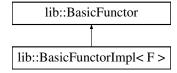
• virtual void **operator()** ()=0

5.12 lib::BasicFunctorImpl < F > Class Template Reference

Basic functor encapsulating anything callable that takes no arguments.

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

Inheritance diagram for lib::BasicFunctorImpl< F >:



Public Member Functions

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

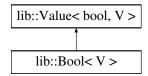
5.12.1 Detailed Description

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

Basic functor encapsulating anything callable that takes no arguments.

5.13 lib::Bool < V > Struct Template Reference

Inheritance diagram for lib::Bool < V >:



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

5.14 lib::Caar < T > Struct Template Reference

Public Member Functions

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

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

5.15 lib::Cadr < T > Struct Template Reference

Public Member Functions

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

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

5.16 lib::Car < T > Struct Template Reference

Public Types

• typedef T::Head Type

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

5.17 lib::Cdar < T > Struct Template Reference

Public Member Functions

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

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

5.18 lib::Cddr< T > Struct Template Reference

Public Member Functions

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

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

5.19 lib::Cdr < T > Struct Template Reference

Public Types

• typedef T::Tail Type

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

5.20 lib::qnx::Channel Class Reference

Public Member Functions

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

Friends

· class Receiver

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

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

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

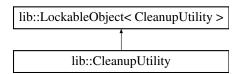
template<typename T, typename E>template<typename, void(*)(const E &)> struct lib::TryCall_apply< T, E, void >::Check< typename, >

5.23 lib::CleanupUtility Class Reference

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

#include <CleanupUtility.h>

Inheritance diagram for lib::CleanupUtility:



Classes

• struct Compare

Public Member Functions

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

Static Public Member Functions

• static CleanupUtility & instance ()

Static Public Attributes

• static const size_t **DEFAULT_PRIORITY** = 10

5.23.1 **Detailed Description**

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

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

lib::CreateTransitionDependencyList< 5.24 List >::Collect-**Dependencies** < E > Struct Template Reference

Classes

struct IsCorrectEvent

Public Member Functions

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

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

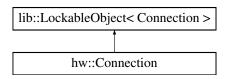
5.25 lib::Condition Class Reference

Public Member Functions

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

5.26 hw::Connection Class Reference

Inheritance diagram for hw::Connection:



Classes

class Impl

Public Member Functions

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

5.27 lib::qnx::Connection Class Reference

Public Member Functions

• void send (Data_ptr) const

Friends

· class Channel

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

Public Types

- · typedef H Head
- · typedef T Tail

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

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

Public Types

• typedef Cons < T1, T2 > Type

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

5.30 lib::ConstructFSMLineage < T > Struct Template Reference

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

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

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

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

Static Public Attributes

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

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

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

Static Public Attributes

• static const bool value = false

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

5.34 lib::CreateStateList < List > Struct Template Reference

Classes

• struct GetStateFromTransition

Public Member Functions

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

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

5.35 lib::CreateTransitionDependencyList > Struct - Template Reference

Classes

- struct CollectDependencies
- struct GetDependency

Public Member Functions

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

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

5.36 lib::CreateTransitionMap < List > Struct Template Reference

Classes

• struct Transform

Public Member Functions

typedef DO (Apply< Transform, List >) Type

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

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

Public Types

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

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

5.38 lib::Data Class Reference

Public Types

typedef lib::SmartPtr< Data > Data_ptr

Public Member Functions

- void * data ()
- const void * data () const
- size t size () const

Static Public Member Functions

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

5.39 lib::Decay < T > Struct Template Reference

Public Types

• typedef T Type

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

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

Public Types

typedef Decay
 Type Type

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

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

Public Types

typedef Decay< T >::Type Type

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

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

Public Types

typedef Decay< T >::Type Type

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

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

Public Types

typedef Decay< T >::Type Type

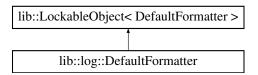
template<typename T> struct lib::Decay< volatile T>

5.44 lib::log::DefaultFormatter Class Reference

Default formatter that lists all information of the passed LogRecord.

#include <DefaultFormat.h>

Inheritance diagram for lib::log::DefaultFormatter:



Public Member Functions

• std::string operator() (const LogRecord &)

Static Public Member Functions

• static std::string toDate (uint64_t)

5.44.1 Detailed Description

Default formatter that lists all information of the passed LogRecord.

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

5.45 hw::Motor::Direction Struct Reference

Static Public Attributes

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

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

Public Member Functions

EmptyLock (MultiThreaded< T > *t)

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

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

Public Member Functions

EmptyLock (SingleThreaded< T > *)

 $template < typename \ T > struct \ lib:: RingBuffer Concurrency:: Single Threaded < T > :: Empty Lock$

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

Public Member Functions

FillLock (MultiThreaded< T > *t)

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

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

Public Member Functions

• FillLock (SingleThreaded< T > *)

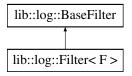
 $template < typename \ T > struct \ lib:: RingBuffer Concurrency:: Single Threaded < T > :: Fill Lock$

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

Filter template that accepts functors.

```
#include <Filter.hpp>
```

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



Public Member Functions

- Filter (F f)
- bool accept (const LogRecord &Ir)

5.50.1 Detailed Description

 ${\it template}{<}{\it typename}\;{\it F}{>}{\it struct}\;{\it lib::}{\it log::}{\it Filter}{<}\;{\it F}{>}$

Filter template that accepts functors.

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

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

Public Types

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

Public Member Functions

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

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

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

Public Types

typedef Nil Type

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

5.53 lib::Flatten < T > Struct Template Reference

Public Types

• typedef T Type

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

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

Public Types

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

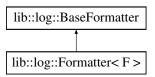
Public Member Functions

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

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

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

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



Public Member Functions

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

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

5.56 lib::Frequency Class Reference

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

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

Static Public Member Functions

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

5.56.1 Detailed Description

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

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

Public Types

typedef TryCall_enter< I, D > EnterFunction

Public Member Functions

• **FSM** (D d)

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

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

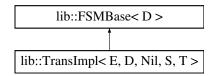
Public Types

typedef TryCall_enter< I, void > EnterFunction

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

5.59 lib::FSMBase < D > Struct Template Reference

Inheritance diagram for lib::FSMBase< D >:



Public Member Functions

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

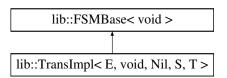
Public Attributes

- · int state_
- D data

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

5.60 lib::FSMBase < void > Struct Template Reference

Inheritance diagram for lib::FSMBase< void >:



Public Member Functions

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

Public Attributes

• int state_

template<> struct lib::FSMBase< void >

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

Classes

• struct CreateTransitionTree

Public Types

- · typedef | 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, Initial-State>)>::value template<typename I, typename D, typename T> struct lib::FSMMaker< I, D, T>

5.62 lib::FtorWrapper < T > Class Template Reference

A functor that calls an object's member function.

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

Public Member Functions

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

5.62.1 Detailed Description

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

A functor that calls an object's member function.

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

Public Types

• typedef Cons< typename T::Origin, typename T::Destination > Tmp

Public Member Functions

• typedef MAKELIST (typename T::Event, Tmp) Type

 $template < typename\ List > template < typename\ T > struct\ lib::Create Transition Dependency List < List > ::Get Dependency < T >$

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

Public Member Functions

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

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

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

Public Member Functions

typedef DO (Car< List >) Type

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

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

Public Member Functions

• typedef MAKELIST (typename T::Origin, typename T::Destination) Type

 $template < typename\ T> struct\ lib:: Create State List < :: Get State From Transition < T>$

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

Public Types

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

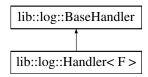
template<typename Map, typename Key> struct lib::GetValue< Map, Key>

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

Handler template that holds a functor.

#include <Handler.hpp>

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



Public Member Functions

• Handler (F f, Formatter ptr p)

5.68.1 Detailed Description

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

Handler template that holds a functor.

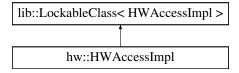
All accepted LogRecords of a Logger instance are passed to the Logger's handlers. There they are run through a formatter; and the formatters output is passed to the functor.

5.69 hw::HWAccessImpl Class Reference

Interface for direct hardware access.

#include <HWAccess.h>

Inheritance diagram for hw::HWAccessImpl:



Classes

struct Lock

Public Types

- typedef lib::LockableClass < HWAccessImpl > Super
- typedef lib::Singleton < HWAccessImpl, lib::SingletonConcurrency::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

5.69.1 Detailed Description

Interface for direct hardware access.

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

5.70 lib::Identity < T > Struct Template Reference

Public Types

· typedef T Type

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

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

Public Types

• typedef T2::Type Type

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

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

Public Types

• typedef T1::Type Type

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

5.73 hw::Connection::Impl Class Reference

Classes

- struct DoneRunning
- struct Packet

Friends

· class Connection

5.74 lib::InheritLineage < T > Struct Template Reference

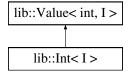
 $template\!<\!typename\ T\!>struct\ lib::lnheritLineage\!<\!T\!>$

5.75 lib::InheritLineage < Nil > Struct Template Reference

template<> struct lib::InheritLineage< Nil >

5.76 lib::Int < I > Struct Template Reference

Inheritance diagram for lib::Int< l >:



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

5.77 lib::CreateTransitionDependencyList< List >::Collect-Dependencies< E >::IsCorrectEvent< T > Struct Template Reference

 $\label{limit} \begin{tabular}{ll} Inheritance & diagram & for & lib::CreateTransitionDependencyList< & List & >::Collect-Dependencies< E >::lsCorrectEvent< T >: \\ \end{tabular}$



 $template < typename \ List > template < typename \ E > template < typename \ T > \ struct \ lib::Create-typename \ TransitionDependencyList < List > ::CollectDependencies < E > ::lsCorrectEvent < T >$

5.78 lib::lsList< T > Struct Template Reference

Static Public Attributes

• static const bool value = false

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

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

Static Public Attributes

• static const bool value = true

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

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

Static Public Attributes

• static const bool value = false

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

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

Static Public Attributes

• static const bool value = true

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

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

Classes

- struct No
- struct Yes

Static Public Member Functions

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

Static Public Attributes

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

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

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

Public Types

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

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

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

Public Types

• typedef Appendage Type

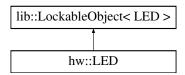
template<typename Appendage> struct lib::Join< Nil, Appendage >

5.85 hw::LED Class Reference

Allows access to LEDs.

#include <LED.h>

Inheritance diagram for hw::LED:



Public Types

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

Public Member Functions

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

Static Public Attributes

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

Friends

· class Actuator

5.85.1 Detailed Description

Allows access to LEDs.

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

5.86 lib::ListAnd List > Struct Template Reference

Classes

• struct AndFn

Public Member Functions

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

Static Public Attributes

• static const bool value = Type::value

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

5.87 lib::ListOr < List > Struct Template Reference

Classes

struct OrFn

Public Member Functions

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

Static Public Attributes

• static const bool value = Type::value

 $template {<} typename\ List{>}\ struct\ lib:: ListOr {<}\ List{>}$

5.88 lib::ListToMap < List > Struct Template Reference

Public Member Functions

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

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

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

Public Types

• typedef Cons< Cons< Int< IDX > , DO(Car< List >)>, > Type

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

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

Public Types

• typedef Nil Type

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

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

Public Member Functions

Lock (Mutex *)

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

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

Public Member Functions

• Lock (Mutex *mtx)

Public Attributes

Mutex * mtx_

 $template < typename \ T > struct \ lib:: Singleton Concurrency:: Multi Threaded < T > :: Lock$

- 5.93 hw::HWAccessImpl::Lock Struct Reference
- 5.94 lib::Lock < T, E, R > Class Template Reference

Public Types

• typedef T Mutex

Public Member Functions

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

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

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

Public Member Functions

• Lock (T *)

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

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

Public Member Functions

• Lock (T *t)

 $template < typename \ T, \ typename \ M = Mutex > struct \ lib::Lockable Object < T, \ M > ::Lockable Object < T, \ M > ::Lockable$

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

Classes

struct Lock

Public Types

· typedef M Mutex

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

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

Classes

struct Lock

Public Types

· typedef M Mutex

Friends

· class Lock

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

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

Public Member Functions

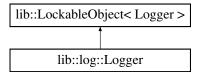
- virtual std::string read ()=0
- virtual bool isEmpty () const =0

5.100 lib::log::Logger Class Reference

Logger class.

```
#include <Logger.h>
```

Inheritance diagram for lib::log::Logger:



Public Types

typedef SmartPtr< Logger > Logger_ptr

Public Member Functions

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

Friends

- class LogManagerImpl
- class SmartPtr< Logger >

5.100.1 Detailed Description

Logger class.

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

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

5.101 lib::log::LogLevel Class Reference

Public Member Functions

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

Static Public Attributes

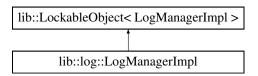
- static const LogLevel INFO
- static const LogLevel WARNING
- static const LogLevel ERROR
- · static const LogLevel CRITICAL

5.102 lib::log::LogManagerImpl Class Reference

LogManager Singleton, grants access to Logger instances.

```
#include <LogManager.h>
```

Inheritance diagram for lib::log::LogManagerImpl:



Public Member Functions

- Logger ptr rootLog ()
- Logger_ptr getLog (const std::string &)

5.102.1 Detailed Description

LogManager Singleton, grants access to Logger instances.

Creates and exposes Logger instances by alphanumerical id.

Offers a "root log" for convenience.

5.103 lib::log::LogRecord Class Reference

Public Member Functions

- LogRecord (LogLevel II, uint64_t ts, uint16_t tid, const std::string &msg, const char *f=NULL, int I=-1)
- const LogLevel & logLevel () const
- uint64_t timestamp () const
- uint16 t threadID () const
- const std::string & message () const
- const char * file () const
- int line () const
- bool hasFile () const

5.104 lib::MakeList_0 Struct Reference

Public Types

· typedef Nil Type

5.105 lib::MakeList_1 < A > Struct Template Reference

Public Types

typedef Cons< A, MAKELIST_0 > Type

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

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

Public Types

• typedef Cons < A, MAKELIST_1(B) > Type

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

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

Public Types

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

template<typename A, typename B, typename C> struct lib::MakeList $_3<$ A, B, C>

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

Public Types

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

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

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

Public Types

typedef Cons
 A, MAKELIST_4(B, C, D, E)
 Type

template<typename A, typename B, typename C, typename D, typename E> struct lib::MakeList_5< A, B, C, D, E>

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

Public Types

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

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

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

Public Types

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

template<typename A, typename B, typename C, typename D, typename E, typename F, typename G> struct lib::MakeList $_-$ 7< A, B, C, D, E, F, G>

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

Public Types

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

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

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

Public Types

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

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

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

Public Member Functions

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

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

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

Public Types

• typedef T Type

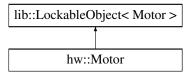
template<template< typename, typename > class F, typename T> struct lib::Merge< F, T, Nil >

5.116 hw::Motor Class Reference

Client interface for controlling the conveyor belt and electromagnetic switch.

#include <Motor.h>

Inheritance diagram for hw::Motor:



Classes

- struct Direction
- struct Speed
- struct State

Public Types

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

Public Member Functions

void controlBelt (pid_t dir, pid_t speed)

Controls conveyor belt.

void controlSwitch (pid_t state)

Controls electromagnetic switch.

Static Public Attributes

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

Friends

class Actuator

5.116.1 Detailed Description

Client interface for controlling the conveyor belt and electromagnetic switch.

5.116.2 Member Function Documentation

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

Controls conveyor belt.

Parameters

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

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

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

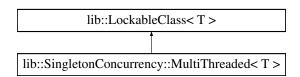
Controls electromagnetic switch.

Parameters

state | State the switch is supposed to be in.

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

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



Classes

struct Lock

 $template < typename \ T > struct \ lib:: Singleton Concurrency:: Multi Threaded < T >$

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

Classes

- class EmptyLock
- class FillLock

Public Member Functions

• std::size_t size () const

 $template {<} typename~T {>}~class~lib:: RingBufferConcurrency:: MultiThreaded {<}~T {>}~$

5.119 lib::Mutex Class Reference

Public Member Functions

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

5.120 lib::Nil Struct Reference

Public Types

• typedef Nil Type

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

Public Attributes

• char v [2]

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

5.122 lib::Not < T > Struct Template Reference

Static Public Attributes

• static const bool value = !T::value

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

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

Static Public Attributes

• static const bool value = T1::value || T2::value

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

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

Public Types

• typedef Or< T1, T2 > **Type**

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

5.125 lib::qnx::Receiver Class Reference

Public Member Functions

• Data_ptr receive () const

Friends

· class Channel

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

Public Member Functions

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

5.127 lib::Reverse < List > Struct Template Reference

Public Member Functions

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

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

5.128 lib::ReverseCons < Cell > Struct Template Reference

Public Types

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

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

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

Public Types

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

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

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

Public Types

· typedef Done Type

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

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

Public Member Functions

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

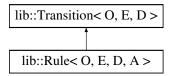
Static Public Attributes

• static const std::size_t capacity = N

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

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

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



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

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

Public Member Functions

• Selector (const std::string &unit_id)

5.134 lib::Semaphore Class Reference

Public Member Functions

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

5.135 lib::Setify < List > Struct Template Reference

Public Member Functions

typedef DO (SetifyImpl< Nil, List >) Type

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

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

Public Types

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

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

Public Member Functions

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

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

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

Classes

struct Lock

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

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

Classes

- struct EmptyLock
- struct FillLock

Public Member Functions

• std::size_t size () const

 $template < typename \ T > class \ lib:: RingBuffer Concurrency:: Single Threaded < T >$

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

Template for convenient Singleton creation.

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

Static Public Member Functions

static T & instance ()
 Access singleton class implementation.

5.140.1 Detailed Description

 $\label{template} \mbox{typename T, template} < \mbox{typename} > \mbox{class TM = SingletonConcurrency::Single-Threaded, size_t P = CleanupUtility::DEFAULT_PRIORITY> class lib::Singleton< T, TM, P> \\ \mbox{typename} < \mbox{T, TM, P} > \\ \mbox{T, T$

Template for convenient Singleton creation.

Parameters are:

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

5.140.2 Member Function Documentation

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

Access singleton class implementation.

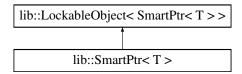
Uses the double checked locking pattern for creation synchronization.

5.141 lib::SmartPtr < T > Class Template Reference

Smart pointer class for automatic life time management.

```
#include <SmartPtr.hpp>
```

Inheritance diagram for lib::SmartPtr< T >:



Public Member Functions

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

5.141.1 Detailed Description

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

Smart pointer class for automatic life time management.

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

5.142 hw::Motor::Speed Struct Reference

Static Public Attributes

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

5.143 hw::Motor::State Struct Reference

Static Public Attributes

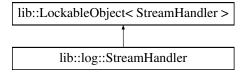
- static const pid_t **OPEN** = 0
- static const pid_t CLOSE = 1

5.144 lib::log::StreamHandler Class Reference

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

```
#include <StreamHandler.h>
```

Inheritance diagram for lib::log::StreamHandler:



Public Member Functions

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

5.144.1 Detailed Description

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

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

5.145 lib::test::TestManager Class Reference

Classes

- struct Log
- struct Registrar
- struct Selector

Public Types

typedef void(* testFn)(void)

Public Member Functions

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

Static Public Member Functions

• static TestManager & Instance ()

5.146 lib::Thread Class Reference

Encapsulates the most important features of a thread.

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

Public Member Functions

• Thread ()

Default constructor.

template<typename F > Thread (F)

Constructor taking functor to execute in new thread.

• Thread (Thread &)

Copy constructor; Moves content to this Thread object.

∼Thread ()

Destructor.

• Thread & operator= (Thread &)

Assignment operator.

• void join ()

Calls join on the Thread.

• bool joinable () const

Wether or not the Thread is joinable.

Protected Member Functions

• void run ()

This is called from the new Thread.

Static Protected Member Functions

static void * entryPoint (void *)

5.146.1 Detailed Description

Encapsulates the most important features of a thread.

5.146.2 Constructor & Destructor Documentation

```
5.146.2.1 lib::Thread::Thread(void)
```

Default constructor.

Initializes inert Thread

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

Constructor taking functor to execute in new thread.

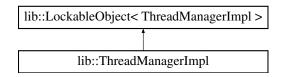
Warning

throws std::runtime_error if thread cannot be started.

```
5.146.2.3 lib::Thread::~Thread (void)
Destructor.
Warning
    terminates if this Thread is still joinable
5.146.3 Member Function Documentation
5.146.3.1 void lib::Thread::join (void)
Calls join on the Thread.
Warning
    must be called from the same context as ctor.
    throws std::runtime_error if this Thread is not joinable
5.146.3.2 boollib::Thread::joinable()const [inline]
Wether or not the Thread is joinable.
Warning
    Thread cannot be destroyed while joinable
5.146.3.3 Thread & lib::Thread::operator= ( Thread & t )
Assignment operator.
Moves content to this Thread object.
Warning
    terminates if this Thread is already joinable
5.146.3.4 void lib::Thread::run ( void ) [protected]
This is called from the new Thread.
It executes the user's functor.
Warning
    terminates if functor throws an exception
```

5.147 lib::ThreadManagerImpl Class Reference

Inheritance diagram for lib::ThreadManagerImpl:



Public Types

• typedef uint16_t tid_t

Public Member Functions

- tid t addThread (pthread t)
- void removeThread (pthread_t)
- tid_t getThread (pthread_t)
- tid_t getCurrent ()

5.148 lib::Time Class Reference

Data class representing a timeframe with microsecond accuracy.

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

Public Types

typedef uint32_t us_t

Public Member Functions

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

Static Public Member Functions

- static Time h (us_t v)
- static Time min (us_t v)
- static Time s (us t v)

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

Static Public Attributes

- static const uint32_t MS_TO_US = 1000
- static const uint32 t S_TO_MS = 1000
- static const uint32_t **M_TO_S** = 60
- static const uint32_t **H_TO_M** = 60
- static const uint32 t **S_TO_US** = S TO MS * MS TO US
- static const uint32_t M_TO_US = M_TO_S * S_TO_US
- static const uint32_t **H_TO_US** = H_TO_M * M_TO_US

5.148.1 Detailed Description

Data class representing a timeframe with microsecond accuracy.

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

5.149 lib::Timer Class Reference

Timer that allows scheduling of functors.

```
#include <Timer.h>
```

Classes

· struct Ftor

Public Types

• typedef uint64_t ts_t

Public Member Functions

- void sync (Time t)
 - Synchronizes execution.
- void reset ()
- Time delta ()

Amount of time elapsed since last reset.

• Time elapsed ()

Amount of time elapsed since last reset.

```
    template<typename F > void executeWhen (Time, F)
```

· bool active () const

Is timer currently waiting for ftor execution.

· void deactivate ()

Static Public Member Functions

• static void deactivateAll ()

Deactivate all timers.

• static ts_t timestamp ()

Returns current system time in nanoseconds since Jan.

5.149.1 Detailed Description

Timer that allows scheduling of functors.

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

Also allows for synchronisation to a specific time frame.

5.149.2 Member Function Documentation

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

Is timer currently waiting for ftor execution.

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

Deactivate all timers.

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

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

Amount of time elapsed since last reset.

Resets timer.

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

Amount of time elapsed since last reset.

Doesn't reset timer.

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

Synchronizes execution.

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

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

Returns current system time in nanoseconds since Jan.

1st 1970.

5.150 lib::TimerPoolImpl Class Reference

Public Types

typedef Singleton
 TimerPoolImpl
 SingletonInst

Friends

· class Timer

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

Public Types

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

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

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

Public Types

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

Public Member Functions

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

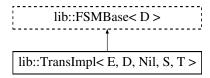
Static Public Attributes

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

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

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

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



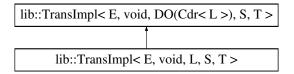
Public Member Functions

• virtual void process (const E &e)

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

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

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



Public Types

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

Public Member Functions

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

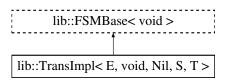
Static Public Attributes

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

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

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

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



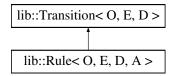
Public Member Functions

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

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

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

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



Public Types

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

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

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

Classes

struct Check

Static Public Member Functions

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

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

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

Classes

struct Check

Static Public Member Functions

- template<typename >
 static void test (const E &e,...)
- static void call (const E &e)

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

5.159 lib::test::UnitTest Class Reference

Static Public Member Functions

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

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

Static Public Attributes

• static const T value = I

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

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

Static Public Attributes

• static const int value = I

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

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

Static Public Attributes

• static const int value = I

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

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

Static Public Attributes

• static const T value = I

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

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

Public Attributes

• char **v** [1]

 $template < typename \ Sub, \ typename \ Super > struct \ lib:: lsSuper Type < Sub, \ Super > :: Yes$