TI4_SE2

Generated by Doxygen 1.7.6.1

Wed May 18 2016 10:33:01

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:continuous} \mbox{lib::Caar} < T > \mbox{Struct Template Reference} \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ \ . \ \ \ \ \ \ \ \ \ \ \ \ \$	22
5.15	$\label{eq:continuous_continuous_continuous} \mbox{lib::Cadr} < T > \mbox{Struct Template Reference} \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ \ \ \ \ \ \ \ \ \ \ \$	22
5.16	haw::Calibrator Class Reference	22
5.17	$\label{eq:car_def} \mbox{lib::Car} < T > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	23
5.18	$\label{eq:continuous} \mbox{lib::Cdar} < T > \mbox{Struct Template Reference} $	23
5.19	$\label{eq:conditional} \mbox{lib::Cddr} < T > \mbox{Struct Template Reference} $	23
5.20	$\label{eq:lib::Cdr} \mbox{lib::Cdr} < T > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	23
5.21	lib::qnx::Channel Class Reference	23
5.22	$\label{lib::TryCall_apply} \begin{subarray}{ll} \$	24
5.23	lib::TryCall_apply< T, E, void >::Check< typename, > Struct Template Reference	24
5.24	lib::CleanupUtility Class Reference	24
	5.24.1 Detailed Description	25
5.25	lib::CreateTransitionDependencyList< List >::CollectDependencies< E > Struct Template Reference	25
5.26	lib::Condition Class Reference	25
5.27	haw::Config Class Reference	26
5.28	lib::qnx::Connection Class Reference	26
5.29	hw::Connection Class Reference	26
5.30	lib::Cons< H, T > Struct Template Reference	27
5.31	lib::ConsFn< T1, T2 > Struct Template Reference	27
5.32	$\label{eq:constructFSML} \mbox{lib::ConstructFSMLineage} < T > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	27
5.33	${\it lib::} {\it ConstructFSMLineage} {< cons} {< t, Nil} > {> struct Template Reference}$	27
5.34	$\label{eq:lib::Contains} \mbox{List, T} > \mbox{Struct Template Reference} $	28
5.35	lib::Contains < Nil, T > Struct Template Reference	28
5.36	$\label{lib::CreateStateList} \mbox{List} > \mbox{Struct Template Reference} $	28
5.37	${\it lib::} Create Transition Dependency List < List > Struct\ Template\ Reference$	28
5.38	$\label{eq:continuous_lib::CreateTransitionMap} \textbf{List} > \textbf{Struct Template Reference} \dots . .$	29
5.39	$\label{lib::FSMMaker} \begin{tabular}{ll} lib::FSMMaker < I, D, T > :: Create Transition Tree < TT > Struct Template \\ Reference & $	29
5.40	lib::Data Class Reference	29
5.41	$\label{eq:lib::Decay} \mbox{lib::Decay} < T > \mbox{Struct Template Reference} $	30

CONTENTS iii

5.42	$\label{eq:lib::Decay} \mbox{const T} > \mbox{Struct Template Reference} $	30
5.43	$\label{eq:lib::Decay} \mbox{const volatile T} > \mbox{Struct Template Reference} $	30
5.44	$\label{eq:lib::Decay} \mbox{Lib::Decay} < \mbox{T \&} > \mbox{Struct Template Reference} $	30
5.45	$\label{eq:lib::Decay} \mbox{lib::Decay} < \mbox{volatile T} > \mbox{Struct Template Reference} $	31
5.46	lib::log::DefaultFormatter Class Reference	31
	5.46.1 Detailed Description	31
5.47	hw::Motor::Direction Struct Reference	31
5.48	$\label{lib::RingBufferConcurrency::SingleThreaded} Iib::RingBufferConcurrency::SingleThreaded < T > ::EmptyLock Struct - Reference$	32
5.49	$\label{lib::RingBufferConcurrency::MultiThreaded} \begin{tabular}{ll} Ibb::RingBufferConcurrency::MultiThreaded<&T>::EmptyLock Class - Reference$	32
5.50	haw::Event Class Reference	32
5.51	$\label{lib::RingBufferConcurrency::SingleThreaded} \begin{tabular}{ll} $T>::FillLock & Struct - Reference & $	33
5.52	$\label{lib::RingBufferConcurrency::MultiThreaded} \begin{tabular}{ll} Ibb::RingBufferConcurrency::MultiThreaded<&T>::FillLock&Class&-Reference&$	33
5.53	$\label{eq:lib::log::Filter} \mbox{lib::log::Filter} < \mbox{F} > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	33
	5.53.1 Detailed Description	34
5.54	$\label{eq:lib::Filter} \mbox{\sc F, List} > \mbox{\sc Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	34
5.55	$\label{eq:lib::Filter} \mbox{Filter} < \mbox{F, Nil} > \mbox{Struct Template Reference} $	34
5.56	$\label{eq:lib::Flatten} \mbox{lib::Flatten} < T > \mbox{Struct Template Reference} \ \ . \ \ \$	34
5.57	lib::Flatten< Cons< H, T >> Struct Template Reference	34
5.58	$\label{lib::log::Formatter} \mbox{lib::log::Formatter} < \mbox{F} > \mbox{Struct Template Reference} $	35
5.59	lib::Frequency Class Reference	35
	5.59.1 Detailed Description	36
5.60	$\label{eq:lib::FSM} \mbox{ID, I, D, Lineage} > \mbox{Struct Template Reference} $	36
5.61	lib::FSM< ID, I, void, Lineage $>$ Struct Template Reference	36
5.62	$\label{eq:lib::FSMBase} \mbox{lib::FSMBase} < D > \mbox{Struct Template Reference} $	36
5.63	$\label{eq:lib::FSMBase} \mbox{\ensuremath{\sf lib::FSMBase}$< void} > \mbox{\ensuremath{\sf Struct}} \mbox{\ensuremath{\sf Template}} \mbox{\ensuremath{\sf Reference}} \ensuremath{\sf$	37
5.64	$\label{eq:lib::FSMMaker} \mbox{lib::FSMMaker} < \mbox{I, D, T} > \mbox{Struct Template Reference} $	37
5.65	$\label{eq:lib::FtorWrapper} \mbox{lib::FtorWrapper} < T > \mbox{Class Template Reference} $	38
	5.65.1 Detailed Description	38
5.66	$\label{lib::CreateTransitionDependencyList} \begin{subarray}{ll} {\tt lib::CreateTransitionDependencyList} < {\tt List} > ::{\tt GetDependency} < {\tt T} > - \\ {\tt Struct\ Template\ Reference} & \ldots & \ldots & \ldots & \ldots & \ldots \\ \end{subarray}$	38
5.67	$\label{eq:lib::GetElem} \mbox{IDX, List} > \mbox{Struct Template Reference} $	39

iv CONTENTS

5.68	$\label{eq:lib::GetElem} \mbox{lib::GetElem} < 0, \mbox{List} > \mbox{Struct Template Reference} $	39
5.69	$\label{lib::CreateStateList} \begin{tabular}{lllllllllllllllllllllllllllllllllll$	39
5.70	$\label{eq:lib::GetValue} \mbox{\sc Map, Key} > \mbox{\sc Struct Template Reference} \dots \dots .$	39
5.71	$\label{eq:lib::log::Handler} \mbox{lib::log::Handler} < F > \mbox{Class Template Reference} $	39
	5.71.1 Detailed Description	40
5.72	haw::HM Class Reference	40
5.73	HWAccess Class Reference	40
	5.73.1 Detailed Description	40
5.74	hw::HWAccessImpl Class Reference	41
5.75	haw::Event::ID Struct Reference	42
5.76	$\label{eq:lib::ldentity} \mbox{lib::ldentity} < T > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	42
5.77	lib::If < false, T1, T2 > Struct Template Reference	42
5.78	$\label{eq:lib::If} \mbox{lib::If} < \mbox{true, T1, T2} > \mbox{Struct Template Reference} \qquad . \qquad . \qquad . \qquad .$	42
5.79	hw::Connection::Impl Class Reference	42
5.80	$\label{eq:lib::InheritLineage} \mbox{lib::InheritLineage} < T > \mbox{Struct Template Reference} \ \ . \ \ \ \ \ . \ \ \ \ \ . \ \ \ \ . \ \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ \ \ . \ \ \ . \ \ . \ \ \ \ \ \ \ \ . \ \ \ \ \ \ \ \ \ \ \ \ \ \$	43
5.81	$\label{eq:lib::InheritLineage} \mbox{\sc Nil} > \mbox{\sc Struct Template Reference} $	43
5.82	haw::Initialization Class Reference	43
5.83	$\label{eq:lib::Int} \mbox{lib::Int} < \mbox{I} > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	43
5.84	$\label{lib::CreateTransitionDependencyList} \begin{subarray}{ll} List > ::CollectDependencies < E \\ > ::IsCorrectEvent < T > Struct\ Template\ Reference \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	44
5.85	$\label{eq:lib::lsList} \mbox{lib::lsList} < T > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	44
5.86	lib::IsList< Cons< T1, T2 >> Struct Template Reference	44
5.87	$\label{eq:lib::lsSame} \mbox{Iib::lsSame} < \mbox{T1, T2} > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	44
5.88	$\label{eq:lib::lsSame} \mbox{lib::lsSame} < \mbox{T, T} > \mbox{Struct Template Reference} $	45
5.89	$\label{eq:lib::lsSuperType} \mbox{Sub, Super} > \mbox{Struct Template Reference} \ . \ . \ . \ . \ . \ .$	45
5.90	$\label{eq:lib::Join} \mbox{List, Appendage} > \mbox{Struct Template Reference} $	45
5.91	$\label{eq:lib::Join} \mbox{lib::Join} < \mbox{Nil, Appendage} > \mbox{Struct Template Reference} \ \ . \ \ \ \ \ . \ \ \ \ \ . \ \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ \ . \ \ \ . \ \ . \ \ . \ \ . \ \ . \ \ . \ \ \ \ . \ \ \ . \ \ \ \ . \ \ \ . \ \ . \ \ \ \ \ \ \ \ \ \ \ \ \ \$	45
5.92	hw::LED Class Reference	46
	5.92.1 Detailed Description	47
5.93	$\label{eq:lib::ListAnd} \mbox{List} > \mbox{Struct Template Reference} $	47
5.94	$\label{eq:lib::ListOr} \mbox{ListOr} < \mbox{List} > \mbox{Struct Template Reference} $	47
5.95	$\label{eq:lib::ListToMap} \mbox{List} > \mbox{Struct Template Reference} \ \ . \ \$	47
5.96	$\label{eq:lib::ListToMapImpl} \mbox{IDX, List} > \mbox{Struct Template Reference} $	48

CONTENTS

5.97 lib::ListToMapImpl < IDX, Nil > Struct Template Reference	48
5.98 lib::SingletonConcurrency::SingleThreaded< T >::Lock Struct Reference	48
5.99 lib::SingletonConcurrency::MultiThreaded< T >::Lock Struct Reference	48
5.100lib::Lock< T, E, R > Class Template Reference $\dots \dots \dots$	49
5.101 lib::Lockable Class $<$ T, M $>$::Lock Struct Reference	49
5.102lib::LockableObject< T, M >::Lock Struct Reference	49
5.103lib::LockableClass< T, M $>$ Class Template Reference	49
5.104lib::LockableObject< T, M > Class Template Reference	50
5.105lib::test::TestManager::Log Struct Reference	50
5.106lib::log::Logger Class Reference	50
5.106.1 Detailed Description	51
5.107lib::log::LogLevel Class Reference	51
5.108lib::log::LogManagerImpl Class Reference	52
5.108.1 Detailed Description	52
5.109lib::log::LogRecord Class Reference	52
5.110lib::MakeList_0 Struct Reference	52
5.111 lib::MakeList_1 < A > Struct Template Reference	53
5.112lib::MakeList_2< A, B $>$ Struct Template Reference	53
5.113lib::MakeList_3< A, B, C > Struct Template Reference $\dots \dots$	53
5.114lib::MakeList_4< A, B, C, D $>$ Struct Template Reference	53
5.115lib::MakeList_5< A, B, C, D, E $>$ Struct Template Reference	53
5.116lib::MakeList_6< A, B, C, D, E, F $>$ Struct Template Reference	54
5.117lib::MakeList_7< A, B, C, D, E, F, G $>$ Struct Template Reference	54
$5.118 lib:: MakeList_8 < A,B,C,D,E,F,G,H > StructTemplateReference . \ \ .$	54
5.119lib::MakeList_9< A, B, C, D, E, F, G, H, I $>$ Struct Template Reference .	54
5.120lib::Merge $<$ F, T, List $>$ Struct Template Reference	55
5.121 lib::Merge $<$ F, T, Nil $>$ Struct Template Reference	55
5.122hw::Motor Class Reference	55
5.122.1 Detailed Description	56
5.122.2 Member Function Documentation	56
5.122.2.1 controlBelt	56
5.122.2.2 controlSwitch	56
5.123lib::RingBufferConcurrency::MultiThreaded< T > Class Template - Reference	57

vi CONTENTS

5.124lib::SingletonConcurrency::Multi1nreaded< 1 > Struct Template -	
Reference	/ د
5.125lib::Mutex Class Reference	57
5.126lib::Nil Struct Reference	58
5.127lib::lsSuperType< Sub, Super >::No Struct Reference	58
$5.128 lib:: Not < T > Struct \ Template \ Reference \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	58
5.129lib::OneParamFtor< R, T > Class Template Reference $\dots \dots \dots$	58
$5.130 lib:: One Param Ftor < R, \ void > Class \ Template \ Reference \ \dots \ \dots \ . \ . \ . \ . \ . \ . \ . \ .$	58
$5.131 lib:: One Param Ftor < void, \ T > Class \ Template \ Reference \ \dots \ \dots \ 5.131 lib:: One Param Ftor < Void, \ T > Class \ Template \ Reference \ \dots \ \dots \ 5.131 lib:: One Param Ftor < Void, \ T > Class \ Template \ Reference \ \dots \ \dots \ 5.131 lib:: One Param Ftor < Void, \ T > Class \ Template \ Reference \ \dots \ \dots \ 5.131 lib:: One Param Ftor < Void, \ T > Class \ Template \ Reference \ \dots \ Mathematical $	59
5.132lib::OPFImpl $<$ F, R, T $>$ Class Template Reference	59
5.133lib::OPFImpl $<$ F, R, void $>$ Class Template Reference	59
5.134lib::OPFImpl < F, void, T > Class Template Reference	30
$5.135 lib:: OPFImplBase < R, T > Struct \ Template \ Reference \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	30
$5.136 lib:: OPFImplBase < R, void > Struct \ Template \ Reference \\ \ \ldots \\ \ \ldots \\ \ \ \ \ \ \ \ \ \ \ \ \ \$	31
$5.137 lib:: OPFImplBase < void, T > Struct\ Template\ Reference \\ AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA$	31
5.138lib::OPFMImpl < TT, R, T > Class Template Reference $\dots \dots \dots \dots $ 6	31
5.139lib::OPFMImpl < TT, R, void > Class Template Reference	32
5.140lib::OPFMImpl < TT, void, T > Class Template Reference	32
5.141 lib::Or $<$ T1, T2 $>$ Struct Template Reference	33
5.142lib::ListOr < List >::OrFn < T1, T2 > Struct Template Reference 6 $$	33
5.143haw::Project Class Reference	63
5.144haw::Puk Class Reference	64
5.145haw::Ready Class Reference	64
5.146lib::qnx::Receiver Class Reference	65
5.147lib::test::TestManager::Registrar Struct Reference	65
5.148lib::Reverse < List > Struct Template Reference	35
$5.149 lib:: Reverse Cons < Cell > Struct \ Template \ Reference \ \dots \ \dots \ \qquad \qquad$	35
5.150lib::ReverseImpl< Done, ToDo $>$ Struct Template Reference 6	35
5.151lib::ReverseImpl< Done, Nil $>$ Struct Template Reference	36
5.152 lib:: RingBuffer < T,N,ThreadingPolicy > ClassTemplateReference...60 decreases a constant of the constant	36
5.153lib::Rule< O, E, D, A > Struct Template Reference	36
5.154haw::Running Class Reference	37
5.155lib::test::TestManager::Selector Struct Reference	37
5.156lib::Semaphore Class Reference	37

CONTENTS vii

5.157hw::Sensor Class Reference	68
5.157.1 Detailed Description	69
5.157.2 Member Function Documentation	69
5.157.2.1 entering	69
5.157.2.2 handlePulse	69
5.158haw::SensorEvent Class Reference	69
5.159haw::SensorEvent::Sensors Struct Reference	70
$5.160 lib:: Set if y < List > Struct \ Template \ Reference \\ \ \ldots \\ \ \ldots$	70
5.161 lib::SetifyImpl < Done, ToDo > Struct Template Reference	70
5.162lib::SetifyImpl< Done, Nil $>$ Struct Template Reference	70
5.163lib::RingBufferConcurrency::SingleThreaded< T > Class Template - Reference	71
5.164lib::SingletonConcurrency::SingleThreaded< T > Struct Template - Reference	71
5.165lib::Singleton< T, TM, P > Class Template Reference	71
5.165.1 Detailed Description	71
5.165.2 Member Function Documentation	72
5.165.2.1 instance	72
5.166lib::SmartPtr< T > Class Template Reference	72
5.166.1 Detailed Description	73
5.167hw::Motor::Speed Struct Reference	73
5.168lib::Speed Class Reference	73
5.169hw::Motor::State Struct Reference	73
5.170haw::State Class Reference	74
5.171 lib::log::StreamHandler Class Reference	74
5.171.1 Detailed Description	75
5.172lib::test::TestManager Class Reference	75
5.173lib::Thread Class Reference	76
5.173.1 Detailed Description	76
5.173.2 Constructor & Destructor Documentation	76
5.173.2.1 Thread	76
5.173.2.2 Thread	77
5.173.2.3 ~Thread	77
5.173.3 Member Function Documentation	77

viii CONTENTS

5.173.3.2 joinable
5.173.3.3 run
5.174lib::ThreadManagerImpl Class Reference
5.175lib::Time Class Reference
5.175.1 Detailed Description
5.176lib::Timer Class Reference
5.176.1 Detailed Description
5.176.2 Member Function Documentation
5.176.2.1 active
5.176.2.2 deactivateAll
5.176.2.3 delta
5.176.2.4 elapsed
5.176.2.5 sync
5.176.2.6 timestamp
5.177lib::TimerPoolImpl Class Reference
5.178lib::CreateTransitionMap< List >::Transform< T > Struct Template -
·
·
Reference
Reference
$\begin{tabular}{ll} Reference & $
$\begin{tabular}{ll} Reference & . & . & . & . & . & . & . & . & . & $
Reference81 5.179 lib::TransImpl< E, D, L, S, T > Struct Template Reference82 5.180 lib::TransImpl< E, D, Nil, S, T > Struct Template Reference82 5.181 lib::TransImpl< E, void, L, S, T > Struct Template Reference83 5.182 lib::TransImpl< E, void, Nil, S, T > Struct Template Reference84 5.183 lib::Transition< O, E, D > Struct Template Reference84
Reference81 5.179 lib::TransImpl< E, D, L, S, T > Struct Template Reference82 5.180 lib::TransImpl< E, D, Nil, S, T > Struct Template Reference82 5.181 lib::TransImpl< E, void, L, S, T > Struct Template Reference83 5.182 lib::TransImpl< E, void, Nil, S, T > Struct Template Reference84 5.183 lib::Transition0, E, D > Struct Template Reference84 5.184 lib::TryCall_apply< T, E, D > Struct Template Reference84
Reference81 5.179 lib::TransImpl< E, D, L, S, T > Struct Template Reference82 5.180 lib::TransImpl< E, D, Nil, S, T > Struct Template Reference82 5.181 lib::TransImpl< E, void, L, S, T > Struct Template Reference83 5.182 lib::TransImpl< E, void, Nil, S, T > Struct Template Reference84 5.183 lib::TransitionO, E, D > Struct Template Reference84 5.184 lib::TryCall_apply< T, E, D > Struct Template Reference84 5.185 lib::TryCall_apply< T, E, void > Struct Template Reference85
Reference81 5.179 lib::TransImpl< E, D, L, S, T > Struct Template Reference82 5.180 lib::TransImpl< E, D, Nil, S, T > Struct Template Reference82 5.181 lib::TransImpl< E, void, L, S, T > Struct Template Reference83 5.182 lib::TransImpl< E, void, Nil, S, T > Struct Template Reference84 5.183 lib::Transition< O, E, D > Struct Template Reference84 5.184 lib::TryCall_apply< T, E, D > Struct Template Reference84 5.185 lib::TryCall_apply< T, E, void > Struct Template Reference85 5.186 haw::Puk::Type Struct Reference85
Reference81 5.179 lib::TransImpl< E, D, L, S, T > Struct Template Reference82 5.180 lib::TransImpl< E, D, Nil, S, T > Struct Template Reference82 5.181 lib::TransImpl< E, void, L, S, T > Struct Template Reference83 5.182 lib::TransImpl< E, void, Nil, S, T > Struct Template Reference84 5.183 lib::TransitionO, E, D > Struct Template Reference84 5.184 lib::TryCall_apply< T, E, D > Struct Template Reference84 5.185 lib::TryCall_apply< T, E, void > Struct Template Reference85 5.186 haw::Puk::Type Struct Reference85 5.187 lib::test::UnitTest Class Reference85
Reference81 5.179 lib::TransImpl< E, D, L, S, T > Struct Template Reference82 5.180 lib::TransImpl< E, D, Nil, S, T > Struct Template Reference82 5.181 lib::TransImpl< E, void, L, S, T > Struct Template Reference83 5.182 lib::TransImpl< E, void, Nil, S, T > Struct Template Reference84 5.183 lib::TransitionO, E, D > Struct Template Reference84 5.184 lib::TryCall_apply< T, E, D > Struct Template Reference84 5.185 lib::TryCall_apply< T, E, void > Struct Template Reference85 5.186 haw::Puk::Type Struct Reference85 5.187 lib::test::UnitTest Class Reference85 5.188 lib::Value< T, I > Struct Template Reference86
Reference81 5.179 lib::TransImpI < E, D, L, S, T > Struct Template Reference82 5.180 lib::TransImpI < E, D, Nil, S, T > Struct Template Reference82 5.181 lib::TransImpI < E, void, L, S, T > Struct Template Reference83 5.182 lib::TransImpI < E, void, Nil, S, T > Struct Template Reference84 5.183 lib::Transition < O, E, D > Struct Template Reference84 5.184 lib::TryCall_apply < T, E, D > Struct Template Reference84 5.185 lib::TryCall_apply < T, E, void > Struct Template Reference85 5.186 haw::Puk::Type Struct Reference85 5.187 lib::test::UnitTest Class Reference85 5.188 lib::Value < T, I > Struct Template Reference86 5.189 lib::Valueldentity < Bool < I > > Struct Template Reference86

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 >
$lib::ListAnd < List > ::AndFn < T1, 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
$\label{lib::BasicFunctorImpl} \mbox{\it Iib::BasicFunctorImpl} < \mbox{\it F} > \dots $
lib::Caar $<$ T $>$
lib::Cadr $<$ T $>$
haw::Calibrator
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
haw::Config
lib:-gpv:-Connection 26

lib::Cons $<$ H, T $>$	27
	27
lib::ConstructFSMLineage <t></t>	27
lib::ConstructFSMLineage $<$ Cons $<$ T, Nil $>>$	27
· · · · · · · · · · · · · · · · · · ·	28
	28
lib::CreateStateList < List >	28
${\it lib::} Create Transition Dependency List < List > \dots $	28
lib::CreateTransitionMap< List >	29
$\label{lib::FSMMaker} \mbox{Iib::FSMMaker} < \mbox{I, D, T} > :: \mbox{CreateTransitionTree} < \mbox{TT} > $	29
lib::Data	29
lib::Decay< T >	30
lib::Decay< const T >	30
lib::Decay< const volatile T >	30
lib::Decay< T & >	30
lib::Decay< volatile T >	31
hw::Motor::Direction	31
lib::RingBufferConcurrency::SingleThreaded< T >::EmptyLock	32
lib::RingBufferConcurrency::MultiThreaded< T >::EmptyLock	32
haw::Event	32
haw::SensorEvent	69
	33
	33
	34
	34
	34
	34
	35
	36
	36
	36
	82
·	
	37
	84
	37
	38
$\label{lib::CreateTransitionDependencyList} \mbox{List} > \mbox{::GetDependency} < \mbox{T} > \mbox{:}$	38
lib::GetElem< IDX, List >	39
, , , , , , , , , , , , , , , , , , ,	39
$\label{lib::CreateStateList} \mbox{List} > \mbox{::GetStateFromTransition} < \mbox{T} > $	39
lib::GetValue < Map, Key >	39
haw::HM	40
	40
haw::Event::ID	42
-,	42
lib:: $lf < false, T1, T2 > \dots$	42
lib::lf< true, T1, T2 >	42
·	42
lib::InheritLineage< T >	43

$\label{lib::InheritLineage} \mbox{lib::InheritLineage} < \mbox{Nil} > \ldots $	43
$lib::lsList < T > \dots \dots$	44
$\label{eq:lib::lsList} \mbox{lib::lsList} < \mbox{Cons} < \mbox{T1}, \mbox{T2} >> \ \dots $	44
lib::IsSame < T1, T2 >	44
lib::IsSame $<$ DO(Car $<$ T $>$), E $>$	44
lib::CreateTransitionDependencyList< List >::CollectDependencies< E	
>::IsCorrectEvent< T >	44
$lib:: lsSame < T, T > \dots \dots$	45
lib::IsSuperType < Sub, Super >	45
lib::Join< List, Appendage >	45
lib::Join< Nil, Appendage >	45
lib::ListAnd< List >	47
lib::ListOr< List >	47
lib::ListToMap < List >	47
lib::ListToMapImpl < IDX, List >	48 48
lib::ListToMapImpl < IDX, Nil >	48
lib::SingletonConcurrency::MultiThreaded< T >::Lock	48
lib::Lock< T, E, R >	49
lib::LockableClass< T, M >::Lock	49
lib::LockableObject< T, M >::Lock	49
lib::LockableClass< T, M >	49
lib::LockableClass< T >	49
lib::SingletonConcurrency::MultiThreaded< T >	57
lib::LockableObject< T, M >	50
lib::LockableObject< Actuator >	50
hw::Actuator	17
lib::LockableObject< CleanupUtility >	50
	24
lib::CleanupUtility	
lib::LockableObject< Connection >	50
hw::Connection	26
lib::LockableObject< DefaultFormatter >	50
lib::log::DefaultFormatter	
$\label{lib::LockableObject} \mbox{lib::LockableObject} < \mbox{HWAccessImpl} > . \ . \ . \ . \ . \ . \ . \ . \ . \ .$	50
hw::HWAccessImpl	41
lib::LockableObject< LED >	50
hw::LED	46
lib::LockableObject< Logger >	50
lib::log::Logger	50
lib::LockableObject< LogManagerImpl >	50
	52
lib::log::LogManagerImpl	
lib::LockableObject < Motor >	50
hw::Motor	55
lib::LockableObject< Sensor >	50
hw::Sensor	68
lib::LockableObject< SmartPtr< T >>	50

lib::SmartPtr< T >
lib::LockableObject < StreamHandler >
lib::log::StreamHandler
lib::LockableObject< ThreadManagerImpl >
lib::ThreadManagerImpl
lib::test::TestManager::Log
lib::log::LogLevel 51 lib::log::LogRecord 52
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 >
lib::Merge < F, T, List >
lib::Merge < F, T, Nil >
lib::RingBufferConcurrency::MultiThreaded< T >
lib::Mutex
lib::Nil
lib::IsSuperType< Sub, Super >::No
lib::Not< T >
lib::OneParamFtor $< R, T > \dots $ 58
lib::OneParamFtor< R, void >
lib::OneParamFtor< void, T >
lib::OPFImplBase $< R, T > \dots \dots$
lib::OPFImpl $<$ F, R, T $>$
lib::OPFMImpl $<$ TT, R, T $>$ 61
$\label{lib::OPFImplBase} \mbox{lib::OPFImplBase} < R, \mbox{void} > \dots $
lib::OPFImpl< F, R, void >
lib::OPFMImpl < TT, R, void >
lib::OPFImplBase < void, T >
lib::OPFImpl< F, void, T >
lib::OPFMImpl< TT, void, T >
lib::Or< T1, T2 >
lib::ListOr< List >::OrFn< T1, T2 >
haw::Project
haw::Puk
lib::qnx::Receiver
lib::test::TestManager::Registrar
lib::Reverse< List >
lib::ReverseCons< Cell >
lib::ReverseImpl< Done, ToDo >
lib::ReverseImpl< Done, Nil >
lib::RingBuffer< T, N, ThreadingPolicy >

lib::test::TestManager::Selector
lib::Semaphore
haw::SensorEvent::Sensors
lib::Setify< List >
$\label{eq:lib::SetifyImpl} \mbox{Lib::SetifyImpl} < \mbox{Done, ToDo} > \mbox{.} \$
$\label{eq:lib::SetifyImpl} \mbox{Lib::SetifyImpl} < \mbox{Done, Nil} > \dots $
lib:: RingBufferConcurrency:: SingleThreaded < T >
$lib:: Singleton Concurrency:: Single Threaded < T > \ \dots \ \dots \ \ 71$
$lib:: Singleton < T, TM, P > \ldots \ldots$
hw::Motor::Speed
lib::Speed
hw::Motor::State
haw::State
haw::Initialization
haw::Ready
haw::Running
lib::test::TestManager
lib::Thread
lib::Time
lib::Timer
lib::TimerPoolImpl
$\label{lib::CreateTransitionMap} \mbox{List} > :: \mbox{Transform} < T > . \ . \ . \ . \ . \ . \ . \ . \ . \ .$
lib::TransImpl $<$ E, D, L, S, T $>$ 82
$lib:: TransImpl < E, void, DO(Cdr < L >), S, T > \dots \dots \dots \dots \dots \\ 82$
$\label{eq:lib::TransImpl} \mbox{lib::TransImpl} < \mbox{E, void, L, S, T} > \dots $
lib::Transition $<$ O, E, D $>$
lib::Rule < O, E, D, A >
lib::TryCall_apply< T, E, D >
lib::TryCall_apply< T, E, void >
haw::Puk::Type
lib::test::UnitTest
lib::Value < T, I >
lib::Value < bool, V >
lib::Bool< V >
lib::Value< int, I >
lib::Int< I >
lib::ValueIdentity < Bool < I >>
lib::ValueIdentity < Int < I > >
lib::ValueIdentity < Value < T, I >>
lib::IsSuperType < Sub. Super >::Yes

Chapter 3

Class Index

3.1 Class List

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

TwActuator	
Singular access point to actuators	7
lib::And $<$ T1, T2 $>$	8
lib::ListAnd < List > ::AndFn < T1, T2 > 1	8
$lib::Apply < F, T > \dots \dots$	8
$lib::Apply < F, Nil > \ \ \ldots \ \ 1$	8
$lib::Array < T,N > \ldots \qquad \qquad 1$	9
lib::Array $<$ T, 0 $>$	C
lib::log::BaseFilter	
lib::log::BaseFormatter	C
lib::log::BaseHandler	C
lib::BasicFunctor	1
lib::BasicFunctorImpl< F >	
Basic functor encapsulating anything callable that takes no argu-	
ments	1
lib::Bool < V >	2
lib::Caar $<$ T $>$	2
lib::Cadr $<$ T $>$	
haw::Calibrator	2
lib::Car $<$ T $>$	
lib::Cdar $<$ T $>$	3
lib::Cddr $<$ T $>$	
lib::Cdr $<$ T $>$	3
lib::qnx::Channel	
$lib:: TryCall_apply < T, E, D > :: Check < typename, > 2$	
$lib:: TryCall_apply < T, \ E, \ void > :: Check < typename, > 2$	4
lib::CleanupUtility	
Utility for controlling the lifetime of static objects, i.e	4
lib::CreateTransitionDependencyList < List >::CollectDependencies < E > 2	5

lib::Condition	25
haw::Config	26
lib::qnx::Connection	26
hw::Connection	
lib::Cons $<$ H, T $>$	27
lib::ConsFn< T1, T2 >	27
lib::ConstructFSMLineage< T >	27
lib::ConstructFSMLineage< Cons< T, Nil >>	27
lib::Contains < List, T >	28
lib::Contains $<$ Nil, $\overset{\cdot}{T}>$	28
lib::CreateStateList >	28
lib::CreateTransitionDependencyList< List >	28
lib::CreateTransitionMap< List >	29
lib::FSMMaker< I, D, T >::CreateTransitionTree< TT >	29
lib::Data	29
lib::Decay< T >	30
lib::Decay < const T >	30
lib::Decay < const volatile T >	30
lib::Decay < T & >	30
lib::Decay < volatile T >	31
lib::log::DefaultFormatter	31
Default formatter that lists all information of the passed LogRecord .	31
hw::Motor::Direction	31
lib::RingBufferConcurrency::SingleThreaded< T >::EmptyLock	32
lib::RingBufferConcurrency::MultiThreaded< T >::EmptyLock	32
haw::Event	32
lib::RingBufferConcurrency::SingleThreaded< T >::FillLock	33
lib::RingBufferConcurrency::MultiThreaded< T >::FillLock	33
lib::log::Filter< F >	00
Filter template that accepts functors	
lib::Filter< F, List >	
lib::Filter< F, Nil >	
lib::Flatten< T >	
lib::Flatten< Cons< H, T >>	34
lib::log::Formatter< F >	35
lib::Frequency	
Convenience class that allows calculation of a signal's period length	
through its frequency	35
lib::FSM< ID, I, D, Lineage >	36
lib::FSM< ID, I, void, Lineage >	36
lib::FSMBase < D >	36
${\sf lib::FSMBase} {<} {\sf void} {>} \dots $	37
$lib:: FSMMaker < I, D, T > \dots \dots$	37
lib::FtorWrapper< T >	
A functor that calls an object's member function	38
$\label{lib::CreateTransitionDependencyList} \textbf{List} > :: \textbf{GetDependency} < \textbf{T} > \ . \ . \ . \ . \ .$	38
${\sf lib::} {\sf GetElem} {<} {\sf IDX, List} {>} \dots $	39
${\sf lib::} {\sf GetElem} {<} {\sf 0}, {\sf List} {>} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	39
$\label{lib::CreateStateList} \mbox{List} > \mbox{::GetStateFromTransition} < \mbox{T} > $	39
${\sf lib::} {\sf GetValue} {<} {\sf Map,} {\sf Key} {>} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	39

3.1 Class List

lib::log::Handler< F >	
•	39
haw::HM	40
Interface for direct hardware access. The HWAccess singleton offers	
	40
·	41
	42
	42
•	42
	42
	42
	43
	43
	43
	43
lib::CreateTransitionDependencyList< List >::CollectDependencies< E >::-	
IsCorrectEvent < T >	44
lib::lsList< T >	44
lib::lsList< Cons< T1, T2 >>	44
lib::lsSame $<$ T1, T2 $>$	44
$lib::lsSame < T, T > \dots \dots$	45
${\sf lib::} {\sf lsSuperType} {<} \ {\sf Sub}, \ {\sf Super} {>} \ \ldots \ldots \ldots \ldots \ldots$	45
, 11	45
${\sf lib::Join} {<} {\sf Nil}, {\sf Appendage} {>} \dots $	45
hw::LED	
Allows access to LEDs. Offers an interface for accessing LEDs on	
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking function-	
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer	46
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer	47
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer	47 47
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer	47 47 47
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List>	47 47 47 48
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List> lib::ListOr< List> lib::ListToMap< List> lib::ListToMapImpl< IDX, List> lib::ListToMapImpl< IDX, Nil>	47 47 47 48 48
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List >	47 47 47 48 48 48
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List >	47 47 48 48 48 48
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List >	47 47 47 48 48 48 48 49
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List >	47 47 47 48 48 48 48 49
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List> lib::ListToMap< List> lib::ListToMap mp < IDX, List> lib::ListToMapImp < IDX, Nil> lib::SingletonConcurrency::SingleThreaded< T>::Lock lib::SingletonConcurrency::MultiThreaded< T>::Lock lib::LockableClass< T, M>::Lock	47 47 48 48 48 49 49
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List >	47 47 47 48 48 48 49 49 49
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List >	47 47 47 48 48 48 49 49 49 50
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List >	47 47 47 48 48 48 49 49 49
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer 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 lib::Lock< T, E, R > lib::LockableClass< T, M >::Lock lib::LockableClass< T, M >::Lock lib::LockableClass< T, M > lib::LockableObject< T, M > lib::L	47 47 48 48 48 49 49 49 50 50
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer 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 lib::Lock< T, E, R > lib::LockableClass< T, M >::Lock lib::LockableClass< T, M > lib::LockableClass	47 47 48 48 48 49 49 49 50 50
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd	47 47 48 48 48 49 49 49 50 50
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List > lib::ListToMap< List > lib::ListToMap IDX, List > lib::ListToMapImp IDX, Nil > lib::SingletonConcurrency::SingleThreaded< T >::Lock lib::SingletonConcurrency::MultiThreaded< T >::Lock lib::Lock< T, E, R > lib::LockableClass< T, M >::Lock lib::LockableObject< T, M > lib::Lo	47 47 48 48 48 49 49 50 50 51
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List > lib::ListOr< List > lib::ListToMap< List > lib::ListToMapImpI< IDX, List > lib::ListToMapImpI< IDX, Nil > lib::SingletonConcurrency::SingleThreaded< T >::Lock lib::SingletonConcurrency::MultiThreaded< T >::Lock lib::Lock /trick to the content of	47 47 48 48 48 49 49 50 50 51
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd < List >	47 47 48 48 48 49 49 50 50 51 52
Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer lib::ListAnd< List >	47 47 48 48 48 49 49 50 50 51

lib::MakeList_2< A, B >	53
lib::MakeList_3 $<$ A, B, C $>$	53
lib::MakeList_4 $<$ A, B, C, D $>$	53
lib::MakeList_5 $<$ A, B, C, D, E $>$	53
lib::MakeList_6 $<$ A, B, C, D, E, F $>$	54
lib::MakeList_7< A, B, C, D, E, F, G $>$	54
lib::MakeList_8 $<$ A, B, C, D, E, F, G, H $>$	54
lib::MakeList_9 $<$ A, B, C, D, E, F, G, H, I $>$	54
lib::Merge < F, T, List >	55
lib::Merge $<$ F, T, Nil $>$	55
hw::Motor	
Interface for controlling the motors. Client interface for controlling the	
conveyor belt and electromagnetic switch. Defines different states	
that the motors can enter	
lib::RingBufferConcurrency::MultiThreaded< T >	
lib::SingletonConcurrency::MultiThreaded< T >	
lib::Mutex	57
lib::Nil	
lib::IsSuperType< Sub, Super >::No	
lib::Not< T >	
lib::OneParamFtor< R, T >	
lib::OneParamFtor< R, void >	58
lib::OneParamFtor< void, T >	
$\label{eq:lib::OPFImpl} \mbox{lib::OPFImpl} < \mbox{F, R, T} > \dots $	
$\label{eq:lib::OPFImpl} \mbox{lib::OPFImpl} < \mbox{F, R, void} > $	
$\label{eq:lib::OPFImpl} \mbox{lib::OPFImpl} < \mbox{F, void, T} > \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \dots \ \ \ \$	
lib::OPFImplBase $<$ R, T $>$	
lib::OPFImplBase< R, void >	
$\label{eq:lib::OPFImplBase} \mbox{lib::OPFImplBase} < \mbox{void}, \mbox{T} > \dots $	
$\label{eq:lib::OPFMImpl} \mbox{lib::OPFMImpl} < \mbox{TT, R, T} > \dots $	
$\label{eq:lib::OPFMImpl} \mbox{Iib::OPFMImpl} < \mbox{TT, R, void} > \ \ \ldots \ \ \ldots \ \ \ \ldots \ \ \ \ \ \ \ \ \$	
$\label{eq:continuous} \mbox{lib::OPFMImpl}{<} \mbox{TT, void, T}{>} \dots $	
lib::Or< T1, T2 >	
lib::ListOr< List >::OrFn< T1, T2 >	
haw::Project	
haw::Puk	64
haw::Ready	64
lib::qnx::Receiver	65
lib::test::TestManager::Registrar	65
lib::Reverse < List >	65
lib::ReverseCons< Cell >	65
lib::ReverseImpl< Done, ToDo >	65
lib::ReverseImpl< Done, Nil >	66
lib::RingBuffer< T, N, ThreadingPolicy >	66
lib::Rule $<$ O, E, D, A $>$	66
haw::Running	67
lib::test::TestManager::Selector	67
lib::Semaphore	67

3.1 Class List

hw::Sensor	
Interface for checking the state of the Sensors. The sensors can be	
checked for different states. For example if the sensor is still active	
or the State of the Sensor changed	68
haw::SensorEvent	69
haw::SensorEvent::Sensors	70
lib::Setify< List >	70
lib::SetifyImpl < Done, ToDo >	70
lib::SetifyImpl < Done, Nil >	70
${\it lib::} {\it RingBufferConcurrency::} {\it SingleThreaded} {\it < T>} \ldots \ldots \ldots$	71
${\it lib::} Singleton Concurrency:: Single Threaded < T > \dots \dots \dots \dots$	71
lib::Singleton< T, TM, P >	
Template for convenient Singleton creation	71
lib::SmartPtr< T >	
Smart pointer class for automatic life time management	72
hw::Motor::Speed	73
lib::Speed	73
hw::Motor::State	73
haw::State	74
lib::log::StreamHandler	
Handler compatible functor that writes its LogRecord to an std-	
::stream instance	74
lib::test::TestManager	75
lib::Thread	
Encapsulates the most important features of a thread	76
lib::ThreadManagerImpl	78
lib::Time	
Data class representing a timeframe with microsecond accuracy	78
lib::Timer	
Timer that allows scheduling of functors	79
lib::TimerPoolImpl	81
lib::CreateTransitionMap< List >::Transform< T >	81
lib::TransImpl $<$ E, D, L, S, T $>$	82
lib::TransImpl $<$ E, D, Nil, S, T $>$	82
$lib:: TransImpl < E, void, L, S, T > \dots \dots \dots \dots \dots \dots$	83
$lib:: TransImpl < E, void, Nil, S, T > \dots \dots \dots \dots \dots \dots$	84
lib::Transition $<$ O, E, D $>$	84
lib::TryCall_apply< T, E, D >	84
lib::TryCall_apply< T, E, void >	85
haw::Puk::Type	85
lib::test::UnitTest	85
$lib::Value < T, I > \ \ldots \ldots \ldots \ldots \ldots$	86
$\label{eq:lib::ValueIdentity} \mbox{Bool} < \mbox{I} >> \dots \dots \dots \dots \dots \dots \dots \dots \dots $	86
$\label{lib::ValueIdentity} \mbox{Int} < \mbox{Int} < \mbox{I} >> \dots \dots \dots \dots \dots \dots \dots \dots \dots $	86
$lib:: Value Identity < Value < T, I >> \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ $	86
lib::IsSuperType < Sub. Super >::Yes	86

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

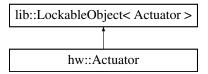
Class Documentation

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

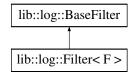
template<typename T, std::size $_{-}$ t N> class lib::Array< T, N>

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:

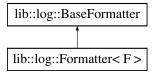


Public Member Functions

• virtual bool accept (const LogRecord &)=0

5.9 lib::log::BaseFormatter Struct Reference

Inheritance diagram for lib::log::BaseFormatter:

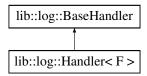


Public Member Functions

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

5.10 lib::log::BaseHandler Struct Reference

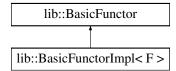
Inheritance diagram for lib::log::BaseHandler:



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

5.11 lib::BasicFunctor Struct Reference

Inheritance diagram for lib::BasicFunctor:



Public Member Functions

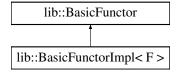
• virtual void **operator()** ()=0

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() ()

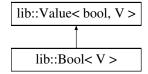
5.12.1 Detailed Description

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 haw::Calibrator Class Reference

Classes

· class Measurer

- Calibrator (Project &p)
- void process (const SensorEvent &e)
- void tick ()
- · bool done () const

5.17 lib::Car < T > Struct Template Reference

Public Types

• typedef T::Head Type

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

5.18 lib::Cdar < T > Struct Template Reference

Public Member Functions

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

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

5.19 lib::Cddr< T > Struct Template Reference

Public Member Functions

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

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

5.20 lib::Cdr < T > Struct Template Reference

Public Types

• typedef T::Tail Type

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

5.21 lib::qnx::Channel Class Reference

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

Friends

· class Receiver

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

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

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

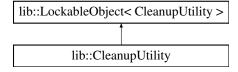
 $\label{template} $$ \textbf{T}$, typename E>template< typename, void(*)(const E \&)> struct lib::TryCall-apply< T, E, void>::Check< typename, >$

5.24 lib::CleanupUtility Class Reference

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

#include <CleanupUtility.h>

Inheritance diagram for lib::CleanupUtility:



Classes

• struct Compare

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

Static Public Member Functions

• static CleanupUtility & instance ()

Static Public Attributes

• static const size_t **DEFAULT_PRIORITY** = 10

5.24.1 **Detailed Description**

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

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

5.25 lib::CreateTransitionDependencyList< >::Collect-List **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< ${\bf List>::CollectDependencies}{<{\bf E}>}$

5.26 lib::Condition Class Reference

Public Member Functions

- void wait ()
- bool wait (timespec *)

- void broadcast ()
- · void lock ()
- void unlock ()

5.27 haw::Config Class Reference

Public Member Functions

- Config (const std::string &)
- · void save () const
- void **setHM** (uint16_t min, uint16_t max)
- void setTimes (lib::Time slow, lib::Time fast, lib::Time hm, lib::Time puk)
- uint16_t getMin () const
- uint16_t getMax () const
- lib::Time getSlow () const
- lib::Time getFast () const
- lib::Time getHM () const
- lib::Time getPuk () const

5.28 lib::qnx::Connection Class Reference

Public Member Functions

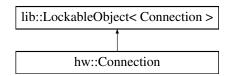
- void send (Data_ptr) const
- void close ()
- · bool open () const
- int raw () const

Friends

· class Channel

5.29 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

5.30 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>

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

Public Types

typedef Cons< T1, T2 > Type

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

5.32 lib::ConstructFSMLineage < T > Struct Template Reference

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

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

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

5.34 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>

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

Static Public Attributes

• static const bool value = false

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

5.36 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>

5.37 lib::CreateTransitionDependencyList > Struct - Template Reference

Classes

- struct CollectDependencies
- struct GetDependency

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

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

5.38 lib::CreateTransitionMap < List > Struct Template Reference

Classes

struct Transform

Public Member Functions

typedef DO (Apply < Transform, List >) Type

 $template {<} typename\ List {>}\ struct\ lib:: Create Transition Map {<}\ List {>}$

5.39 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>

5.40 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)

5.41 lib::Decay < T > Struct Template Reference

Public Types

• typedef T Type

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

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

Public Types

typedef Decay< T >::Type Type

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

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

Public Types

typedef Decay< T >::Type Type

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

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

Public Types

typedef Decay< T >::Type Type

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

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

Public Types

typedef Decay
 Type Type

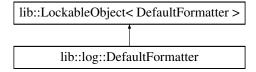
template<typename T> struct lib::Decay< volatile T>

5.46 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)

5.46.1 Detailed Description

Default formatter that lists all information of the passed LogRecord.

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

5.47 hw::Motor::Direction Struct Reference

Static Public Attributes

• static const pid t NONE = 0x00

no pin
 static const pid_t RIGHT = 0x01
 port A pin 0
 static const pid_t LEFT = 0x02
 port A pin 1

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

Public Member Functions

EmptyLock (SingleThreaded< T > *)

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

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

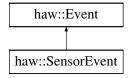
Public Member Functions

EmptyLock (MultiThreaded< T > *t)

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

5.50 haw::Event Class Reference

Inheritance diagram for haw::Event:



Classes

• struct ID

Public Types

• typedef uint32_t event_id_t

• virtual event_id_t id () const =0

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

Public Member Functions

FillLock (SingleThreaded< T > *)

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

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

Public Member Functions

FillLock (MultiThreaded< T > *t)

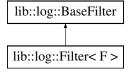
template < typename T > class lib::RingBufferConcurrency::MultiThreaded < T >::FillLock

5.53 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)

5.53.1 Detailed Description

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

Filter template that accepts functors.

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

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

Public Types

• typedef lf< F< DO(Car< List >)>::value, ldentity< Cons< DO(Car < List >), Rest > >, ldentity < 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>

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

Public Types

typedef Nil Type

 ${\tt template}{<}{\tt template}{<}{\tt typename} > {\tt class} \; {\tt F}{>} \; {\tt struct} \; {\tt lib::Filter}{<} \; {\tt F, \, Nil} >$

5.56 lib::Flatten < T > Struct Template Reference

Public Types

· typedef T Type

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

5.57 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

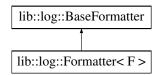
Public Member Functions

• typedef **DO** (Flatten< T >) Rest

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

5.58 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)

 $template {<} typename \ F{>} \ struct \ lib::log::Formatter {<} \ F{>}$

5.59 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)

5.59.1 Detailed Description

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

5.60 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 >

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

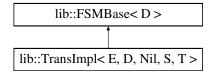
Public Types

typedef TryCall_enter< I, void > EnterFunction

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

5.62 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)

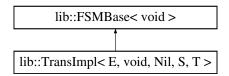
Public Attributes

- int state_
- D data_

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

5.63 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 >

5.64 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

- 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>

5.65 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() ()

5.65.1 Detailed Description

 $template {<} typename \ T {>} class \ lib:: Ftor Wrapper {<} \ T {>}$

A functor that calls an object's member function.

5.66 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

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

5.67 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>

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

Public Member Functions

typedef DO (Car< List >) Type

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

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

Public Member Functions

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

 $\label{template} $$ \textbf{typename List} > \textbf{template} < \textbf{typename T} > \textbf{struct lib::CreateStateList} < \textbf{List} > :: \textbf{GetState-FromTransition} < \textbf{T} > $$$

5.70 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

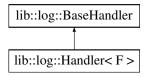
template<typename Map, typename Key> struct lib::GetValue< Map, Key>

5.71 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)

5.71.1 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.

5.72 haw::HM Class Reference

Public Member Functions

- void calibrate (uint16_t min, uint16_t max)
- void start ()
- · void tick ()
- uint32_t **stop** ()
- bool running () const

5.73 HWAccess Class Reference

Interface for direct hardware access. The HWAccess singleton offers read/write operations to the three ports of the hw unit.

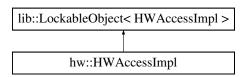
```
#include <HWAccess.h>
```

5.73.1 Detailed Description

Interface for direct hardware access. The HWAccess singleton offers read/write operations to the three ports of the hw unit.

5.74 hw::HWAccessImpl Class Reference

Inheritance diagram for hw::HWAccessImpl:



Public Types

- typedef lib::LockableObject < HWAccessImpl > Super
- typedef lib::Singleton < HWAccessImpl, lib::SingletonConcurrency::Multi-Threaded > SingletonInst
- typedef Super::Lock Lock
- typedef uint16_t port_t
- typedef uint8_t pin_t

Public Member Functions

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

Static Public Attributes

- static const port_t **DIO_BASE** = 0x300
- static const port_t AIO_BASE = 0x320
- static const port_t PORT_A = DIO_BASE + 0
- static const port_t PORT_B = DIO_BASE + 1
- static const port_t **PORT_C** = DIO_BASE + 2
- static const port t DIO IRQ CHECK = DIO BASE + 0x18
- static const port_t DIO_IRQ_RESET = DIO_BASE + 0x0f
- static const port_t **DIO_IRQ_MASK** = DIO_BASE + 0x0b
- static const port_t **DIO_IRQ** = 11
- static const port t AIO LOW = AIO BASE + 2
- static const port_t AIO_HIGH = AIO_BASE + 3
- static const port_t AIO_CONVERT = AIO_BASE + 2
- static const port_t AIO_START_CONVERSION = 0x10

5.75 haw::Event::ID Struct Reference

Static Public Attributes

• static const event_id_t SENSOR = 1

5.76 lib::Identity < T > Struct Template Reference

Public Types

• typedef T Type

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

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

Public Types

• typedef T2::Type Type

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

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

Public Types

• typedef T1::Type Type

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

5.79 hw::Connection::Impl Class Reference

Classes

- struct DoneRunning
- struct Packet

Friends

· class Connection

5.80 lib::InheritLineage < T > Struct Template Reference

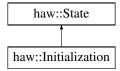
template < typename T> struct lib::InheritLineage < T>

5.81 lib::InheritLineage < Nil > Struct Template Reference

 ${\tt template}{<>} {\tt struct~lib::InheritLineage}{<} {\tt Nil}{>}$

5.82 haw::Initialization Class Reference

Inheritance diagram for haw::Initialization:

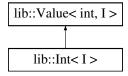


Public Member Functions

- Initialization (Project &p)
- virtual void enter ()
- virtual void exit ()
- virtual void **process** (const Event &)
- virtual void execute ()

5.83 lib::Int < I > Struct Template Reference

Inheritance diagram for lib::Int< l >:



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

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

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

lib::IsSame< DO(Car< T >), E >

lib:: Create Transition Dependency List < List > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < T > :: Collect Dependencies < E > :: Is Correct Event < E >

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

5.85 lib::lsList < T > Struct Template Reference

Static Public Attributes

• static const bool value = false

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

5.86 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 > >

5.87 lib::lsSame < T1, T2 > Struct Template Reference

Static Public Attributes

• static const bool value = false

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

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

Static Public Attributes

• static const bool value = true

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

5.89 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)

 ${\it template}{<}{\it typename~Sub,~typename~Super}{>}{\it struct~lib::}{\it lsSuperType}{<}{\it Sub,~Super}{>}$

5.90 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 >

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

Public Types

• typedef Appendage Type

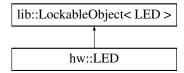
template<typename Appendage> struct lib::Join< Nil, Appendage>

5.92 hw::LED Class Reference

Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer.

```
#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

5.92.1 Detailed Description

Allows access to LEDs. Offers an interface for accessing LEDs on the hw unit. LEDs are defined as Pins. Implements blinking functionality via lib::Timer.

5.93 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 >

5.94 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>

5.95 lib::ListToMap < List > Struct Template Reference

Public Member Functions

typedef DO (ListToMapImpl < 0, List >) Type

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

5.96 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>

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

Public Types

• typedef Nil Type

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

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

Public Member Functions

Lock (Mutex *)

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

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

Public Member Functions

• Lock (Mutex *mtx)

Public Attributes

Mutex * mtx_

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

5.100 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 >

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

Public Member Functions

Lock (T *)

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

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

Public Member Functions

• Lock (T *t)

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

5.103 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 >

5.104 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 >

5.105 lib::test::TestManager::Log Struct Reference

Public Member Functions

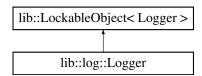
- virtual std::string **read** ()=0
- virtual bool **isEmpty** () const =0

5.106 lib::log::Logger Class Reference

Logger class.

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

Inheritance diagram for lib::log::Logger:



Public Types

 $\bullet \ \ typedef \ {\color{red} SmartPtr} {\color{red} < } \ {\color{red} Logger} > {\color{red} Logger_ptr}$

- 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 >

5.106.1 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.

5.107 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

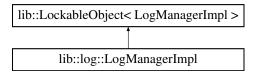
- static const LogLevel INFO
- static const LogLevel WARNING
- static const LogLevel ERROR
- static const LogLevel CRITICAL

5.108 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 &)

5.108.1 Detailed Description

LogManager Singleton, grants access to Logger instances.

Creates and exposes Logger instances by alphanumerical id.

Offers a "root log" for convenience.

5.109 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

5.110 lib::MakeList_0 Struct Reference

Public Types

• typedef Nil Type

5.111 lib::MakeList_1 < A > Struct Template Reference

Public Types

typedef Cons
 A, MAKELIST_0 > Type

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

5.112 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>

5.113 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 >

5.114 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 >

5.115 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 >

5.116 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>

5.117 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>

5.118 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>

5.119 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>

5.120 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>

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

Public Types

• typedef T Type

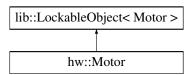
template<template< typename, typename > class F, typename T> struct lib::Merge< F, T, Nil >

5.122 hw::Motor Class Reference

Interface for controlling the motors. Client interface for controlling the conveyor belt and electromagnetic switch. Defines different states that the motors can enter.

```
#include <Motor.h>
```

Inheritance diagram for hw::Motor:



Classes

- struct Direction
- struct Speed
- struct State

Public Types

 $\bullet \ \ \mathsf{typedef} \ \mathsf{lib} \\ \mathsf{::LockableObject} < \mathsf{Motor} > \mathbf{Super} \\$

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

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
 port A pin 4

Friends

· class Actuator

5.122.1 Detailed Description

Interface for controlling the motors. Client interface for controlling the conveyor belt and electromagnetic switch. Defines different states that the motors can enter.

5.122.2 Member Function Documentation

5.122.2.1 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 off, but never suppressed.

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

Controls electromagnetic switch.

Parameters

state | State the switch is supposed to be in.

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

Classes

- class EmptyLock
- class FillLock

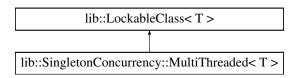
Public Member Functions

• std::size_t size () const

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

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

 $Inheritance\ diagram\ for\ lib:: Singleton Concurrency:: MultiThreaded < T>:$



Classes

struct Lock

 $template {<} typename~T{>}~struct~lib:: Singleton Concurrency:: MultiThreaded {<}~T{>}~$

5.125 lib::Mutex Class Reference

Public Member Functions

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

5.126 lib::Nil Struct Reference

Public Types

• typedef Nil Type

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

Public Attributes

• char v [2]

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

5.128 lib::Not < T > Struct Template Reference

Static Public Attributes

• static const bool value = !T::value

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

5.129 lib::OneParamFtor < R, T > Class Template Reference

Public Member Functions

- template<typename F >
 - OneParamFtor (F f)
- template<typename TT >
 - OneParamFtor (TT *t, R(TT::*f)(T))
- R operator() (T t)
- · operator bool () const

template < typename R, typename T> class lib::OneParamFtor < R, T>

5.130 lib::OneParamFtor < R, void > Class Template Reference

Public Member Functions

• template<typename F > OneParamFtor (F f)

- template<typename TT >
 OneParamFtor (TT *t, R(TT::*f)(void))
- R operator() ()
- operator bool () const

template<typename R> class lib::OneParamFtor< R, void >

5.131 lib::OneParamFtor < void, T > Class Template Reference

Public Member Functions

- template<typename F >OneParamFtor (F f)
- template<typename TT >

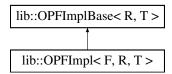
OneParamFtor (TT *t, void(TT::*f)(T))

- void operator() (T t)
- · operator bool () const

template<typename T> class lib::OneParamFtor< void, T>

5.132 lib::OPFImpl < F, R, T > Class Template Reference

Inheritance diagram for lib::OPFImpl< F, R, T >:



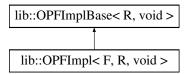
Public Member Functions

- OPFImpl (F f)
- R operator() (T t)

template<typename F, typename T> class lib::OPFImpl< F, R, T>

5.133 lib::OPFImpl < F, R, void > Class Template Reference

Inheritance diagram for lib::OPFImpl< F, R, void >:

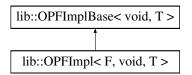


- OPFImpl (F f)
- R operator() ()

template<typename F, typename R> class lib::OPFImpl< F, R, void >

5.134 lib::OPFImpl < F, void, T > Class Template Reference

Inheritance diagram for lib::OPFImpl< F, void, T >:



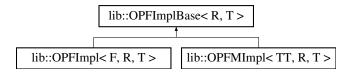
Public Member Functions

- OPFImpl (F f)
- void operator() (T t)

template<typename F, typename T> class lib::OPFImpl< F, void, T>

5.135 lib::OPFImplBase < R, T > Struct Template Reference

Inheritance diagram for lib::OPFImplBase < R, T >:

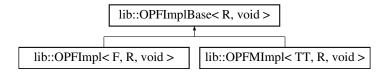


• virtual R operator() (T)=0

template < typename R, typename T> struct lib::OPFImplBase < R, T>

5.136 lib::OPFImplBase < R, void > Struct Template Reference

Inheritance diagram for lib::OPFImplBase< R, void >:



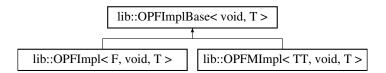
Public Member Functions

• virtual R operator() ()=0

template<typename R> struct lib::OPFImplBase< R, void >

5.137 lib::OPFImplBase < void, T > Struct Template Reference

Inheritance diagram for lib::OPFImplBase< void, T >:



Public Member Functions

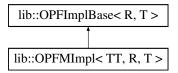
• virtual void operator() (T)=0

template<typename T> struct lib::OPFImplBase< void, T>

5.138 lib::OPFMImpl < TT, R, T > Class Template Reference

Inheritance diagram for lib::OPFMImpl< TT, R, T >:

Generated on Wed May 18 2016 10:32:50 for TI4_SE2 by Doxygen

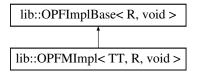


- OPFMImpI (TT *t, fn_t f)
- R operator() (T t)

template<typename TT, typename R, typename T> class lib::OPFMImpl< TT, R, T>

5.139 lib::OPFMImpl < TT, R, void > Class Template Reference

Inheritance diagram for lib::OPFMImpl< TT, R, void >:



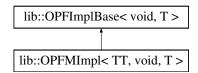
Public Member Functions

- OPFMImpI (TT *t, fn_t f)
- R operator() ()

 ${\it template}{<}{\it typename~R}{>}{\it class~lib::}{\it OPFMImpl}{<}{\it TT,R,void}{>}$

5.140 lib::OPFMImpl < TT, void, T > Class Template Reference

Inheritance diagram for lib::OPFMImpl < TT, void, T >:



- **OPFMImpI** (TT *t, fn t f)
- void operator() (T t)

template<typename TT, typename T> class lib::OPFMImpl< TT, void, T>

5.141 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>

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

Public Types

• typedef **O**r< T1, T2 > **Type**

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

5.143 haw::Project Class Reference

- void run ()
- void calibrateHM (uint16_t, uint16_t)
- · void startHM ()
- uint32_t stopHM ()
- void calibrateDistances (lib::Time slow, lib::Time fast, lib::Time toHM, lib::Time puk)
- void calibrateFromConfig ()
- void saveConfig ()
- lib::Speed getSpeed (int s) const
- uint32 t getPukWidth () const
- uint32_t hmPosition () const
- uint32_t endPosition () const

Static Public Attributes

- static const int SPEED_STOP = 0
- static const int **SPEED_SLOW** = 1
- static const int SPEED_FAST = 2
- static const int CSPEEDS = 3

5.144 haw::Puk Class Reference

Classes

struct Type

Public Member Functions

- **Puk** (uint32_t id, int32_t w)
- uint32 t id () const
- uint32_t type () const
- bool upsideDown () const
- bool metal () const
- void **update** (int32_t p)
- void **setType** (uint32_t t)
- void setUpsideDown ()
- void setMetal ()
- int32_t position () const
- bool isln (int32_t p, int f) const

Static Public Attributes

- static const uint32_t UPSIDE_DOWN = 4
- static const uint32_t IS_METAL = 8

5.145 haw::Ready Class Reference

Inheritance diagram for haw::Ready:



- virtual void enter ()
- virtual void exit ()
- virtual void **process** (const Event &)

5.146 lib::qnx::Receiver Class Reference

Public Member Functions

• Data_ptr receive ()

Friends

· class Channel

5.147 lib::test::TestManager::Registrar Struct Reference

Public Member Functions

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

5.148 lib::Reverse < List > Struct Template Reference

Public Member Functions

typedef DO (ReverseImpl< Nil, List >) Type

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

5.149 lib::ReverseCons < Cell > Struct Template Reference

Public Types

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

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

5.150 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>

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

Public Types

· typedef Done Type

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

5.152 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

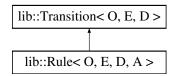
Static Public Attributes

• 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 >

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

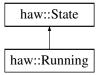
Inheritance diagram for lib::Rule < O, E, D, A >:



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

5.154 haw::Running Class Reference

Inheritance diagram for haw::Running:



Public Member Functions

- Running (Project &)
- virtual void enter ()
- · virtual void exit ()
- virtual void update (lib::Time)
- virtual void **process** (const Event &)
- virtual void execute ()

Static Public Attributes

- static const uint32 t PUK_FLAT = 0
- static const uint32_t PUK_LARGE = 1
- static const uint32 t PUK METAL = 2
- static const uint32_t CPUKS = 3

5.155 lib::test::TestManager::Selector Struct Reference

Public Member Functions

• Selector (const std::string &unit id)

5.156 lib::Semaphore Class Reference

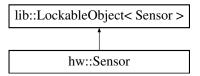
- Semaphore (unsigned=0)
- void **up** ()
- void down ()
- unsigned get () const

5.157 hw::Sensor Class Reference

Interface for checking the state of the Sensors. The sensors can be checked for different states. For example if the sensor is still active or the State of the Sensor changed.

```
#include <Sensor.h>
```

Inheritance diagram for hw::Sensor:



Public Member Functions

· bool entering ()

Public functions to check the state of the Sensors.

- · bool enteringChanged () const
- bool inHM ()
- bool inHMChanged () const
- bool hmValid ()
- bool hmValidChanged () const
- bool inSwitch ()
- bool inSwitchChanged () const
- bool isMetal ()
- bool isMetalChanged () const
- bool switchOpen ()
- bool switchOpenChanged () const
- bool rampFull ()
- bool rampFullChanged () const
- bool leaving ()
- bool leavingChanged () const
- bool start ()
- bool startChanged () const
- bool stop ()
- bool stopChanged () const
- bool reset ()
- · bool resetChanged () const
- bool estop ()
- · bool estopChanged () const
- uint16_t getHeight () const
- void handlePulse (uint32_t)
- void shutdown ()

5.157.1 Detailed Description

Interface for checking the state of the Sensors. The sensors can be checked for different states. For example if the sensor is still active or the State of the Sensor changed.

Additionally the current height of the HM can be read.

5.157.2 Member Function Documentation

```
5.157.2.1 bool hw::Sensor::entering() [inline]
```

Public functions to check the state of the Sensors.

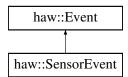
Functions containing Changed in the Title check the flags, if the state of the Sensor changed. Functions without Changed in the Title check the current State of the Sensor.

```
5.157.2.2 void hw::Sensor::handlePulse ( uint32_t pulse )
```

Holds the current pulse. Sensor signals are bit flipped to 0 when not activated.

5.158 haw::SensorEvent Class Reference

Inheritance diagram for haw::SensorEvent:



Classes

struct Sensors

Public Types

- typedef uint32_t sensor_t
- typedef Event::event_id_t event_id_t

- SensorEvent (sensor_t s, bool v)
- · virtual event id t id () const

- sensor t sensor () const
- · bool value () const

5.159 haw::SensorEvent::Sensors Struct Reference

Static Public Attributes

- static const uint32_t ENTERING = 0
- static const uint32 t LEAVING = 1
- static const uint32_t IN_HM = 2
- static const uint32_t IN_SWITCH = 3
- static const uint32 t START = 4
- static const uint32 t STOP = 5
- static const uint32 t RESET = 6
- static const uint32 t ESTOP = 7

5.160 lib::Setify < List > Struct Template Reference

Public Member Functions

typedef DO (SetifyImpl< Nil, List >) Type

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

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

Public Types

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

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

Public Member Functions

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

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

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

Classes

- struct EmptyLock
- struct FillLock

Public Member Functions

• std::size_t size () const

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

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

Classes

struct Lock

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

5.165 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.

5.165.1 Detailed Description

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

Template for convenient Singleton creation.

Parameters are:

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

5.165.2 Member Function Documentation

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

Access singleton class implementation.

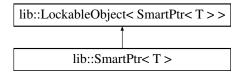
Uses the double checked locking pattern for creation synchronization.

5.166 lib::SmartPtr< T > Class Template Reference

Smart pointer class for automatic life time management.

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

Inheritance diagram for lib::SmartPtr< T >:



- 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

5.166.1 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.

5.167 hw::Motor::Speed Struct Reference

Static Public Attributes

```
• static const pid_t FAST = 0x00
```

no pin

• static const pid_t SLOW = 0x04

port A pin 2

• static const pid_t STOP = 0x08

port A pin 3

5.168 lib::Speed Class Reference

Public Member Functions

- Speed (Time t)
- uint32_t in (Time t) const

Static Public Attributes

• static const uint32_t reference = 1 << 26

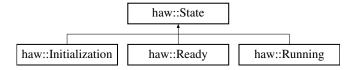
5.169 hw::Motor::State Struct Reference

Static Public Attributes

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

5.170 haw::State Class Reference

Inheritance diagram for haw::State:



Public Types

• typedef int32_t update_t

Public Member Functions

- virtual void enter ()
- virtual void exit ()
- virtual void update (lib::Time)
- virtual void process (const Event &)
- virtual void execute ()
- virtual update_t getNext ()

Static Public Attributes

- static const update_t PREVIOUS = -1
- static const update_t THIS = 0
- static const update_t **NEXT** = 1

Protected Member Functions

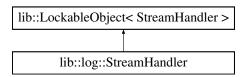
void setNext (update_t u)

5.171 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:



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

5.171.1 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.

5.172 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 ()

5.173 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.

• \sim Thread ()

Destructor.

• 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 *)

5.173.1 Detailed Description

Encapsulates the most important features of a thread.

5.173.2 Constructor & Destructor Documentation

5.173.2.1 lib::Thread::Thread (void)

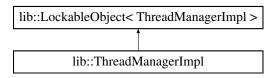
Default constructor.

Initializes inert Thread

```
5.173.2.2 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.
5.173.2.3 lib::Thread::~Thread (void)
Destructor.
Warning
    terminates if this Thread is still joinable
5.173.3 Member Function Documentation
5.173.3.1 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
5.173.3.2 boollib::Thread::joinable()const [inline]
Wether or not the Thread is joinable.
Warning
    Thread cannot be destroyed while joinable
5.173.3.3 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
```

5.174 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 ()

5.175 lib::Time Class Reference

Data class representing a timeframe with microsecond accuracy.

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

Public Types

• typedef uint32_t us_t

- Time (us_t t)
- · void wait () const
- void toTimespec (timespec *)
- us_t raw () const
- Time & operator+= (const Time &t)
- Time operator+ (const Time &t) 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

5.175.1 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.

5.176 lib::Timer Class Reference

Timer that allows scheduling of functors.

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

Classes

struct Ftor

Public Types

typedef uint64_t ts_t

• Time 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.

 $\bullet \;\; {\sf template}{<} {\sf typename} \; {\sf F} >$

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.

5.176.1 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.

5.176.2 Member Function Documentation

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

Is timer currently waiting for ftor execution.

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

Deactivate all timers.

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

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

Amount of time elapsed since last reset.

Resets timer.

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

Amount of time elapsed since last reset.

Doesn't reset timer.

5.176.2.5 Time 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 ($f{1}{t}$)

5.176.2.6 Timer::ts_t lib::Timer::timestamp (void) [static]

Returns current system time in nanoseconds since Jan.

1st 1970.

5.177 lib::TimerPoolImpl Class Reference

Public Types

 $\bullet \ \ typedef \ Singleton < TimerPoolImpl > SingletonInst$

Friends

· class Timer

5.178 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>

5.179 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)

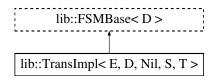
Static Public Attributes

- static const int OriginID = ValueIdentity<DO(GetValue<StateList, Origin>)>-
- 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>

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

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

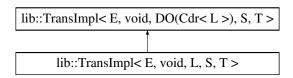


• virtual void process (const E &e)

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

5.181 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)

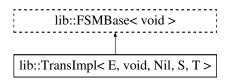
Static Public Attributes

- 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 >

5.182 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 >

5.183 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 >

5.184 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 >

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

Classes

struct Check

Static Public Member Functions

```
    template<typename TT >
        static void test (const E &e, Check< TT,&TT::apply > *)
```

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

• static void call (const E &e)

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

5.186 haw::Puk::Type Struct Reference

Static Public Attributes

- static const uint32_t UNKNOWN = 0
- static const uint32 t FLAT = 1
- static const uint32_t LARGE = 2

5.187 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)

5.188 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>

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

Static Public Attributes

• static const int value = I

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

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

Static Public Attributes

• static const int value = I

 $template < int \ l > struct \ lib:: Value I dentity < Int < \ l > >$

5.191 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> >

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

Public Attributes

• char v [1]