

Tekkotsu Reference Manual

1.4

Generated by Doxygen 1.3.3

Sun Jul 27 17:00:13 2003

Contents

1	Tekkotsu Namespace Index	1
1.1	Tekkotsu Namespace List	1
2	Tekkotsu Hierarchical Index	3
2.1	Tekkotsu Class Hierarchy	3
3	Tekkotsu Compound Index	9
3.1	Tekkotsu Compound List	9
4	Tekkotsu File Index	17
4.1	Tekkotsu File List	17
5	Tekkotsu Page Index	27
5.1	Tekkotsu Related Pages	27
6	Tekkotsu Namespace Documentation	29
6.1	ButtonSourceID Namespace Reference	29
6.2	ERS210Info Namespace Reference	31
6.3	ERS220Info Namespace Reference	58
6.4	ERS2xxInfo Namespace Reference	88
6.5	GVector Namespace Reference	116
6.6	mathutils Namespace Reference	121
6.7	PowerSourceID Namespace Reference	123
6.8	RobotInfo Namespace Reference	126
6.9	SensorSourceID Namespace Reference	127

6.10	SocketNS Namespace Reference	128
6.11	std Namespace Reference	130
6.12	VisionEventNS Namespace Reference	131
6.13	VisionInterface Namespace Reference	132
6.14	WM2Kludge Namespace Reference	136
7	Tekkotsu Class Documentation	139
7.1	_afsLandmarkLoc Struct Reference	139
7.2	_afsLastObservation Struct Reference	142
7.3	_afsParticle Struct Reference	144
7.4	_afsPose Struct Reference	146
7.5	_afsRRP Struct Reference	148
7.6	_afsXY Struct Reference	149
7.7	_dm_cell Struct Reference	150
7.8	_FastSLAM_update Struct Reference	152
7.9	_hm_cell Struct Reference	154
7.10	AGM Class Reference	156
7.11	Aibo3DControllerBehavior Class Reference	158
7.12	Aibo3DMonitorBehavior Class Reference	164
7.13	ALM Class Reference	167
7.14	AutoGetupBehavior Class Reference	171
7.15	BanditMachine Class Reference	175
7.16	BanditMachine::DecideNode Class Reference	180
7.17	BanditMachine::PressNode Class Reference	183
7.18	BanditMachine::WaitNode Class Reference	186
7.19	basic_iNetStream< charT, traits > Class Template Reference	190
7.20	basic_ioNetStream< charT, traits > Class Template Reference	195
7.21	basic_netbuf< charT, traits > Class Template Reference	200
7.22	basic_oNetStream< charT, traits > Class Template Reference	208
7.23	BatteryCheckControl Class Reference	213
7.24	BatteryMonitorBehavior Class Reference	217

7.25 BehaviorActivatorControl Class Reference	224
7.26 BehaviorBase Class Reference	228
7.27 BehaviorSwitchActivatorControl Class Reference	234
7.28 BehaviorSwitchControl< B, AI > Class Template Reference	238
7.29 BehaviorSwitchControlBase Class Reference	245
7.30 BehaviorSwitchControlBase::BehaviorGroup Class Reference	249
7.31 BodyPosition Struct Reference	252
7.32 CameraBehavior Class Reference	254
7.33 char_traits< T > Class Template Reference	259
7.34 ChaseBallBehavior Class Reference	261
7.35 CompareTrans< T > Class Template Reference	265
7.36 Config Class Reference	270
7.37 Config::behaviors_config Struct Reference	275
7.38 Config::controller_config Struct Reference	276
7.39 Config::main_config Struct Reference	278
7.40 Config::motion_config Struct Reference	282
7.41 Config::sound_config Struct Reference	284
7.42 Config::vision_config Struct Reference	286
7.43 Config::wireless_config Struct Reference	289
7.44 Config::worldmodel2_config Struct Reference	290
7.45 ControlBase Class Reference	292
7.46 Controller Class Reference	306
7.47 dmPickCluster Struct Reference	322
7.48 dmPickColor Struct Reference	323
7.49 dmPickConfidence Struct Reference	324
7.50 dmPickDepth Struct Reference	325
7.51 dmPicker Struct Reference	326
7.52 DriveMeBehavior Class Reference	327
7.53 DumbWM2Behavior Class Reference	332
7.54 DumpFileControl Class Reference	336
7.55 DynamicMotionSequence Class Reference	338

7.56 EmergencyStopMC Class Reference	344
7.57 EStopControllerBehavior Class Reference	352
7.58 EventBase Class Reference	357
7.59 EventListener Class Reference	372
7.60 EventLogger Class Reference	374
7.61 EventRouter Class Reference	378
7.62 EventRouter::EventManager Class Reference	394
7.63 EventRouter::TimerEntry Struct Reference	400
7.64 EventRouter::TimerEntryPtrCmp Class Reference	404
7.65 EventTranslator Class Reference	405
7.66 EventTrapper Class Reference	410
7.67 EvtRptBehavior Class Reference	412
7.68 Factory< B > Class Template Reference	415
7.69 Factory1Arg< B, A1, a1 > Class Template Reference	417
7.70 FileBrowserControl Class Reference	418
7.71 FollowHeadBehavior Class Reference	425
7.72 FreeMemReportControl Class Reference	430
7.73 HeadLevelBehavior Class Reference	436
7.74 HeadPointControllerBehavior Class Reference	440
7.75 HeadPointerMC Class Reference	447
7.76 HelpControl Class Reference	457
7.77 HermiteSplineSegment Class Reference	460
7.78 hmPickCluster Struct Reference	462
7.79 hmPickColor Struct Reference	463
7.80 hmPickConfidence Struct Reference	464
7.81 hmPicker Struct Reference	465
7.82 hmPickHeight Struct Reference	466
7.83 hmPickTrav Struct Reference	467
7.84 iostream Class Reference	468
7.85 istream Class Reference	469
7.86 karmedbanditExp3 Class Reference	470

7.87 karmedbanditExp3.1 Class Reference	474
7.88 LedEngine Class Reference	477
7.89 LedEngine::LEDInfo Struct Reference	489
7.90 LedMC Class Reference	492
7.91 ListMemBuf< T_t, MAX, idx_t > Class Template Reference	495
7.92 ListMemBuf< T_t, MAX, idx_t >::entry_t Struct Reference	509
7.93 LoadPostureControl Class Reference	511
7.94 LoadSave Class Reference	514
7.95 LoadWalkControl Class Reference	537
7.96 LockScope< num_doors > Class Template Reference	539
7.97 LocomotionEvent Class Reference	541
7.98 Marker Struct Reference	546
7.99 MCValueEditControl< T > Class Template Reference	547
7.100MMAccessor< MC_t > Class Template Reference	549
7.101MMCombo Class Reference	556
7.102MotionCommand Class Reference	571
7.103MotionManager Class Reference	580
7.104MotionManager::CommandEntry Struct Reference	600
7.105MotionManager::OutputState Class Reference	603
7.106MotionManager::PIDUpdate Struct Reference	606
7.107MotionManagerMsg Struct Reference	608
7.108MotionRequest Struct Reference	612
7.109MotionSequence Class Reference	614
7.110MotionSequence::Move Struct Reference	633
7.111MotionSequenceMC< MAXMOVE > Class Template Reference	635
7.112MutexLock< num_doors > Class Template Reference	642
7.113MutexLock< num_doors >::door_t Struct Reference	649
7.114NonUniformHermiteSplineSegment Class Reference	652
7.115NullControl Class Reference	654
7.116VisionInterface::ObjectInfo Struct Reference	658
7.117OObject Class Reference	659

7.118ostream Class Reference	660
7.119OutputCmd Class Reference	661
7.120OutputNode Class Reference	665
7.121OutputPID Class Reference	668
7.122PIDMC Class Reference	671
7.123PlaySoundControl Class Reference	677
7.124PostureEngine Class Reference	679
7.125PostureMC Class Reference	690
7.126ProcessID Class Reference	698
7.127Profiler Class Reference	701
7.128Profiler::SectionInfo Struct Reference	710
7.129Profiler::Timer Class Reference	714
7.130ProfilerCheckControl Class Reference	719
7.131ProfilerOfSize< MaxSections > Class Template Reference	721
7.132RebootControl Class Reference	723
7.133ReferenceCounter Class Reference	725
7.134RemoteControllerMC Class Reference	728
7.135RemoteProcess Class Reference	732
7.136RunSequenceControl< SequenceSize > Class Template Reference	737
7.137SavePostureControl Class Reference	740
7.138SaveWalkControl Class Reference	742
7.139Serializer Class Reference	745
7.140SharedObject< MC > Class Template Reference	747
7.141SharedObjectBase Class Reference	751
7.142SharedQueue< maxsize, maxentries > Class Template Reference	754
7.143SharedQueue< maxsize, maxentries >::entry_t Struct Reference	760
7.144ShutdownControl Class Reference	761
7.145SimpleChaseBallBehavior Class Reference	763
7.146SmoothCompareTrans< T > Class Template Reference	766
7.147Socket Class Reference	770
7.148SoundManager Class Reference	783

7.149SoundManager::PlayState Struct Reference	800
7.150SoundManager::SoundData Struct Reference	802
7.151SoundManagerMsg Struct Reference	805
7.152SoundPlay Class Reference	810
7.153SoundTestBehavior Class Reference	818
7.154SplinePath Class Reference	823
7.155StareAtBallBehavior Class Reference	826
7.156StartupBehavior Class Reference	830
7.157StateNode Class Reference	835
7.158streambuf Class Reference	843
7.159StringInputControl Class Reference	844
7.160TailWagMC Class Reference	849
7.161TextMsgEvent Class Reference	855
7.162TimeET Class Reference	860
7.163TimeOutTrans Class Reference	867
7.164timeval Struct Reference	870
7.165timezone Struct Reference	871
7.166ToggleHeadLightBehavior Class Reference	872
7.167Transition Class Reference	875
7.168ValueEditControl< T > Class Template Reference	878
7.169ValueSetControl< T > Class Template Reference	886
7.170GVector::vector2d< num > Class Template Reference	891
7.171GVector::vector3d< num > Class Template Reference	897
7.172Vision Class Reference	904
7.173VisionEvent Class Reference	918
7.174VisionEventSpec Struct Reference	924
7.175VisionObjectInfo Struct Reference	926
7.176VisionSerializer Class Reference	927
7.177VisualTargetCloseTrans Class Reference	930
7.178VisionInterface::VObject Struct Reference	933
7.179WalkControllerBehavior Class Reference	935

7.180WalkMC Class Reference	944
7.181WalkMC::LegParam Struct Reference	958
7.182WalkMC::LegWalkState Struct Reference	960
7.183WalkMC::WalkParam Struct Reference	962
7.184WalkToTargetMachine Class Reference	965
7.185WAV Class Reference	970
7.186Wireless Class Reference	974
7.187WorldModel2 Class Reference	985
7.188WorldModel2Behavior Class Reference	996
7.189WorldModel2Behavior::GawkNode Struct Reference	1000
7.190WorldModel2Behavior::WaitNode Struct Reference	1004
7.191WorldModel2Behavior::WalkNode Struct Reference	1007
7.192WorldState Class Reference	1011
7.193WorldStateSerializer Class Reference	1020
 8 Tekkotsu File Documentation	 1023
8.1 afsFindBestPose.cc File Reference	1023
8.2 afsFindBestPose.h File Reference	1026
8.3 afsMain.cc File Reference	1029
8.4 afsMain.h File Reference	1032
8.5 afsMeasurementUpdate.cc File Reference	1035
8.6 afsMeasurementUpdate.h File Reference	1037
8.7 afsMotionResample.cc File Reference	1038
8.8 afsMotionResample.h File Reference	1039
8.9 afsParticle.cc File Reference	1040
8.10 afsParticle.h File Reference	1041
8.11 afsTriangulator.cc File Reference	1044
8.12 afsTriangulator.h File Reference	1045
8.13 afsUtility.cc File Reference	1046
8.14 afsUtility.h File Reference	1047
8.15 agmMain.cc File Reference	1048

8.16	agmMain.h File Reference	1049
8.17	Aibo3DControllerBehavior.h File Reference	1050
8.18	Aibo3DMonitorBehavior.h File Reference	1053
8.19	almMain.cc File Reference	1054
8.20	almMain.h File Reference	1056
8.21	almStructures.h File Reference	1057
8.22	almUtility.cc File Reference	1060
8.23	almUtility.h File Reference	1063
8.24	AutoGetupBehavior.h File Reference	1066
8.25	BanditMachine.h File Reference	1068
8.26	BatteryCheckControl.h File Reference	1070
8.27	BatteryMonitorBehavior.h File Reference	1072
8.28	BehaviorActivatorControl.h File Reference	1074
8.29	BehaviorBase.h File Reference	1076
8.30	BehaviorSwitchActivatorControl.h File Reference	1078
8.31	BehaviorSwitchControl.h File Reference	1079
8.32	CameraBehavior.h File Reference	1081
8.33	ChaseBallBehavior.cc File Reference	1083
8.34	ChaseBallBehavior.h File Reference	1085
8.35	CompareTrans.h File Reference	1087
8.36	Config.cc File Reference	1089
8.37	Config.h File Reference	1091
8.38	Configuration.h File Reference	1094
8.39	Configuration.h File Reference	1097
8.40	ControlBase.cc File Reference	1102
8.41	ControlBase.h File Reference	1103
8.42	Controller.cc File Reference	1105
8.43	Controller.h File Reference	1107
8.44	debuget.h File Reference	1109
8.45	DriveMeBehavior.cc File Reference	1113
8.46	DriveMeBehavior.h File Reference	1115

8.47 DumbWM2Behavior.h File Reference	1117
8.48 DumpFileControl.h File Reference	1119
8.49 DynamicMotionSequence.h File Reference	1121
8.50 EmergencyStopMC.cc File Reference	1123
8.51 EmergencyStopMC.h File Reference	1125
8.52 entry.h File Reference	1127
8.53 ERS210Info.h File Reference	1128
8.54 ERS220Info.h File Reference	1130
8.55 ERS2xxInfo.h File Reference	1132
8.56 EStopControllerBehavior.cc File Reference	1134
8.57 EStopControllerBehavior.h File Reference	1135
8.58 EventBase.cc File Reference	1137
8.59 EventBase.h File Reference	1138
8.60 EventListener.h File Reference	1140
8.61 EventLogger.cc File Reference	1142
8.62 EventLogger.h File Reference	1143
8.63 EventRouter.cc File Reference	1145
8.64 EventRouter.h File Reference	1147
8.65 EventTranslator.cc File Reference	1151
8.66 EventTranslator.h File Reference	1153
8.67 EventTrapper.h File Reference	1155
8.68 EvtRptBehavior.cc File Reference	1157
8.69 EvtRptBehavior.h File Reference	1158
8.70 Factory.h File Reference	1159
8.71 FileBrowserControl.cc File Reference	1160
8.72 FileBrowserControl.h File Reference	1161
8.73 FollowHeadBehavior.cc File Reference	1163
8.74 FollowHeadBehavior.h File Reference	1165
8.75 FreeMemReportControl.cc File Reference	1167
8.76 FreeMemReportControl.h File Reference	1168
8.77 Geometry.h File Reference	1170

8.78	get_time.cc File Reference	1173
8.79	get_time.h File Reference	1175
8.80	gvector.h File Reference	1177
8.81	HeadLevelBehavior.h File Reference	1183
8.82	HeadPointControllerBehavior.cc File Reference	1185
8.83	HeadPointControllerBehavior.h File Reference	1186
8.84	HeadPointerMC.cc File Reference	1188
8.85	HeadPointerMC.h File Reference	1189
8.86	HelpControl.cc File Reference	1191
8.87	HelpControl.h File Reference	1192
8.88	ionetstream.h File Reference	1194
8.89	karmedbandit.h File Reference	1196
8.90	Kinematics.cc File Reference	1198
8.91	Kinematics.h File Reference	1206
8.92	LedEngine.cc File Reference	1214
8.93	LedEngine.h File Reference	1215
8.94	LedMC.h File Reference	1217
8.95	ListMemBuf.h File Reference	1219
8.96	LoadPostureControl.h File Reference	1220
8.97	LoadSave.cc File Reference	1222
8.98	LoadSave.h File Reference	1223
8.99	LoadWalkControl.h File Reference	1225
8.100	LockScope.h File Reference	1227
8.101	LocomotionEvent.h File Reference	1229
8.102	mathutils.h File Reference	1231
8.103	MCValueEditControl.h File Reference	1232
8.104	MMAccessor.h File Reference	1234
8.105	MMCombo.cc File Reference	1236
8.106	MMCombo.h File Reference	1239
8.107	MotionCommand.cc File Reference	1241
8.108	MotionCommand.h File Reference	1242

8.109MotionManager.cc File Reference	1244
8.110MotionManager.h File Reference	1246
8.111MotionManagerMsg.h File Reference	1250
8.112motionReshapeKludge.h File Reference	1251
8.113MotionSequenceMC.cc File Reference	1255
8.114MotionSequenceMC.h File Reference	1256
8.115MutexLock.h File Reference	1258
8.116NullControl.h File Reference	1260
8.117OutputCmd.h File Reference	1262
8.118OutputNode.h File Reference	1263
8.119OutputPID.h File Reference	1265
8.120Path.h File Reference	1266
8.121PIDMC.h File Reference	1268
8.122PlaySoundControl.h File Reference	1270
8.123Poses.h File Reference	1272
8.124PostureEngine.cc File Reference	1275
8.125PostureEngine.h File Reference	1276
8.126PostureMC.h File Reference	1278
8.127ProcessID.cc File Reference	1280
8.128ProcessID.h File Reference	1281
8.129Profiler.cc File Reference	1282
8.130Profiler.h File Reference	1283
8.131ProfilerCheckControl.h File Reference	1286
8.132RebootControl.cc File Reference	1288
8.133RebootControl.h File Reference	1289
8.134ReferenceCounter.h File Reference	1291
8.135RemoteControllerMC.h File Reference	1293
8.136RemoteProcess.cc File Reference	1295
8.137RemoteProcess.h File Reference	1296
8.138RobotInfo.h File Reference	1298
8.139RunSequenceControl.h File Reference	1300

8.140SavePostureControl.h File Reference	1302
8.141SaveWalkControl.h File Reference	1304
8.142Serializer.h File Reference	1306
8.143SharedObject.h File Reference	1307
8.144SharedQueue.h File Reference	1309
8.145ShutdownControl.cc File Reference	1311
8.146ShutdownControl.h File Reference	1312
8.147SimpleChaseBallBehavior.h File Reference	1314
8.148SmoothCompareTrans.h File Reference	1316
8.149Socket.cc File Reference	1318
8.150Socket.h File Reference	1320
8.151SoundManager.cc File Reference	1322
8.152SoundManager.h File Reference	1325
8.153SoundManagerMsg.h File Reference	1328
8.154SoundPlay.cc File Reference	1329
8.155SoundPlay.h File Reference	1331
8.156SoundTestBehavior.h File Reference	1333
8.157Spline.h File Reference	1335
8.158StareAtBallBehavior.cc File Reference	1338
8.159StareAtBallBehavior.h File Reference	1339
8.160StartupBehavior.cc File Reference	1341
8.161StartupBehavior.h File Reference	1345
8.162StateNode.cc File Reference	1346
8.163StateNode.h File Reference	1347
8.164StringInputControl.cc File Reference	1349
8.165StringInputControl.h File Reference	1350
8.166SystemUtility.h File Reference	1352
8.167TailWagMC.h File Reference	1354
8.168Test.c File Reference	1356
8.169TextMsgEvent.h File Reference	1358
8.170TimeET.cc File Reference	1360

8.171TimeET.h File Reference	1361
8.172TimeOutTrans.h File Reference	1364
8.173ToggleHeadLightBehavior.h File Reference	1366
8.174Transition.cc File Reference	1368
8.175Transition.h File Reference	1369
8.176Util.h File Reference	1370
8.177ValueEditControl.h File Reference	1375
8.178ValueSetControl.h File Reference	1377
8.179Vision.cc File Reference	1379
8.180Vision.h File Reference	1383
8.181Visiondefines.h File Reference	1388
8.182VisionEvent.h File Reference	1391
8.183VisionInterface.h File Reference	1393
8.184VisionSerializer.cc File Reference	1395
8.185VisionSerializer.h File Reference	1397
8.186VisualTargetCloseTrans.h File Reference	1399
8.187WalkControllerBehavior.cc File Reference	1401
8.188WalkControllerBehavior.h File Reference	1402
8.189WalkMC.cc File Reference	1404
8.190WalkMC.h File Reference	1407
8.191WalkMotionModel.cc File Reference	1410
8.192WalkMotionModel.h File Reference	1411
8.193WalkToTargetMachine.cc File Reference	1412
8.194WalkToTargetMachine.h File Reference	1414
8.195WAV.cc File Reference	1416
8.196WAV.h File Reference	1418
8.197Wireless.cc File Reference	1420
8.198Wireless.h File Reference	1423
8.199WorldModel2.cc File Reference	1426
8.200WorldModel2.h File Reference	1429
8.201WorldModel2Behavior.cc File Reference	1432

8.202WorldModel2Behavior.h File Reference	1434
8.203WorldState.cc File Reference	1436
8.204WorldState.h File Reference	1439
8.205WorldStateSerializer.cc File Reference	1443
8.206WorldStateSerializer.h File Reference	1444
9 Tekkotsu Page Documentation	1445
9.1 Todo List	1445
9.2 Test List	1448

Chapter 1

Tekkotsu Namespace Index

1.1 Tekkotsu Namespace List

Here is a list of all namespaces with brief descriptions:

ButtonSourceID (Holds source ID types for button events EventBase Button-SourceID.t)	29
ERS210Info (Contains information about the ERS-210 Robot, such as number of joints, PID defaults, timing information, etc)	31
ERS220Info (Contains information about the ERS-220 Robot, such as number of joints, PID defaults, timing information, etc)	58
ERS2xxInfo (Contains information about the ERS-2xx series of robot, such as number of joints, PID defaults, timing information, etc)	88
GVector	116
mathutils	121
PowerSourceID (Holds source ID types for power events EventBase Power-SourceID.t)	123
RobotInfo (Contains information about the robot, such as number of joints, PID defaults, timing information, etc)	126
SensorSourceID (Holds source ID types for sensor events EventBase Sensor-SourceID.t)	127
SocketNS	128
std	130
VisionEventNS (Contains source IDs for the objects we can recognize) . . .	131
VisionInterface	132
WM2Kludge (Symbols for kludges that alter the behavior of WorldModel2 mapping)	136

Chapter 2

Tekkotsu Hierarchical Index

2.1 Tekkotsu Class Hierarchy

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

_afsLandmarkLoc	139
_afsLastObservation	142
_afsParticle	144
_afsPose	146
_afsRRP	148
_afsXY	149
_dm_cell	150
_FastSLAM_update	152
_hm_cell	154
AGM	156
ALM	167
BodyPosition	252
char_traits< T >	259
Config	270
Config::behaviors_config	275
Config::controller_config	276
Config::main_config	278
Config::motion_config	282
Config::sound_config	284
Config::vision_config	286
Config::wireless_config	289
Config::worldmodel2_config	290
ControlBase	292
BatteryCheckControl	213
BehaviorSwitchActivatorControl	234

BehaviorSwitchControlBase	245
BehaviorSwitchControl< B, AI >	238
EventLogger	374
FileBrowserControl	418
DumpFileControl	336
LoadPostureControl	511
LoadWalkControl	537
PlaySoundControl	677
RunSequenceControl< SequenceSize >	737
FreeMemReportControl	430
NullControl	654
BehaviorActivatorControl	224
HelpControl	457
RebootControl	723
ShutdownControl	761
ProfilerCheckControl	719
SaveWalkControl	742
StringInputControl	844
SavePostureControl	740
ValueEditControl< T >	878
MCValueEditControl< T >	547
ValueSetControl< T >	886
dmPicker	326
dmPickCluster	322
dmPickColor	323
dmPickConfidence	324
dmPickDepth	325
EventListener	372
BatteryCheckControl	213
BehaviorBase	228
Aibo3DControllerBehavior	158
Aibo3DMonitorBehavior	164
AutoGetupBehavior	171
BatteryMonitorBehavior	217
CameraBehavior	254
ChaseBallBehavior	261
Controller	306
DriveMeBehavior	327
DumbWM2Behavior	332
EStopControllerBehavior	352
EvtRptBehavior	412
FollowHeadBehavior	425
FreeMemReportControl	430
HeadLevelBehavior	436

HeadPointControllerBehavior	440
SimpleChaseBallBehavior	763
SoundTestBehavior	818
StareAtBallBehavior	826
StartupBehavior	830
StateNode	835
BanditMachine	175
BanditMachine::DecideNode	180
BanditMachine::PressNode	183
BanditMachine::WaitNode	186
OutputNode	665
WalkToTargetMachine	965
WorldModel2Behavior	996
WorldModel2Behavior::GawkNode	1000
WorldModel2Behavior::WaitNode	1004
WorldModel2Behavior::WalkNode	1007
ToggleHeadLightBehavior	872
WalkControllerBehavior	935
CompareTrans< T >	265
SmoothCompareTrans< T >	766
EventLogger	374
EventRouter	378
LoadPostureControl	511
RunSequenceControl< SequenceSize >	737
TimeOutTrans	867
ValueEditControl< T >	878
VisualTargetCloseTrans	930
WorldModel2	985
EventRouter::EventManager	394
EventRouter::TimerEntry	400
EventRouter::TimerEntryPtrCmp	404
EventTrapper	410
Controller	306
EventTranslator	405
Factory< B >	415
Factory1Arg< B, A1, a1 >	417
HermiteSplineSegment	460
hmPicker	465
hmPickCluster	462
hmPickColor	463
hmPickConfidence	464
hmPickHeight	466
hmPickTrav	467
iostream	468

basic_ioNetStream< charT, traits >	195
istream	469
basic_iNetStream< charT, traits >	190
karmedbanditExp3	470
karmedbanditExp3_1	474
LedEngine	477
LedMC	492
LedEngine::LEDInfo	489
ListMemBuf< T_t, MAX, idx_t >	495
ListMemBuf< T_t, MAX, idx_t >::entry_t	509
LoadSave	514
EventBase	357
LocomotionEvent	541
TextMsgEvent	855
VisionEvent	918
MotionSequence	614
DynamicMotionSequence	338
MotionSequenceMC< MAXMOVE >	635
PostureEngine	679
PostureMC	690
EmergencyStopMC	344
LockScope< num_doors >	539
Marker	546
MMAccessor< MC_t >	549
MotionManager	580
MotionManager::CommandEntry	600
MotionManager::OutputState	603
MotionManager::PIDUpdate	606
MotionManagerMsg	608
MotionCommand	571
HeadPointerMC	447
LedMC	492
MotionSequence	614
PIDMC	671
PostureMC	690
RemoteControllerMC	728
TailWagMC	849
WalkMC	944
MotionRequest	612
MotionSequence::Move	633
MutexLock< num_doors >	642
MutexLock< num_doors >::door_t	649
NonUniformHermiteSplineSegment	652
VisionInterface::ObjectInfo	658

OObject	659
MMCombo	556
RemoteProcess	732
SoundPlay	810
ostream	660
basic_oNetStream< charT, traits >	208
OutputCmd	661
OutputPID	668
ProcessID	698
Profiler	701
ProfilerOfSize< MaxSections >	721
Profiler::SectionInfo	710
Profiler::Timer	714
ReferenceCounter	725
BehaviorBase	228
BehaviorSwitchControlBase::BehaviorGroup	249
Serializer	745
VisionSerializer	927
WorldModel2	985
WorldStateSerializer	1020
SharedObjectBase	751
SharedObject< MC >	747
SharedQueue< maxsize, maxentries >	754
SharedQueue< maxsize, maxentries >::entry_t	760
Socket	770
SoundManager	783
SoundManager::PlayState	800
SoundManager::SoundData	802
SoundManagerMsg	805
SplinePath	823
streambuf	843
basic_netbuf< charT, traits >	200
TimeET	860
timeval	870
timezone	871
Transition	875
CompareTrans< T >	265
TimeOutTrans	867
VisualTargetCloseTrans	930
GVector::vector2d< num >	891
GVector::vector3d< num >	897
Vision	904
VisionEventSpec	924

VisionObjectInfo	926
VisionInterface::VObject	933
WalkMC::LegParam	958
WalkMC::LegWalkState	960
WalkMC::WalkParam	962
WAV	970
Wireless	974
WorldState	1011

Chapter 3

Tekkotsu Compound Index

3.1 Tekkotsu Compound List

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

_afsLandmarkLoc	139
_afsLastObservation	142
_afsParticle	144
_afsPose	146
_afsRRP	148
_afsXY	149
_dm_cell (A cell for the spherical depth map)	150
_FastSLAM_update	152
_hm_cell (A cell for the horizontal height map)	154
AGM	156
Aibo3DControllerBehavior (Listens to aibo3d control commands coming in from the command port)	158
Aibo3DMonitorBehavior (Sends current pose to Aibo3D GUI, ignores in- coming commands)	164
ALM	167
AutoGetupBehavior (A little background behavior to keep the robot on its feet)	171
BanditMachine (Plays K-armed bandit)	175
BanditMachine::DecideNode (Uses one of the algorithms in karmedbandit.h to decide which paw to press next)	180
BanditMachine::PressNode (This node is used to move a paw down using a MotionSequenceMC)	183
BanditMachine::WaitNode (Waits to see if a reward is received, lights up LEDs to let the user know)	186
basic_iNetStream< charT, traits >	190

basic_ioNetStream< charT, traits >	195
basic_netbuf< charT, traits >	200
basic_oNetStream< charT, traits >	208
BatteryCheckControl (When activated, this will print a battery report to std-out and light up LEDs to specify power level)	213
BatteryMonitorBehavior (A background behavior which will monitor the power level and flip the ears when appropriate on a 210, or blink the headlight if a 220)	217
BehaviorActivatorControl (Upon activation, will start, stop, or toggle a behavior)	224
BehaviorBase (The basis from which all other Behaviors should inherit)	228
BehaviorSwitchActivatorControl (Upon activation, will tell the specified BehaviorSwitchControl to start or stop the behavior)	234
BehaviorSwitchControl< B, AI > (Allows proper switching between major behaviors, calling DoStart and DoStop)	238
BehaviorSwitchControlBase (Holds some utility classes and functions for BehaviorSwitchControl which shouldn't be stored in a templated class)	245
BehaviorSwitchControlBase::BehaviorGroup (A simple utility class to allow the BehaviorSwitchControl 's to be able to deactivate the current behavior when a new one becomes active)	249
BodyPosition (Holds the current location of the body, as a delta from when walking started)	252
CameraBehavior (Will take images and write to log file)	254
char_traits< T >	259
ChaseBallBehavior (A simple behavior to chase after any objects seen by the vision system)	261
CompareTrans< T > (Causes a transition if a value (through a pointer) goes above a given value)	265
Config (Provides global access to system configuration information)	270
Config::behaviors_config (Placeholder)	275
Config::controller_config (Controller information)	276
Config::main_config (Core functionality information)	278
Config::motion_config (Motion information)	282
Config::sound_config (Sound information)	284
Config::vision_config (Vision information)	286
Config::wireless_config (Wireless information)	289
Config::worldmodel2_config (World model information)	290
ControlBase (Base class for all items in the Controller hierarchy)	292
Controller (Handles the menu/command system... when it detects the EmergencyStopMC is activated, it'll kick into high priority)	306
dmPickCluster (Picks cluster membership out of the SDM. For completeness—not use!)	322
dmPickColor (Picks color measurements out of the SDM)	323
dmPickConfidence (Picks confidence values out of the SDM)	324
dmPickDepth (Picks depth measurements out of the SDM)	325

dmPicker (Abstract base class for depth map data pickers. Implementations available)	326
DriveMeBehavior (A very simple behavior that asks the user for WalkMC walking parameters and a walk duration)	327
DumbWM2Behavior (Simply turns on a WM2 object. Useful for running concurrently with other behaviors and seeing what shows up in the world model)	332
DumpFileControl (Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...))	336
DynamicMotionSequence (Uses STL's vector for dynamic memory allocation - don't use this as a motion command, pointers in shared memory regions can be invalid in other processes)	338
EmergencyStopMC (Overrides all joints with high priority freeze, blinks tail pink/red/blue cycle)	344
EStopControllerBehavior (Listens to control commands coming in from the command port for remotely controlling the head)	352
EventBase (The basis of events passed around the high level system)	357
EventListener (An interface to allow a standard method of passing events) . .	372
EventLogger (Allows logging of events to the console or a file)	374
EventRouter (This class will handle distribution of events as well as management of timers)	378
EventRouter::EventManager (Does the actual storage of the mapping between EventBase's and the EventListeners/EventTrappers who should receive them)	394
EventRouter::TimerEntry (Contains all the information needed to maintain a timer by the EventRouter)	400
EventRouter::TimerEntryPtrCmp (Used by STL to sort the timer list in order of activation time)	404
EventTranslator (EventTranslator receives events from EventRouters in non-Main processes and adds them into a SharedQueue for Main to pick up)	405
EventTrapper (An interface to allow a standard method of trapping events) .	410
EvtRptBehavior (A simple behavior to test event reports)	412
Factory< B > (A lightweight class to override for constructing new objects (if you need to pass constructors parameters, etc.))	415
Factory1Arg< B, A1, a1 > (Uses template to specify a constant parameter to the constructor)	417
FileBrowserControl (Displays the contents of a directory in a control menu, probably useful as a baseclass for other controls)	418
FollowHeadBehavior (Will walk where the head is pointing)	425
FreeMemReportControl (Gives reports on free memory size at variable rates, can also warn if free memory drops too low)	430
HeadLevelBehavior (Tests the head leveling code of HeadPointerMC)	436
HeadPointControllerBehavior (Listens to control commands coming in from the command port for remotely controlling the head)	440

HeadPointerMC (This class gives some quick and easy functions to point the head at things)	447
HelpControl (Recurses through the menu system and outputs the name and description of each item)	457
HermiteSplineSegment	460
hmPickCluster (Picks cluster membership out of the HHM or GHM. For completeness—not use!)	462
hmPickColor (Picks color measurements out of the HHM or GHM)	463
hmPickConfidence (Picks confidence values out of the HHM or GHM)	464
hmPicker (Abstract base class for height map data pickers. Implementations available)	465
hmPickHeight (Picks height measurements out of the HHM or GHM)	466
hmPickTrav (Picks traversability values out of the HHM or GHM)	467
iostream	468
istream	469
karmedbanditExp3 (Makes decisions regarding an adversarial k-armed bandit)	470
karmedbanditExp3.1 (Makes decisions regarding an adversarial k-armed bandit)	474
LedEngine (Provides basic LED effects to anything that inherits from (recommended method for MotionCommands) or instantiates it (just in case you have reason to))	477
LedEngine::LEDInfo (Holds all the information needed by each of the LEDs)	489
LedMC (This is just a simple wrapper - you probably want to be looking at LedEngine.h)	492
ListMemBuf< T_t, MAX, idx_t > (Provides some degree of dynamic allocation of a templated type from a buffer of set size)	495
ListMemBuf< T_t, MAX, idx_t >::entry_t (Holds data about an entry in the free/used lists)	509
LoadPostureControl (Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...))	511
LoadSave (Intended as an interface to allow easy and uniform file operations)	514
LoadWalkControl (When activated, loads a set of walk parameters from a file specified by user)	537
LockScope< num_doors > (Locks a MutexLock until the LockScope goes out of scope)	539
LocomotionEvent (Gives updates regarding the current movement of the robot through the world)	541
Marker	546
MCValueEditControl< T > (Allows you to modify a value in memory, much like ValueEditControl , but will check out a MotionCommand first to maintain proper mutual exclusion)	547
MMAccessor< MC_t > (This class allows convenient ways of accessing a motion command)	549
MMCombo (Contains code for both MainObj and MotoObj processes)	556

MotionCommand (The abstract base class for motions, provides common interface. All motions should inherit from this)	571
MotionManager (The purpose of this class is to serialize access to the MotionCommands and simplify their sharing between memory spaces)	580
MotionManager::CommandEntry (All the information we need to maintain about a MotionCommand)	600
MotionManager::OutputState (Holds the full requested value of an output) .	603
MotionManager::PIDUpdate (Used to request pids for a given joint)	606
MotionManagerMsg (A small header that precedes data sent by MotionManager between processes)	608
MotionRequest (Structure for containing motion requests)	612
MotionSequence (A handy little (or not so little) class for switching between a sequence of postures)	614
MotionSequence::Move (This struct holds all the information needed about a frame for a particular output)	633
MotionSequenceMC< MAXMOVE > (Instantiates MotionSequences - when you want to make a new MotionSequence , make one of these)	635
MutexLock< num_doors > (A software only mutual exclusion lock)	642
MutexLock< num_doors >::door_t (Holds per process shared info, one of these per process)	649
NonUniformHermiteSplineSegment	652
NullControl (When activated, this will return immediately (handy for fake items in a menu))	654
VisionInterface::ObjectInfo	658
OObject	659
ostream	660
OutputCmd (This object holds all the information needed to control a single output)	661
OutputNode (A very simple StateNode that outputs its name to a given ostream upon activation, handy for debugging)	665
OutputPID (This object holds all the information needed to control a single output)	668
PIDMC (A nice little MotionCommand for manually manipulating the PID values)	671
PlaySoundControl (Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...))	677
PostureEngine (A class for storing a set of positions and weights for all the outputs)	679
PostureMC (A MotionCommand shell for PostureEngine)	690
ProcessID (This is a class instead of a namespace so i can limit write access of the ID value to the OObjects)	698
Profiler (Managers a hierarchy of timers for profiling time spent in code, gives microsecond resolution)	701
Profiler::SectionInfo (Holds all the information needed for book keeping for each timer)	710

Profiler::Timer (Measures the time that this class exists, reports result to a profiler)	714
ProfilerCheckControl (Causes the WorldState::mainProfile and WorldState::motionProfile to display reports to cout)	719
ProfilerOfSize< MaxSections > (Templated subclass allows compile-time flexibility of how much memory to use)	721
RebootControl (When activated, this will cause the aibo to reboot)	723
ReferenceCounter (Performs simple reference counting, will delete the object when removing the last reference)	725
RemoteControllerMC (This class is used for setting all PIDJoints to a certain set of values (not the gains, just the joint positions))	728
RemoteProcess (Sample RemoteProcessingOPENR process)	732
RunSequenceControl< SequenceSize > (Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...))	737
SavePostureControl (Upon activation, saves the current position to a file name read from user (stored in /ms/data/motion/...))	740
SaveWalkControl (When activated, saves walk parameters to a file specified from cin)	742
Serializer (Provides a default serializer base class for simple objects)	745
SharedObject< MC > (This templated class allows convenient creation of any type of class wrapped in a shared memory region)	747
SharedObjectBase (It's nice to have a parent class of SharedObject (which is what you probably want to be reading) so that you can pass around the data structure without worrying about what type is inside the shared memory region)	751
SharedQueue< maxsize, maxentries > (SharedQueue is a shared memory message buffer for interprocess communication)	754
SharedQueue< maxsize, maxentries >::entry_t (Entry information)	760
ShutdownControl (When activated, this will cause the aibo to shut down) . .	761
SimpleChaseBallBehavior (A simple behavior to chase after any objects seen by the vision system)	763
SmoothCompareTrans< T > (A subclass of CompareTrans , which provides monitoring of exponentially weighted averages to a threshold) . . .	766
Socket (Tekkotsu wireless Socket class)	770
SoundManager (Provides sound effects and caching services, as well as mixing buffers for the SoundPlay process)	783
SoundManager::PlayState (Holds data about sounds currently being played) .	800
SoundManager::SoundData (Holds data about the loaded sounds)	802
SoundManagerMsg (A small header that preceeds data sent by SoundManager between processes)	805
SoundPlay (The process (a.k.a. OObject), which is responsible for sending sound buffers to the system to play)	810
SoundTestBehavior (Allows you to experiment with playing sounds different ways)	818
SplinePath	823

StareAtBallBehavior (A simple behavior to chase after any objects seen by the vision system)	826
StartupBehavior (This Behavior is the only hardcoded behavior by the framework to start up once all the data structures and such are set up) . . .	830
StateNode (Recursive data structure - both a state machine controller as well as a node within a state machine itself)	835
streambuf	843
StringInputControl (Upon activation, prompts the user for a string and stores it)	844
TailWagMC (A simple motion command for wagging the tail - you can specify period, magnitude, and tilt)	849
TextMsgEvent (Extends EventBase to also include actual message text) . . .	855
TimeET (A nice class for handling time values with high precision)	860
TimeOutTrans (Causes a transition after a specified amount of time has passed)	867
timeval (Would be defined by system - we redefine the same structure in case we're compiling for Aperios)	870
timezone (Would be defined by system - we redefine the same structure in case we're compiling for Aperios)	871
ToggleHeadLightBehavior (Opens or closes the head light on an ERS-220) .	872
Transition (Represents a transition between StateNodes)	875
ValueEditControl< T > (Allows real-time modification of a value through a pointer)	878
ValueSetControl< T > (Upon activation, this control will set the target pointer to the specified value)	886
GVector::vector2d< num >	891
GVector::vector3d< num >	897
Vision	904
VisionEvent (Extends EventBase to also include location in the visual field and distance (though distance is not implimented yet))	918
VisionEventSpec	924
VisionObjectInfo	926
VisionSerializer (Encodes and transmits camera images)	927
VisualTargetCloseTrans (Causes a transition when a visual object is "close")	930
VisionInterface::VObject	933
WalkControllerBehavior (Listens to control commands coming in from the command port for remotely controlling the walk)	935
WalkMC (A nice walking class from Carnegie Mellon University's 2001 Robosoccer team, modified to fit this framework, see their license)	944
WalkMC::LegParam (Holds parameters about how to move each leg)	958
WalkMC::LegWalkState (Holds state about each leg's path)	960
WalkMC::WalkParam (Holds more general parameters about the walk) . . .	962
WalkToTargetMachine (A state machine for walking towards a visual target)	965
WAV	970
Wireless (Tekkotsu wireless class)	974

WorldModel2 (Tekkotsu's localization and mapping system (IN ACTIVE DEVELOPMENT))	985
WorldModel2Behavior (Implements a stateful behavior specifically designed to feed data to the second WorldModel)	996
WorldModel2Behavior::GawkNode (This one wags the head around)	1000
WorldModel2Behavior::WaitNode (This one lets the AIBO have a reflective pause)	1004
WorldModel2Behavior::WalkNode (This one does a random walk)	1007
WorldState (The state of the robot and its environment)	1011
WorldStateSerializer	1020

Chapter 4

Tekkotsu File Index

4.1 Tekkotsu File List

Here is a list of all files with brief descriptions:

afsFindBestPose.cc	1023
afsFindBestPose.h	1026
afsMain.cc	1029
afsMain.h	1032
afsMeasurementUpdate.cc	1035
afsMeasurementUpdate.h	1037
afsMotionResample.cc	1038
afsMotionResample.h	1039
afsParticle.cc	1040
afsParticle.h	1041
afsTriangulator.cc	1044
afsTriangulator.h	1045
afsUtility.cc	1046
afsUtility.h	1047
agmMain.cc	1048
agmMain.h	1049
Aibo3DControllerBehavior.h (Defines Aibo3DControllerBehavior , which listens to commands from the Aibo3D gui and shows current state)	1050
Aibo3DMonitorBehavior.h (Defines Aibo3DMonitorBehavior , which sends current pose to Aibo3D GUI, ignores incoming commands)	1053
almMain.cc	1054
almMain.h	1056
almStructures.h	1057
almUtility.cc	1060
almUtility.h	1063

AutoGetupBehavior.h (Defines AutoGetupBehavior , a little background behavior to keep the robot on its feet)	1066
BanditMachine.h (Defines BanditMachine , A state machine for playing k-armed bandit)	1068
BatteryCheckControl.h (Defines BatteryCheckControl , which will spew a power report to stdout upon activation)	1070
BatteryMonitorBehavior.h (Defines BatteryMonitorBehavior , a background behavior to trigger BatteryMonitorMC to warn when the power is low)	1072
BehaviorActivatorControl.h (Defines BehaviorActivatorControl , which can either start, stop, or toggle a behavior when activated)	1074
BehaviorBase.h (Defines BehaviorBase from which all Behaviors should inherit)	1076
BehaviorSwitchActivatorControl.h (Defines BehaviorSwitchActivatorControl , which will tell the specified BehaviorSwitchControl to start or stop the behavior)	1078
BehaviorSwitchControl.h (Defines BehaviorSwitchControl and the BehaviorSwitch namespace - a control for turning behaviors on and off)	1079
CameraBehavior.h (Defines CameraBehavior , for taking pictures)	1081
ChaseBallBehavior.cc (Implements ChaseBallBehavior , which runs around after whatever the dog sees)	1083
ChaseBallBehavior.h (Describes ChaseBallBehavior , which runs around after whatever the dog sees)	1085
CompareTrans.h (Defines CompareTrans , which causes a transition if a value (through a pointer) goes above a given value)	1087
Config.cc (Implements Config , which provides global access to system configuration information)	1089
Config.h (Describes Config , which provides global access to system configuration information)	1091
FastSLAM/Configuration.h	1094
Maps/Configuration.h	1097
ControlBase.cc (Implements ControlBase from which all items in the control system should inherit)	1102
ControlBase.h (Defines ControlBase from which all items in the control system should inherit)	1103
Controller.cc (Implements Controller class, a behavior that should be started whenever the emergency stop goes on to provide menus for robot control)	1105
Controller.h (Describes Controller class, a behavior that should be started whenever the emergency stop goes on to provide menus for robot control)	1107
debuget.h (Defines several debugging functions and macros, including AS-SERT (and variations))	1109

DriveMeBehavior.cc (Implements DriveMeBehavior , a very simple behavior that asks the user for WalkMC walking parameters and a walk duration)	1113
DriveMeBehavior.h (Describes DriveMeBehavior , a very simple behavior that asks the user for WalkMC walking parameters and a walk duration)	1115
DumbWM2Behavior.h (Describes DumbWM2Behavior - Simply turns on a WM2 object. Useful for running concurrently with other behaviors and seeing what shows up in the world model)	1117
DumpFileControl.h (Defines DumpFileControl , which when activated, plays a sound selected from the memory stick)	1119
DynamicMotionSequence.h (Uses STL's vector for dynamic memory allocation - don't use this as a motion command, pointers in shared memory regions can be invalid in other processes)	1121
EmergencyStopMC.cc (Implements EmergencyStopMC , overrides all joints, allows modelling, blinks tail)	1123
EmergencyStopMC.h (Describes EmergencyStopMC , overrides all joints, allows modelling, blinks tail)	1125
entry.h	1127
ERS210Info.h (Defines RobotInfo namespace for ERS-210 models, gives some information about the robot's capabilities, such as joint counts, offsets, names and PID values)	1128
ERS220Info.h (Defines RobotInfo namespace for ERS-220 models, gives some information about the robot's capabilities, such as joint counts, offsets, names and PID values)	1130
ERS2xxInfo.h (Defines RobotInfo namespace for ERS-2xx series of robots, such as joint counts, offsets, names and PID values)	1132
EStopControllerBehavior.cc (Implements EStopControllerBehavior , listens to commands coming in from the command port for remotely controlling toggling the estop)	1134
EStopControllerBehavior.h (Describes EStopControllerBehavior , listens to control commands coming in from the command port for remotely toggling the estop)	1135
EventBase.cc (Implements EventBase , the basic class for sending events around the system)	1137
EventBase.h (Describes EventBase , the basic class for sending events around the system)	1138
EventListener.h (Defines EventListener class, an interface for anything that wants to receive events)	1140
EventLogger.cc (Describes EventLogger , which allows logging of events to the console or a file)	1142
EventLogger.h (Describes EventLogger , which allows logging of events to the console or a file)	1143
EventRouter.cc (Implements EventRouter class, for distribution and trapping of events to listeners)	1145

EventRouter.h (Describes EventRouter class, for distribution and trapping of events to listeners)	1147
EventTranslator.cc (Implements EventTranslator , which receives events from EventRouters in non-Main processes and adds them into a Shared-Queue for Main to pick up)	1151
EventTranslator.h (Describes EventTranslator , which receives events from EventRouters in non-Main processes and adds them into a Shared-Queue for Main to pick up)	1153
EventTrapper.h (Defines EventTrapper class, an interface for anything that wants to trap events)	1155
EvtRptBehavior.cc (Implements EvtRptBehavior , which counts information about events it sees)	1157
EvtRptBehavior.h (Describes EvtRptBehavior , which counts information about events it sees)	1158
Factory.h (Defines Factory , a lightweight class to override for constructing new objects)	1159
FileBrowserControl.cc (Implements FileBrowserControl , which displays the contents of a directory)	1160
FileBrowserControl.h (Describes FileBrowserControl , which displays the contents of a directory)	1161
FollowHeadBehavior.cc (Implements FollowHeadBehavior , walks where the head is pointing)	1163
FollowHeadBehavior.h (Describes FollowHeadBehavior , walks where the head is pointing)	1165
FreeMemReportControl.cc (Implements FreeMemReportControl , which gives reports on free memory size at various (configurable) rates)	1167
FreeMemReportControl.h (Describes FreeMemReportControl , which gives reports on free memory size at various (configurable) rates)	1168
Geometry.h (Typedefs commonly used GVector's to vector3d, vector2d, vector3f, and vector2f)	1170
get_time.cc (Implementation of get_time() , a simple way to get the current time since boot in milliseconds)	1173
get_time.h (Prototype for get_time() , a simple way to get the current time since boot in milliseconds)	1175
gvector.h (Vector class to aid in mathematical vector calculations)	1177
HeadLevelBehavior.h (Defines HeadLevelBehavior , which prototypes head leveler)	1183
HeadPointControllerBehavior.cc (Implements HeadPointControllerBehavior , listens to control commands coming in from the command port for remotely controlling the head)	1185
HeadPointControllerBehavior.h (Describes HeadPointControllerBehavior , listens to control commands coming in from the command port for remotely controlling the head)	1186
HeadPointerMC.cc (Implements HeadPointerMC , a class for various ways to control where the head is looking)	1188

HeadPointerMC.h (Describes HeadPointerMC , a class for various ways to control where the head is looking)	1189
HelpControl.cc (Implements HelpControl , which recurses through the menu system and outputs the name and description of each item)	1191
HelpControl.h (Describes HelpControl , which recurses through the menu system and outputs the name and description of each item)	1192
ionetstream.h	1194
karmedbandit.h (Defines karmedbandit - implements an algorithm which makes decisions regarding an adversarial k-armed bandit)	1196
Kinematics.cc (Functions to provide kinematics calculations)	1198
Kinematics.h (Functions to provide kinematics calculations)	1206
LedEngine.cc (Implements LedEngine , which provides basic LED effects to anything that inherits or instantiates it)	1214
LedEngine.h (Describes LedEngine , which provides basic LED effects to anything that inherits or instantiates it)	1215
LedMC.h (Defines LedMC , which provides a basic MotionCommand wrapper to LedEngine)	1217
ListMemBuf.h (Defines ListMemBuf , which provides some degree of dynamic allocation of a templated type from a buffer of set size) . . .	1219
LoadPostureControl.h (Defines LoadPostureControl , which when activated, loads a position from a file name read from cin (stored in ms/data/motion...))	1220
LoadSave.cc (Implements LoadSave , inherit from this to use a standard interface for loading and saving)	1222
LoadSave.h (Describes LoadSave , inherit from this to use a standard interface for loading and saving)	1223
LoadWalkControl.h (Defines LoadWalkControl , which when activated, loads a set of walk parameters from a file read from cin)	1225
LockScope.h (Defines LockScope , which locks a MutexLock until the LockScope goes out of scope)	1227
LocomotionEvent.h (Defines LocomotionEvent , which gives updates regarding the current movement of the robot through the world)	1229
mathutils.h	1231
MCValueEditControl.h (Defines MCValueEditControl , which allows you to modify a value in memory, much like ValueEditControl , but will check out a MotionCommand first to maintain proper mutual exclusion)	1232
MMAccessor.h (Defines MMAccessor , allows convenient ways to check MotionCommands in and out of the MotionManager)	1234
MMCombo.cc (Implements MMCombo , the OObject which "forks" (sort of) into Main and Motion processes)	1236
MMCombo.h (Describes MMCombo , the OObject which "forks" (sort of) into Main and Motion processes)	1239
MotionCommand.cc (Declares the static MotionCommand::queue variable, that's all)	1241

MotionCommand.h (Defines the MotionCommand class, used for creating motions of arbitrary complexity)	1242
MotionManager.cc (Implements MotionManager , simplifies sharing of MotionCommand's and provides mutual exclusion to their access)	1244
MotionManager.h (Describes MotionManager , simplifies sharing of MotionCommand's and provides mutual exclusion to their access)	1246
MotionManagerMsg.h (Defines MotionManagerMsg , a small header used by MotionManager for sending messages between processes)	1250
motionReshapeKludge.h	1251
MotionSequenceMC.cc (Implements MotionSequence , handy little (or not so little) class for switching between a sequence of postures)	1255
MotionSequenceMC.h (Describes MotionSequence and defines MotionSequenceMC , handy little (or not so little) classes for switching between a sequence of postures)	1256
MutexLock.h (Defines MutexLock , a software only mutual exclusion lock)	1258
NullControl.h (Defines NullControl , which does absolutely nothing (handy for fake items in a menu))	1260
OutputCmd.h (Describes OutputCmd , holds information needed to control a single output)	1262
OutputNode.h (Defines OutputNode , a very simple StateNode that outputs its name to a given ostream upon activation, handy for debugging)	1263
OutputPID.h (Describes OutputPID , holds information needed to control a single output)	1265
Path.h (Performs calculations regarding splines for path execution)	1266
PIDMC.h (Defines PIDMC , a nice little MotionCommand for manually manipulating the PID values)	1268
PlaySoundControl.h (Defines PlaySoundControl , which when activated, plays a sound selected from the memory stick)	1270
Poses.h	1272
PostureEngine.cc (Implements PostureEngine , a base class for managing the values and weights of all the outputs)	1275
PostureEngine.h (Describes PostureEngine , a base class for managing the values and weights of all the outputs)	1276
PostureMC.h (Defines PostureMC , a MotionCommand shell for PostureEngine)	1278
ProcessID.cc (Declares the static ProcessID::ID , that's all)	1280
ProcessID.h (Defines ProcessID - simple little global for checking which process is currently running, kind of. (see ProcessID::getID()))	1281
Profiler.cc (Implements Profiler , which manages a hierarchy of timers for profiling time spent in code)	1282
Profiler.h (Describes Profiler , which manages a hierarchy of timers for profiling time spent in code)	1283
ProfilerCheckControl.h (Defines ProfilerCheckControl , which causes the WorldState::mainProfile and WorldState::motionProfile to display reports to cout)	1286

RebootControl.cc (Implements RebootControl , which causes the aibo to reboot (very short - one function separated out to limit recompile of the OPENR headers))	1288
RebootControl.h (Defines RebootControl , which causes the aibo to reboot)	1289
ReferenceCounter.h (Defines the ReferenceCounter base class, which allows for automatic memory deallocation)	1291
RemoteControllerMC.h (Describes RemoteControllerMC , a class for various ways to control where the head is looking)	1293
RemoteProcess.cc (Describes RemoteProcess , sample RemoteProcessing-OPENR process)	1295
RemoteProcess.h (Describes RemoteProcess , sample RemoteProcessing-OPENR process)	1296
RobotInfo.h (Checks the define's to load the appropriate header and namespace)	1298
RunSequenceControl.h (Defines RunSequenceControl , which when activated, loads and runs a motion sequence from a file name read from cin (stored in ms/data/motion))	1300
SavePostureControl.h (Defines SavePostureControl , which when activated, saves the current position to a file name read from user (stored in /ms/data/motion/...))	1302
SaveWalkControl.h (Defines SaveWalkControl , which when activated, saves walk parameters to a file specified from cin)	1304
Serializer.h (Defines the Serializer base class, which provides a default serializer for simple objects)	1306
SharedObject.h (Defines SharedObject , a wrapper for objects in order to facilitate sending them between processes)	1307
SharedQueue.h (Defines SharedQueue , a shared memory message buffer for interprocess communication)	1309
ShutdownControl.cc (Implements ShutdownControl , which causes the aibo to shutdown (very short - one function separated out to limit recompile of the OPENR headers))	1311
ShutdownControl.h (Describes ShutdownControl , which initiates the shutdown sequence)	1312
SimpleChaseBallBehavior.h (Describes SimpleChaseBallBehavior , which runs around after whatever the dog sees)	1314
SmoothCompareTrans.h (Defines SmoothCompareTrans , subclass of CompareTrans , which provides monitoring of exponentially weighted averages to a threshold)	1316
Socket.cc (Implements Tekkotsu wireless Socket class, also sout and serr)	1318
Socket.h (Defines Tekkotsu wireless Socket class, also sout and serr)	1320
SoundManager.cc (Implements SoundManager , which provides sound effects and caching services, as well as mixing buffers for the SoundPlay process)	1322
SoundManager.h (Describes SoundManager , which provides sound effects and caching services, as well as mixing buffers for the SoundPlay process)	1325

SoundManagerMsg.h (Defines SoundManagerMsg , a small header used by SoundManager for sending messages between processes)	1328
SoundPlay.cc (Implements the SoundPlay process (a.k.a. OObject), which is responsible for sending sound buffers to the system to play)	1329
SoundPlay.h (Describes the SoundPlay process (a.k.a. OObject), which is responsible for sending sound buffers to the system to play)	1331
SoundTestBehavior.h (Defines the SoundTestBehavior demo, which allows you to experiment with playing sounds different ways)	1333
Spline.h (Performs calculations regarding splines for path execution)	1335
StareAtBallBehavior.cc (Implements StareAtBallBehavior , which points the head at the ball)	1338
StareAtBallBehavior.h (Describes StareAtBallBehavior , which runs around after whatever the dog sees)	1339
StartupBehavior.cc	1341
StartupBehavior.h	1345
StateNode.cc (Describes StateNode , which is both a state machine controller as well as a node within a state machine itself)	1346
StateNode.h (Describes StateNode , which is both a state machine controller as well as a node within a state machine itself)	1347
StringInputControl.cc (Implements StringInputControl , which prompts for and stores a string from the user)	1349
StringInputControl.h (Defines StringInputControl , which prompts for and stores a string from the user)	1350
SystemUtility.h (Wrappers for getting large memory regions from Aperios) .	1352
TailWagMC.h (Defines TailWagMC , which will wag the tail on a ERS-210 robot)	1354
Test.c	1356
TextMsgEvent.h (Defines TextMsgEvent , which extends EventBase to also include actual message text)	1358
TimeET.cc (Implements TimeET , a nice class for handling time values with high precision (but all that's in the .cc is implementation of struct timezone TimeET::tz))	1360
TimeET.h (Describes TimeET , a nice class for handling time values with high precision)	1361
TimeOutTrans.h (Defines TimeOutTrans , which causes a transition after a specified amount of time has passed)	1364
ToggleHeadLightBehavior.h (Defines ToggleHeadLightBehavior , which will open or close the head light on an ERS-220)	1366
Transition.cc (Implements Transition , represents a transition between State-Nodes)	1368
Transition.h (Describes Transition , represents a transition between State-Nodes)	1369
Util.h (Numerical Utilities)	1370
ValueEditControl.h (Defines ValueEditControl class, which will allow modification of a value through a pointer)	1375

ValueSetControl.h (Defines ValueSetControl class, which will assign a value through a pointer upon activation)	1377
Vision.cc (Does majority of vision processing)	1379
Vision.h (Does majority of vision processing)	1383
Visiondefines.h	1388
VisionEvent.h (Provides information about objects recognized in the camera image)	1391
VisionInterface.h (Interfaces between CMVision and the Vision module) . .	1393
VisionSerializer.cc (Implements VisionSerializer , which encodes and transmits camera images)	1395
VisionSerializer.h (Describes VisionSerializer , which encodes and transmits camera images)	1397
VisualTargetCloseTrans.h (Defines VisualTargetCloseTrans , which causes a transition when a visual object is "close")	1399
WalkControllerBehavior.cc (Implements WalkControllerBehavior , listens to mecha control commands coming in from the command port for remotely controlling the walk)	1401
WalkControllerBehavior.h (Describes WalkControllerBehavior , listens to control commands coming in from the command port for remotely controlling the walk)	1402
WalkMC.cc (Implements WalkMC , a MotionCommand for walking around)	1404
WalkMC.h (Describes WalkMC , a MotionCommand for walking around) . .	1407
WalkMotionModel.cc	1410
WalkMotionModel.h	1411
WalkToTargetMachine.cc (Implements WalkToTargetMachine , a state machine for walking towards a visual target)	1412
WalkToTargetMachine.h (Describes WalkToTargetMachine , a state machine for walking towards a visual target)	1414
WAV.cc (Allows you to load WAV files from the memory stick)	1416
WAV.h (Allows you to load WAV files from the memory stick)	1418
Wireless.cc (Interacts with the system to provide networking services) . . .	1420
Wireless.h (Interacts with the system to provide networking services) . . .	1423
WorldModel2.cc	1426
WorldModel2.h	1429
WorldModel2Behavior.cc	1432
WorldModel2Behavior.h	1434
WorldState.cc (Implements WorldState , maintains information about the robot's environment, namely sensors and power status)	1436
WorldState.h (Describes WorldState , maintains information about the robot's environment, namely sensors and power status)	1439
WorldStateSerializer.cc	1443
WorldStateSerializer.h	1444

Chapter 5

Tekkotsu Page Index

5.1 Tekkotsu Related Pages

Here is a list of all related documentation pages:

Todo List	1445
Test List	1448

Chapter 6

Tekkotsu Namespace Documentation

6.1 ButtonSourceID Namespace Reference

6.1.1 Detailed Description

holds source ID types for button events [EventBase](#) ButtonSourceID.t

Enumerations

- enum [ButtonSourceID.t](#) {
 [LFrPawSID](#) = LFrPawOffset, [RFrPawSID](#) = RFrPawOffset, [LBkPawSID](#) = LBkPawOffset, [RBkPawSID](#) = RBkPawOffset,
 [ChinButSID](#) = ChinButOffset, [BackButSID](#), [HeadFrButSID](#), [HeadBkButSID](#),
 [TailLeftButSID](#), [TailCenterButSID](#), [TailRightButSID](#) }
 holds source ID types for button events

6.1.2 Enumeration Type Documentation

6.1.2.1 enum [ButtonSourceID::ButtonSourceID.t](#)

holds source ID types for button events

Isn't necessarily a straight mapping to ButtonOffset.t's, depends on model (210 doesn't have tail buttons)

See also:

ButtonOffset_t

Enumeration values:

LFrPawSID

RFrPawSID

LBkPawSID

RBkPawSID

ChinButSID

BackButSID

HeadFrButSID

HeadBkButSID

TailLeftButSID

TailCenterButSID

TailRightButSID

Definition at line 30 of file WorldState.h.

6.2 ERS210Info Namespace Reference

6.2.1 Detailed Description

Contains information about the ERS-210 Robot, such as number of joints, PID defaults, timing information, etc.

LED Bitmasks

Bitmasks for use when specifying combinations of LEDs (see LEDEngine) Note that L/R are robot's POV

- typedef unsigned int [LEDBitMask_t](#)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) BotLLEDMask = 1<<(BotLLEDOffset-LEDOffset)
bottom left (red - sad)
- const [LEDBitMask_t](#) BotRLEDMask = 1<<(BotRLEDOffset-LEDOffset)
bottom right (red - sad)
- const [LEDBitMask_t](#) MidLLEDMask = 1<<(MidLLEDOffset-LEDOffset)
middle left (green - happy)
- const [LEDBitMask_t](#) MidRLEDMask = 1<<(MidRLEDOffset-LEDOffset)
middle right (green - happy)
- const [LEDBitMask_t](#) TopLLEDMask = 1<<(TopLLEDOffset-LEDOffset)
top left (red - angry)
- const [LEDBitMask_t](#) TopRLEDMask = 1<<(TopRLEDOffset-LEDOffset)
top right (red - angry)
- const [LEDBitMask_t](#) TopBrLEDMask = 1<<(TopBrLEDOffset-LEDOffset)
top bar (green)
- const [LEDBitMask_t](#) TIRedLEDMask = 1<<(TIRedLEDOffset-LEDOffset)
red tail light
- const [LEDBitMask_t](#) TIBluLEDMask = 1<<(TIBluLEDOffset-LEDOffset)
blue tail light

- const [LEDBitMask_t](#) FaceLEDMask = BotLLEDMask|BotRLEDMask|MidLLEDMask|MidRLEDMask|TopLLEDMask|TopRLEDMask|TopBrLEDMask
LEDs for face (all but tail).
- const [LEDBitMask_t](#) HeadLEDMask = BotLLEDMask|BotRLEDMask|MidLLEDMask|MidRLEDMask|TopLLEDMask|TopRLEDMask|TopBrLEDMask
LEDs for face (all but tail).
- const [LEDBitMask_t](#) BackLEDMask = 0
210 has no back LEDs
- const [LEDBitMask_t](#) TailLEDMask = TIRedLEDMask|TIBluLEDMask
LEDs on tail.
- const [LEDBitMask_t](#) AllLEDMask = ~0
selects all of the leds

Input Offsets

The order in which inputs should be stored

- enum [ButtonOffset_t](#) {
[LFrPawOffset](#) = LFrLegOrder, [RFrPawOffset](#) = RFrLegOrder, [LBkPawOffset](#) = LBkLegOrder, [RBkPawOffset](#) = RBkLegOrder,
[ChinButOffset](#) = 4, [BackButOffset](#), [HeadFrButOffset](#), [HeadBkButOffset](#) }
holds offsets to different buttons in [WorldState::buttons\[\]](#)
- enum [SensorOffset_t](#) {
[IRDistOffset](#) = 0, [BAccelOffset](#), [LAccelOffset](#), [DAccelOffset](#),
[ThermoOffset](#), [PowerRemainOffset](#), [PowerThermoOffset](#), [PowerCapacityOffset](#),
[PowerVoltageOffset](#), [PowerCurrentOffset](#) }
holds offset to different sensor values in [WorldState::sensors\[\]](#)

Output Types Information

- const unsigned [NumPIDJoints](#) = 18

The number of joints which use PID motion - everything except ears.

- const unsigned NumLEDs = 9

The number LEDs which can be controlled.

- const unsigned NumBinJoints = 2

The number of binary joints - just the ears.

- const unsigned NumOutputs = NumPIDJoints + NumBinJoints + NumLEDs

the total number of outputs

- [illegible]

we need this so you can tell programmatically which joints are "real" and which are "fake" in compatability mode

Output Offsets

Corresponds to entries in PrimitiveName, defined at the end of this file

- const unsigned **PIDJointOffset** = 0

The beginning of the PID Joints.

- const unsigned **LegOffset** = PIDJointOffset

the offset of the beginning of the leg joints

- const unsigned HeadOffset = LegOffset+NumLegJoints

the offset of the beginning of the head joints

- const unsigned TailOffset = HeadOffset+NumHeadJoints

the offset of the beginning of the tail joints

- const unsigned MouthOffset = TailOffset+NumTailJoints

the offset of the beginning of the mouth joint

- const unsigned LEDOffset = PIDJointOffset + NumPIDJoints

the offset of LEDs in `WorldState::outputs` and `MotionCommand` functions

- const unsigned [BinJointOffset](#) = [LEDOffset](#) + [NumLEDs](#)

The beginning of the binary joints.

- const unsigned [EarOffset](#) = [BinJointOffset](#)

the offset of the beginning of the ear joints - note that ears aren't sensed. They can be flicked by the environment and you won't know. Nor will they be flicked back

CPC IDs

values defined by OPEN-R, used to interface with lower level OPEN-R code to read sensors - DOESN'T correspond to ::PrimitiveName

- const int [CPCJointNeckTilt](#) = 0

Head.

- const int [CPCJointNeckPan](#) = 1

Head.

- const int [CPCJointNeckRoll](#) = 2

Head.

- const int [CPCSensorHeadBackPressure](#) = 3

Head.

- const int [CPCSensorHeadFrontPressure](#) = 4

Head.

- const int [CPCSensorPSD](#) = 5

Head.

- const int [CPCJointMouth](#) = 6

Head.

- const int [CPCSensorChinSwitch](#) = 7

Head.

- const int [CPCJointLFRotator](#) = 8

Left front leg.

- const int [CPCJointLFElevator](#) = 9

Head.

- const int [CPCJointLFKnee](#) = 10

Head.

- const int [CPCSensorLFPaw](#) = 11

Head.

- const int [CPCJointLHRotator](#) = 12

Left hind leg.

- const int [CPCJointLHElevator](#) = 13

Head.

- const int [CPCJointLHKnee](#) = 14

Head.

- const int [CPCSensorLHPaw](#) = 15

Head.

- const int [CPCJointRFRotator](#) = 16

Right front leg.

- const int [CPCJointRFElevator](#) = 17

Head.

- const int [CPCJointRFKnee](#) = 18

Head.

- const int [CPCSensorRFPaw](#) = 19

Head.

- const int [CPCJointRHRotator](#) = 20

Right hind leg.

- const int [CPCJointRHElevator](#) = 21

Head.

- const int [CPCJointRHKnee](#) = 22

Head.

- const int [CPCSensorRHPaw](#) = 23

Head.

- const int [CPCJointTailPan](#) = 24

Tail.

- const int [CPCJointTailTilt](#) = 25

Head.

- const int [CPCSensorThermoSensor](#) = 26

Head.

- const int [CPCSensorBackSwitch](#) = 27

Head.

- const int [CPCSensorAccelFB](#) = 28

Front-back RobotInfo::BAccelOffset.

- const int [CPCSensorAccelLR](#) = 29

Left-right RobotInfo::LAccelOffset.

- const int [CPCSensorAccelUD](#) = 30

Up-down RobotInfo::DAccelOffset.

Enumerations

- enum [LegOrder.t](#) { [LFrLegOrder](#) = 0, [RFrLegOrder](#), [LBkLegOrder](#), [RBkLegOrder](#) }

the ordering of legs

- enum [LegOffset.t](#) { [LFrLegOffset](#) = LegOffset+LFrLegOrder*JointsPerLeg, [RFrLegOffset](#) = LegOffset+RFrLegOrder*JointsPerLeg, [LBkLegOffset](#) = LegOffset+LBkLegOrder*JointsPerLeg, [RBkLegOffset](#) = LegOffset+RBkLegOrder*JointsPerLeg }

The offsets of the individual legs.

- enum [REKOffset.t](#) { [RotatorOffset](#) = 0, [ElevatorOffset](#), [KneeOffset](#) }

*The offsets within appendages (the legs) Note that the ordering matches the actual physical ordering of joints on the appendage (and not that of the head's TPROffset-
t's).*

- enum [TPROffset.t](#) { [TiltOffset](#) = 0, [PanOffset](#), [RollOffset](#) }

The offsets of appendages with tilt (elevation), pan (heading), and roll joints (i.e. head) Note that the ordering matches the actual physical ordering of joints on the appendage (and not that of the leg's REKOffset_t's).

- enum `LEDOffset_t` {
`BotLLEDOffset` = `LEDOffset`, `BotRLEDOffset`, `MidLLEDOffset`, `MidRLEDOffset`,
`TopLLEDOffset`, `TopRLEDOffset`, `TopBrLEDOffset`, `TIRedLEDOffset`,
`TIBluLEDOffset`, `FaceFrontLeftLEDOffset` = `BotLLEDOffset`, `FaceFrontRightLEDOffset` = `BotRLEDOffset`, `FaceCenterLeftLEDOffset` = `MidLLEDOffset`,
`FaceCenterRightLEDOffset` = `MidRLEDOffset`, `FaceBackLeftLEDOffset` = `TopLLEDOffset`, `FaceBackRightLEDOffset` = `TopRLEDOffset`, `ModeLEDOffset` = `TopBrLEDOffset`,
`TailRightLEDOffset` = `TIRedLEDOffset`, `TailLeftLEDOffset` = `TIBluLEDOffset`
}

The offsets of the individual LEDs on the head and tail. Note that L/R are robot's POV. See also LEDBitMask_t.

- enum `MinMaxRange_t` { `MinRange`, `MaxRange` }
Defines the min and max index of entries in #outputRanges and #mechanicalLimits.

Variables

- const unsigned int `FrameTime` = 8
time between frames in the motion system (milliseconds)
- const unsigned int `NumFrames` = 4
the number of frames per buffer (don't forget also double buffered)
- const unsigned int `SlowFrameTime` = 128
time between frames for the ears (which move slower for some reason, don't want to mix with other outputs) (milliseconds)
- const unsigned int `NumSlowFrames` = 1
the number of frames per buffer being sent to ears (double buffered as well)
- const unsigned int `SoundBufferTime` = 32
the number of milliseconds per sound buffer... I'm not sure if this can be changed
- const unsigned `JointsPerLeg` = 3
The number of joints per leg.

- const unsigned `NumLegs` = 4
The number of legs.
- const unsigned `NumLegJoints` = `JointsPerLeg*NumLegs`
the TOTAL number of joints on ALL legs
- const unsigned `NumHeadJoints` = 3
The number of joints in the neck.
- const unsigned `NumTailJoints` = 2
The number of joints assigned to the tail.
- const unsigned `NumMouthJoints` = 1
the number of joints that control the mouth
- const unsigned `NumEarJoints` = 2
The number of joints which control the ears (NOT per ear, is total).
- const unsigned `NumButtons` = 8
the number of buttons that are available, see `ButtonOffset_t`
- const unsigned `NumSensors` = 1+3+1+5
1 dist, 3 accel, 1 thermo, 5 from power, see `SensorOffset_t`
- const unsigned `outputNameLen` = 9
The length of the strings used for each of the outputs in `outputNames` (doesn't include null term).
- const char *const `outputNames` [`NumOutputs`]
A name of uniform length for referring to joints - handy for posture files, etc.
- const char *const `PrimitiveName` [`NumOutputs`]
the joint identifier strings used to refer to specific joints in OPEN-R (but not needed for others)
- const float `DefaultPIDs` [`NumPIDJoints`][3]
This table holds the default PID values for each joint. `PIDMC`.
- const float `MaxOutputSpeed` [`NumOutputs`]
These values are Sony's recommended maximum joint velocities, in rad/ms.
- const double `outputRanges` [`NumOutputs`][2]

This table holds the software limits of each of the outputs.

- const double [mechanicalLimits](#) [NumOutputs][2]

This table holds the mechanical limits of each of the outputs.

6.2.2 Typedef Documentation

6.2.2.1 typedef unsigned int [ERS210Info::LEDBitMask_t](#)

So you can be clear when you're referring to a LED bitmask.

Definition at line 133 of file ERS210Info.h.

6.2.3 Enumeration Type Documentation

6.2.3.1 enum [ERS210Info::ButtonOffset_t](#)

holds offsets to different buttons in [WorldState::buttons](#)[]

Should be a straight mapping to the ButtonSourceIDs

Note that the chest (power) button is not a normal button. It kills power to the motors at a hardware level, and isn't sensed in the normal way. If you want to know when it is pressed (and you are about to shut down) see [PowerSourceID::PauseSID](#).

See also:

[WorldState::buttons](#)

ButtonSourceID_t

Enumeration values:

LFrPawOffset

RFrPawOffset

LBkPawOffset

RBkPawOffset

ChinButOffset

BackButOffset

HeadFrButOffset not in reliable pressure units, but 1.0 is fairly stiff pressure, 0 is none

HeadBkButOffset not in reliable pressure units, but 1.0 is fairly stiff pressure, 0 is none

Definition at line 169 of file ERS210Info.h.

6.2.3.2 enum [ERS210Info::LEDOffset_t](#)

The offsets of the individual LEDs on the head and tail. Note that L/R are robot's POV. See also LEDBitMask_t.

Enumeration values:

- BotLLEDOffset** bottom left (red - sad)
- BotRLEDOffset** bottom right (red - sad)
- MidLLEDOffset** middle left (green - happy)
- MidRLEDOffset** middle right (green - happy)
- TopLLEDOffset** top left (red - angry)
- TopRLEDOffset** top right (red - angry)
- TopBrLEDOffset** top bar (green)
- TIRedLEDOffset** red tail light
- TIBluLEDOffset** blue tail light
- FaceFrontLeftLEDOffset** alias for 220 cross-compatibility
- FaceFrontRightLEDOffset** alias for 220 cross-compatibility
- FaceCenterLeftLEDOffset** alias for 220 cross-compatibility
- FaceCenterRightLEDOffset** alias for 220 cross-compatibility
- FaceBackLeftLEDOffset** alias for 220 cross-compatibility
- FaceBackRightLEDOffset** alias for 220 cross-compatibility
- ModeLEDOffset** alias for 220 cross-compatibility
- TailRightLEDOffset** alias for 220 cross-compatibility
- TailLeftLEDOffset** alias for 220 cross-compatibility

Definition at line 108 of file ERS210Info.h.

6.2.3.3 enum [ERS210Info::LegOffset_t](#)

The offsets of the individual legs.

Enumeration values:

- LFrLegOffset** beginning of left front leg
- RFrLegOffset** beginning of right front leg
- LBkLegOffset** beginning of left back leg
- RBkLegOffset** beginning of right back leg

Definition at line 86 of file ERS210Info.h.

6.2.3.4 enum [ERS210Info::LegOrder_t](#)

the ordering of legs

Enumeration values:

- LFrLegOrder** left front leg
- RFrLegOrder** right front leg
- LBkLegOrder** left back leg
- RBkLegOrder** right back leg

Definition at line 78 of file ERS210Info.h.

6.2.3.5 enum [ERS210Info::MinMaxRange_t](#)

Defines the min and max index of entries in [outputRanges](#) and [mechanicalLimits](#).

Enumeration values:

- MinRange**
- MaxRange**

Definition at line 378 of file ERS210Info.h.

6.2.3.6 enum [ERS210Info::REKOffset_t](#)

The offsets within appendages (the legs) Note that the ordering matches the actual physical ordering of joints on the appendage (and not that of the head's TPROffset_t's).

Enumeration values:

- RotatorOffset** moves leg forward or backward along body
- ElevatorOffset** moves leg toward or away from body
- KneeOffset** moves knee

Definition at line 94 of file ERS210Info.h.

6.2.3.7 enum [ERS210Info::SensorOffset_t](#)

holds offset to different sensor values in [WorldState::sensors\[\]](#)

See also:

[WorldState::sensors\[\]](#)

Enumeration values:**IRDistOffset** in millimeters**BAccelOffset** backward acceleration, in m/s^2 , negative if sitting on butt (positive for faceplant)**LAccelOffset** acceleration to the robot's left, in m/s^2 , negative if lying on robot's left side**DAccelOffset** downward acceleration, in m/s^2 , negative if standing up... be careful about the signs on all of these...**ThermoOffset** in degrees Celcius**PowerRemainOffset** percentage, 0-1**PowerThermoOffset** degrees Celcius**PowerCapacityOffset** milli-amp hours**PowerVoltageOffset** volts**PowerCurrentOffset** milli-amp negative values (maybe positive while charging?)

Definition at line 182 of file ERS210Info.h.

6.2.3.8 enum [ERS210Info::TPROffset_t](#)

The offsets of appendages with tilt (elevation), pan (heading), and roll joints (i.e. head)
 Note that the ordering matches the actual physical ordering of joints on the appendage
 (and not that of the leg's REKOffset.t's).

Enumeration values:**TiltOffset** tilt/elevation (vertical)**PanOffset** pan/heading (horizontal)**RollOffset** roll (rotational)

Definition at line 101 of file ERS210Info.h.

6.2.4 Variable Documentation**6.2.4.1 const [LEDBitMask_t](#) [ERS210Info::AllLEDMask](#) = ~0**

selects all of the leds

Definition at line 148 of file ERS210Info.h.

6.2.4.2 `const LEDBitMask_t ERS210Info::BackLEDMask = 0`

210 has no back LEDs

Definition at line 146 of file ERS210Info.h.

6.2.4.3 `const unsigned ERS210Info::BinJointOffset = LEDOffset + NumLEDs`

The beginning of the binary joints.

Definition at line 73 of file ERS210Info.h.

6.2.4.4 `const LEDBitMask_t ERS210Info::BotLLEDMask = 1<<(BotLLEDOffset-LEDOffset)`

bottom left (red - sad)

Definition at line 134 of file ERS210Info.h.

6.2.4.5 `const LEDBitMask_t ERS210Info::BotRLEDMask = 1<<(BotRLEDOffset-LEDOffset)`

bottom right (red - sad)

Definition at line 135 of file ERS210Info.h.

6.2.4.6 `const int ERS210Info::CPCJointLFElevator = 9 [static]`

Head.

Definition at line 436 of file ERS210Info.h.

6.2.4.7 `const int ERS210Info::CPCJointLFKnee = 10 [static]`

Head.

Definition at line 437 of file ERS210Info.h.

6.2.4.8 `const int ERS210Info::CPCJointLFRotator = 8 [static]`

Left front leg.

Definition at line 435 of file ERS210Info.h.

6.2.4.9 `const int ERS210Info::CPCJointLHElevator = 13` [static]

Head.

Definition at line 440 of file ERS210Info.h.

6.2.4.10 `const int ERS210Info::CPCJointLHKnee = 14` [static]

Head.

Definition at line 441 of file ERS210Info.h.

6.2.4.11 `const int ERS210Info::CPCJointLHRotator = 12` [static]

Left hind leg.

Definition at line 439 of file ERS210Info.h.

6.2.4.12 `const int ERS210Info::CPCJointMouth = 6` [static]

Head.

Definition at line 433 of file ERS210Info.h.

6.2.4.13 `const int ERS210Info::CPCJointNeckPan = 1` [static]

Head.

Definition at line 428 of file ERS210Info.h.

6.2.4.14 `const int ERS210Info::CPCJointNeckRoll = 2` [static]

Head.

Definition at line 429 of file ERS210Info.h.

6.2.4.15 `const int ERS210Info::CPCJointNeckTilt = 0` [static]

Head.

Definition at line 427 of file ERS210Info.h.

6.2.4.16 `const int ERS210Info::CPCJointRFElevator = 17` [static]

Head.

Definition at line 444 of file ERS210Info.h.

6.2.4.17 `const int ERS210Info::CPCJointRFKnee = 18` [static]

Head.

Definition at line 445 of file ERS210Info.h.

6.2.4.18 `const int ERS210Info::CPCJointRFRotator = 16` [static]

Right front leg.

Definition at line 443 of file ERS210Info.h.

6.2.4.19 `const int ERS210Info::CPCJointRHElevator = 21` [static]

Head.

Definition at line 448 of file ERS210Info.h.

6.2.4.20 `const int ERS210Info::CPCJointRHKnee = 22` [static]

Head.

Definition at line 449 of file ERS210Info.h.

6.2.4.21 `const int ERS210Info::CPCJointRHRotator = 20` [static]

Right hind leg.

Definition at line 447 of file ERS210Info.h.

6.2.4.22 `const int ERS210Info::CPCJointTailPan = 24` [static]

Tail.

Definition at line 451 of file ERS210Info.h.

6.2.4.23 `const int ERS210Info::CPCJointTailTilt = 25` `[static]`

Head.

Definition at line 452 of file ERS210Info.h.

6.2.4.24 `const int ERS210Info::CPCSensorAccelFB = 28` `[static]`

Front-back RobotInfo::BAccelOffset.

Definition at line 455 of file ERS210Info.h.

6.2.4.25 `const int ERS210Info::CPCSensorAccelLR = 29` `[static]`

Left-right RobotInfo::LAccelOffset.

Definition at line 456 of file ERS210Info.h.

6.2.4.26 `const int ERS210Info::CPCSensorAccelUD = 30` `[static]`

Up-down RobotInfo::DAccelOffset.

Definition at line 457 of file ERS210Info.h.

6.2.4.27 `const int ERS210Info::CPCSensorBackSwitch = 27` `[static]`

Head.

Definition at line 454 of file ERS210Info.h.

6.2.4.28 `const int ERS210Info::CPCSensorChinSwitch = 7` `[static]`

Head.

Definition at line 434 of file ERS210Info.h.

6.2.4.29 `const int ERS210Info::CPCSensorHeadBackPressure = 3` `[static]`

Head.

Definition at line 430 of file ERS210Info.h.

6.2.4.30 `const int ERS210Info::CPCSensorHeadFrontPressure = 4` [static]

Head.

Definition at line 431 of file ERS210Info.h.

6.2.4.31 `const int ERS210Info::CPCSensorLFPaw = 11` [static]

Head.

Definition at line 438 of file ERS210Info.h.

6.2.4.32 `const int ERS210Info::CPCSensorLHPaw = 15` [static]

Head.

Definition at line 442 of file ERS210Info.h.

6.2.4.33 `const int ERS210Info::CPCSensorPSD = 5` [static]

Head.

Definition at line 432 of file ERS210Info.h.

6.2.4.34 `const int ERS210Info::CPCSensorRFPaw = 19` [static]

Head.

Definition at line 446 of file ERS210Info.h.

6.2.4.35 `const int ERS210Info::CPCSensorRHPaw = 23` [static]

Head.

Definition at line 450 of file ERS210Info.h.

6.2.4.36 `const int ERS210Info::CPCSensorThermoSensor = 26` [static]

Head.

Definition at line 453 of file ERS210Info.h.

6.2.4.37 `const float ERS210Info::DefaultPIDs[NumPIDJoints][3]`

Initial value:

```

{
  { 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
  { 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
  { 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },
  { 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
  { 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
  { 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },
  { 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
  { 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
  { 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },
  { 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
  { 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
  { 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },

  { 0x0A/(double)(1<<(16-0xE)), 0x08/(double)(1<<(16-0x2)), 0x0C/(double)(1<<(16-0xF)) },
  { 0x0D/(double)(1<<(16-0xE)), 0x08/(double)(1<<(16-0x2)), 0x0B/(double)(1<<(16-0xF)) },
  { 0x0A/(double)(1<<(16-0xE)), 0x08/(double)(1<<(16-0x2)), 0x0C/(double)(1<<(16-0xF)) },

  { 0x0A/(double)(1<<(16-0xE)), 0x00/(double)(1<<(16-0x2)), 0x18/(double)(1<<(16-0xF)) },
  { 0x07/(double)(1<<(16-0xE)), 0x00/(double)(1<<(16-0x2)), 0x11/(double)(1<<(16-0xF)) },

  { 0x0E/(double)(1<<(16-0xE)), 0x08/(double)(1<<(16-0x2)), 0x10/(double)(1<<(16-0xF)) }
}

```

This table holds the default PID values for each joint. [PIDMC](#).

Definition at line 315 of file ERS210Info.h.

6.2.4.38 const unsigned [ERS210Info::EarOffset](#) = [BinJointOffset](#)

the offset of the beginning of the ear joints - note that ears aren't sensed. They can be flicked by the environment and you won't know. Nor will they be flicked back

Definition at line 74 of file ERS210Info.h.

6.2.4.39 const [LEDBitMask_t](#) [ERS210Info::FaceLEDMask](#) = [BotLLEDMask|BotRLEDMask|MidLLEDMask|MidRLEDMask|Top- LLEDMask|TopRLEDMask|TopBrLEDMask](#)

LEDs for face (all but tail).

Definition at line 144 of file ERS210Info.h.

6.2.4.40 const unsigned int [ERS210Info::FrameTime](#) = 8

time between frames in the motion system (milliseconds)

Definition at line 27 of file ERS210Info.h.

6.2.4.41 `const LEDBitMask_t ERS210Info::HeadLEDMask =
BotLLEDMask|BotRLEDMask|MidLLEDMask|MidRLEDMask|Top-
LLEDMask|TopRLEDMask|TopBrLEDMask`

LEDs for face (all but tail).

Definition at line 145 of file ERS210Info.h.

6.2.4.42 `const unsigned ERS210Info::HeadOffset = LegOffset+NumLegJoints`

the offset of the beginning of the head joints

Definition at line 67 of file ERS210Info.h.

6.2.4.43 `const bool ERS210Info::IsFastOutput[NumOutputs] = {
true,
}`

true for joints which can be updated every 32 ms (all but the ears)

Definition at line 41 of file ERS210Info.h.

6.2.4.44 `const bool ERS210Info::IsRealERS210[NumOutputs] = {
true,
}`

we need this so you can tell programmatically which joints are "real" and which are "fake" in compatability mode

Definition at line 43 of file ERS210Info.h.

6.2.4.45 `const unsigned ERS210Info::JointsPerLeg = 3`

The number of joints per leg.

Definition at line 46 of file ERS210Info.h.

6.2.4.46 `const unsigned ERS210Info::LEDOffset = PIDJointOffset +
NumPIDJoints`

the offset of LEDs in `WorldState::outputs` and `MotionCommand` functions

Definition at line 71 of file ERS210Info.h.

6.2.4.47 `const unsigned` [ERS210Info::LegOffset](#) = [PIDJointOffset](#)

the offset of the beginning of the leg joints

Definition at line 66 of file ERS210Info.h.

6.2.4.48 `const float` [ERS210Info::MaxOutputSpeed](#)[[NumOutputs](#)]

Initial value:

```
{
    2.8143434e-03,
    2.4980025e-03,
    2.8361600e-03,
    2.8143434e-03,
    2.4980025e-03,
    2.8361600e-03,
    2.8143434e-03,
    2.4980025e-03,
    2.8361600e-03,
    2.8143434e-03,
    2.4980025e-03,
    2.8361600e-03,

    2.1053034e-03,
    3.0106930e-03,
    3.0106930e-03,

    4.4724062e-03,
    4.4724062e-03,

    4.3742314e-03,

    0,0,0,0,0,0,0,0,0,

    0,0
}
```

These values are Sony's recommended maximum joint velocities, in rad/ms.

a value ≤ 0 means infinite speed (e.g. LEDs)

Definition at line 342 of file ERS210Info.h.

6.2.4.49 `const double` [ERS210Info::mechanicalLimits](#)[[NumOutputs](#)][2]

Initial value:

```
{
    { RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },
    { RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },
}
```



```

{ RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },
{ RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },

{ RAD(-85),RAD(46) }, { RAD(-92.6),RAD(92.6) }, { RAD(-32),RAD(32) },

{ RAD(-25),RAD(25) }, { RAD(-25),RAD(25) },

{ RAD(-50),RAD(0) },

{ 0,1 }, { 0,1 }, { 0,1 }, { 0,1 }, { 0,1 }, { 0,1 }, { 0,1 }, { 0,1 }, { 0,1 },
{ 0,1 }, { 0,1 }
}

```

This table holds the mechanical limits of each of the outputs.

Definition at line 400 of file ERS210Info.h.

6.2.4.50 `const LEDBitMask_t ERS210Info::MidLLEDMask = 1<<(MidLEDOffset-LEDOffset)`

middle left (green - happy)

Definition at line 136 of file ERS210Info.h.

6.2.4.51 `const LEDBitMask_t ERS210Info::MidRLEDMask = 1<<(MidRLEDOffset-LEDOffset)`

middle right (green - happy)

Definition at line 137 of file ERS210Info.h.

6.2.4.52 `const unsigned ERS210Info::MouthOffset = TailOffset+NumTailJoints`

the offset of the beginning of the mouth joint

Definition at line 69 of file ERS210Info.h.

6.2.4.53 `const unsigned ERS210Info::NumBinJoints = 2`

The number of binary joints - just the ears.

Definition at line 38 of file ERS210Info.h.

6.2.4.54 `const unsigned ERS210Info::NumButtons = 8`

the number of buttons that are available, see ButtonOffset.t

Definition at line 53 of file ERS210Info.h.

6.2.4.55 `const unsigned ERS210Info::NumEarJoints = 2`

The number of joints which control the ears (NOT per ear, is total).

Definition at line 52 of file ERS210Info.h.

6.2.4.56 `const unsigned int ERS210Info::NumFrames = 4`

the number of frames per buffer (don't forget also double buffered)

Definition at line 28 of file ERS210Info.h.

6.2.4.57 `const unsigned ERS210Info::NumHeadJoints = 3`

The number of joints in the neck.

Definition at line 49 of file ERS210Info.h.

6.2.4.58 `const unsigned ERS210Info::NumLEDs = 9`

The number LEDs which can be controlled.

Definition at line 37 of file ERS210Info.h.

6.2.4.59 `const unsigned ERS210Info::NumLegJoints = JointsPerLeg*NumLegs`

the TOTAL number of joints on ALL legs

Definition at line 48 of file ERS210Info.h.

6.2.4.60 `const unsigned ERS210Info::NumLegs = 4`

The number of legs.

Definition at line 47 of file ERS210Info.h.

6.2.4.61 `const unsigned ERS210Info::NumMouthJoints = 1`

the number of joints that control the mouth

Definition at line 51 of file ERS210Info.h.

6.2.4.62 `const unsigned ERS210Info::NumOutputs = NumPIDJoints + NumBinJoints + NumLEDs`

the total number of outputs

Definition at line 39 of file ERS210Info.h.

6.2.4.63 `const unsigned ERS210Info::NumPIDJoints = 18`

The number of joints which use PID motion - everything except ears.

Right now all binary joints are slow, but perhaps this won't always be the case... hence the IsFast/Slow bitmasks to select which type, in order to be more general

Definition at line 36 of file ERS210Info.h.

6.2.4.64 `const unsigned ERS210Info::NumSensors = 1+3+1+5`

1 dist, 3 accel, 1 thermo, 5 from power, see SensorOffset.t

Definition at line 54 of file ERS210Info.h.

6.2.4.65 `const unsigned int ERS210Info::NumSlowFrames = 1`

the number of frames per buffer being sent to ears (double buffered as well)

Definition at line 30 of file ERS210Info.h.

6.2.4.66 `const unsigned ERS210Info::NumTailJoints = 2`

The number of joints assigned to the tail.

Definition at line 50 of file ERS210Info.h.

6.2.4.67 `const unsigned ERS210Info::outputNameLen = 9`

The length of the strings used for each of the outputs in outputNames (doesn't include null term).

Definition at line 199 of file ERS210Info.h.

6.2.4.68 `const char* const ERS210Info::outputNames[NumOutputs]`

A name of uniform length for referring to joints - handy for posture files, etc.

Definition at line 201 of file ERS210Info.h.

6.2.4.69 `const double ERS210Info::outputRanges[NumOutputs][2]`**Initial value:**

```
{
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },

  { RAD(-82),RAD(43) }, { RAD(-89.6),RAD(89.6) }, { RAD(-29),RAD(29) },

  { RAD(-22),RAD(22) }, { RAD(-22),RAD(22) },

  { RAD(-47),RAD(-3) },

  { 0,1 }, { 0,1 }, { 0,1 }, { 0,1 }, { 0,1 }, { 0,1 }, { 0,1 }, { 0,1 },
  { 0,1 }, { 0,1 }
}
```

This table holds the software limits of each of the outputs.

Definition at line 381 of file ERS210Info.h.

6.2.4.70 `const unsigned ERS210Info::PIDJointOffset = 0`

The beginning of the PID Joints.

Definition at line 65 of file ERS210Info.h.

6.2.4.71 `const char* const ERS210Info::PrimitiveName[NumOutputs]`**Initial value:**

```
{
  "PRM:/r2/c1-Joint2:j1",
  "PRM:/r2/c1/c2-Joint2:j2",
  "PRM:/r2/c1/c2/c3-Joint2:j3",
  "PRM:/r4/c1-Joint2:j1",
  "PRM:/r4/c1/c2-Joint2:j2",
  "PRM:/r4/c1/c2/c3-Joint2:j3",

  "PRM:/r3/c1-Joint2:j1",
  "PRM:/r3/c1/c2-Joint2:j2",
  "PRM:/r3/c1/c2/c3-Joint2:j3",
  "PRM:/r5/c1-Joint2:j1",
  "PRM:/r5/c1/c2-Joint2:j2",
  "PRM:/r5/c1/c2/c3-Joint2:j3",

  "PRM:/r1/c1-Joint2:j1",
}
```

```

"PRM:/r1/c1/c2-Joint2:j2",
"PRM:/r1/c1/c2/c3-Joint2:j3",

"PRM:/r6/c2-Joint2:j2",
"PRM:/r6/c1-Joint2:j1",

"PRM:/r1/c1/c2/c3/c4-Joint2:j4",

"PRM:/r1/c1/c2/c3/l1-LED2:l1",
"PRM:/r1/c1/c2/c3/l4-LED2:l4",
"PRM:/r1/c1/c2/c3/l2-LED2:l2",
"PRM:/r1/c1/c2/c3/l5-LED2:l5",
"PRM:/r1/c1/c2/c3/l3-LED2:l3",
"PRM:/r1/c1/c2/c3/l6-LED2:l6",
"PRM:/r1/c1/c2/c3/l7-LED2:l7",

"PRM:/r6/l2-LED2:l2",
"PRM:/r6/l1-LED2:l1",

"PRM:/r1/c1/c2/c3/e1-Joint3:j5",
"PRM:/r1/c1/c2/c3/e2-Joint3:j6"
}

```

the joint identifier strings used to refer to specific joints in OPEN-R (but not needed for others)

Warning:

IMPORTANT!!!! DO NOT CHANGE THE ORDER OF ITEMS IN THIS TABLE!!!

The offset consts defined in this file correspond to this table and will make life easier if you feel the need to reorder things, but they aren't used perfect *everywhere*

In particular, assumptions are made that the pid joints will be in slots 0-numPIDJoints and that the fast outputs (ie NOT ears) will be in slots 0-NumFastOutputs

There may be other assumptions not noted here!!!

Note:

These entries DON'T correspond to the CPC index numbers defined in [World-State](#) (this only lists joints, and in a different order defined by OPEN-R, that one has sensors as well

Definition at line 250 of file ERS210Info.h.

6.2.4.72 const unsigned int [ERS210Info::SlowFrameTime](#) = 128

time between frames for the ears (which move slower for some reason, don't want to mix with other outputs) (milliseconds)

Definition at line 29 of file ERS210Info.h.

6.2.4.73 `const unsigned int ERS210Info::SoundBufferTime = 32`

the number of milliseconds per sound buffer... I'm not sure if this can be changed

Definition at line 31 of file ERS210Info.h.

6.2.4.74 `const LEDBitMask_t ERS210Info::TailLEDMask =
TIRedLEDMask|TIBluLEDMask`

LEDs on tail.

Definition at line 147 of file ERS210Info.h.

6.2.4.75 `const unsigned ERS210Info::TailOffset = HeadOffset+NumHeadJoints`

the offset of the beginning of the tail joints

Definition at line 68 of file ERS210Info.h.

6.2.4.76 `const LEDBitMask_t ERS210Info::TIBluLEDMask =
1<<(TIBluLEDOffset-LEDOffset)`

blue tail light

Definition at line 142 of file ERS210Info.h.

6.2.4.77 `const LEDBitMask_t ERS210Info::TIRedLEDMask =
1<<(TIRedLEDOffset-LEDOffset)`

red tail light

Definition at line 141 of file ERS210Info.h.

6.2.4.78 `const LEDBitMask_t ERS210Info::TopBrLEDMask =
1<<(TopBrLEDOffset-LEDOffset)`

top bar (green)

Definition at line 140 of file ERS210Info.h.

6.2.4.79 `const LEDBitMask_t ERS210Info::TopLLEDMask =
1<<(TopLLEDOffset-LEDOffset)`

top left (red - angry)

Definition at line 138 of file ERS210Info.h.

6.2.4.80 `const LEDBitMask_t ERS210Info::TopRLEDMask =
 1<<(TopRLEDOffset-LEDOffset)`

top right (red - angry)

Definition at line 139 of file ERS210Info.h.

6.3 ERS220Info Namespace Reference

6.3.1 Detailed Description

Contains information about the ERS-220 Robot, such as number of joints, PID defaults, timing information, etc.

LED Bitmasks

Bitmasks for use when specifying combinations of LEDs (see LEDEngine) Note that L/R are robot's POV

- typedef unsigned int [LEDBitMask_t](#)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceFrontLeftLEDMask](#) = 1<<(FaceFrontLeftLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceFrontRightLEDMask](#) = 1<<(FaceFrontRightLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceCenterLeftLEDMask](#) = 1<<(FaceCenterLeftLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceCenterRightLEDMask](#) = 1<<(FaceCenterRightLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceBackLeftLEDMask](#) = 1<<(FaceBackLeftLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceBackRightLEDMask](#) = 1<<(FaceBackRightLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [ModeLEDMask](#) = 1<<(ModeLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t BackLeft1LEDMask = 1<<(BackLeft1LEDOffset-LEDOffset)`
So you can be clear when you're referring to a LED bitmask.
- `const LEDBitMask_t BackLeft2LEDMask = 1<<(BackLeft2LEDOffset-LEDOffset)`
So you can be clear when you're referring to a LED bitmask.
- `const LEDBitMask_t BackLeft3LEDMask = 1<<(BackLeft3LEDOffset-LEDOffset)`
So you can be clear when you're referring to a LED bitmask.
- `const LEDBitMask_t BackRight3LEDMask = 1<<(BackRight3LEDOffset-LEDOffset)`
So you can be clear when you're referring to a LED bitmask.
- `const LEDBitMask_t BackRight2LEDMask = 1<<(BackRight2LEDOffset-LEDOffset)`
So you can be clear when you're referring to a LED bitmask.
- `const LEDBitMask_t BackRight1LEDMask = 1<<(BackRight1LEDOffset-LEDOffset)`
So you can be clear when you're referring to a LED bitmask.
- `const LEDBitMask_t TailLeftLEDMask = 1<<(TailLeftLEDOffset-LEDOffset)`
So you can be clear when you're referring to a LED bitmask.
- `const LEDBitMask_t TailCenterLEDMask = 1<<(TailCenterLEDOffset-LEDOffset)`
So you can be clear when you're referring to a LED bitmask.
- `const LEDBitMask_t TailRightLEDMask = 1<<(TailRightLEDOffset-LEDOffset)`
So you can be clear when you're referring to a LED bitmask.
- `const LEDBitMask_t FaceFrontBLEDMask = 1<<(FaceFrontBLEDOffset-LEDOffset)`
So you can be clear when you're referring to a LED bitmask.
- `const LEDBitMask_t FaceFrontALEDMask = 1<<(FaceFrontALEDOffset-LEDOffset)`
So you can be clear when you're referring to a LED bitmask.

- const `LEDBitMask_t FaceFrontCLEDMask` = `1<<(FaceFrontCLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- const `LEDBitMask_t RetractableHeadLEDMask` = `1<<(RetractableHeadLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- const `LEDBitMask_t BotLLEDMask` = `1<<(BotLLEDOffset-LEDOffset)`

bottom left (red - sad)

- const `LEDBitMask_t BotRLEDMask` = `1<<(BotRLEDOffset-LEDOffset)`

bottom right (red - sad)

- const `LEDBitMask_t MidLLEDMask` = `1<<(MidLLEDOffset-LEDOffset)`

middle left (green - happy)

- const `LEDBitMask_t MidRLEDMask` = `1<<(MidRLEDOffset-LEDOffset)`

middle right (green - happy)

- const `LEDBitMask_t TopLLEDMask` = `1<<(TopLLEDOffset-LEDOffset)`

top left (red - angry)

- const `LEDBitMask_t TopRLEDMask` = `1<<(TopRLEDOffset-LEDOffset)`

top right (red - angry)

- const `LEDBitMask_t TopBrLEDMask` = `1<<(TopBrLEDOffset-LEDOffset)`

top bar (green)

- const `LEDBitMask_t TIRedLEDMask` = `1<<(TIRedLEDOffset-LEDOffset)`

red tail light

- const `LEDBitMask_t TIBluLEDMask` = `1<<(TIBluLEDOffset-LEDOffset)`

blue tail light

- const `LEDBitMask_t FaceLEDMask`

LEDs for face.

- const `LEDBitMask_t HeadLEDMask`

LEDs on head (face plus retractable light).

- const `LEDBitMask_t BackLEDMask`

LEDs on back.

- const [LEDBitMask_t](#) TailLEDMask
LEDs for tail.
- const [LEDBitMask_t](#) AllLEDMask = ~0
selects all of the leds

Input Offsets

The order in which inputs should be stored

- enum [ButtonOffset_t](#) {
[LFrPawOffset](#) = LFrLegOrder, [RFrPawOffset](#) = RFrLegOrder, [LBkPawOffset](#) =
LBkLegOrder, [RBkPawOffset](#) = RBkLegOrder,
[ChinButOffset](#) = 4, [BackButOffset](#), [HeadFrButOffset](#), [HeadBkButOffset](#),
[TailLeftButOffset](#), [TailCenterButOffset](#), [TailRightButOffset](#) }
holds offsets to different buttons in [WorldState::buttons\[\]](#)
- enum [SensorOffset_t](#) {
[IRDistOffset](#) = 0, [BAccelOffset](#), [LAccelOffset](#), [DAccelOffset](#),
[ThermoOffset](#), [PowerRemainOffset](#), [PowerThermoOffset](#), [PowerCapacity-](#)
[Offset](#),
[PowerVoltageOffset](#), [PowerCurrentOffset](#) }
holds offset to different sensor values in [WorldState::sensors\[\]](#)

Output Types Information

- const unsigned [NumPIDJoints](#) = 15
The number of joints which use PID motion - everything.
- const unsigned [NumLEDs](#) = 20
The number LEDs which can be controlled.
- const unsigned [NumBinJoints](#) = 0
The number of binary joints (210 has ears).
- const unsigned [NumOutputs](#) = [NumPIDJoints](#) + [NumBinJoints](#) + [NumLEDs](#)

the total number of outputs

- const bool `IsFastOutput` [`NumOutputs`]
true for joints which can be updated every 32 ms (all but the ears on a 210)
- const bool `IsRealERS220` [`NumOutputs`]
true for joints which can be updated every 32 ms (all but the ears on a 210)
- const unsigned `JointsPerLeg` = 3
The number of joints per leg.
- const unsigned `NumLegs` = 4
The number of legs.
- const unsigned `NumLegJoints` = `JointsPerLeg*NumLegs`
the TOTAL number of joints on ALL legs
- const unsigned `NumHeadJoints` = 3
The number of joints in the neck.
- const unsigned `NumTailJoints` = 0
The number of joints assigned to the tail.
- const unsigned `NumMouthJoints` = 0
the number of joints that control the mouth
- const unsigned `NumEarJoints` = 0
The number of joints which control the ears (NOT per ear, is total).
- const unsigned `NumButtons` = 11
the number of buttons that are available, see `ButtonOffset_t`
- const unsigned `NumSensors` = 1+3+1+5
1 dist, 3 accel, 1 thermo, 5 from power, see `SensorOffset_t`

Output Offsets

Corresponds to entries in `PrimitiveName`, defined at the end of this file

- const unsigned `PIDJointOffset` = 0
The beginning of the PID Joints.

- const unsigned `LegOffset` = `PIDJointOffset`
the offset of the beginning of the leg joints
- const unsigned `HeadOffset` = `LegOffset`+`NumLegJoints`
the offset of the beginning of the head joints
- const unsigned `LEDOffset` = `PIDJointOffset` + `NumPIDJoints`
the offset of LEDs in `WorldState::outputs` and `MotionCommand` functions
- const unsigned `BinJointOffset` = `NumOutputs`
The beginning of the binary joints.

CPC IDs

values defined by OPEN-R, used to interface with lower level OPEN-R code to read sensors - DOESN'T correspond to `::PrimitiveName`

- const int `CPCJointNeckTilt` = 0
- const int `CPCJointNeckPan` = 1
- const int `CPCJointNeckRoll` = 2
- const int `CPCSensorPSD` = 3
- const int `CPCSensorHeadBackPressure` = 4
- const int `CPCSensorHeadFrontPressure` = 5
- const int `CPCSensorChinSwitch` = 6
- const int `CPCJointLFRotator` = 7
- const int `CPCJointLFElevator` = 8
- const int `CPCJointLFKnee` = 9
- const int `CPCSensorLFPaw` = 10
- const int `CPCJointLHRotator` = 11
- const int `CPCJointLHElevator` = 12
- const int `CPCJointLHKnee` = 13
- const int `CPCSensorLHPaw` = 14
- const int `CPCJointRFRotator` = 15
- const int `CPCJointRFElevator` = 16
- const int `CPCJointRFKnee` = 17
- const int `CPCSensorRFPaw` = 18
- const int `CPCJointRHRotator` = 19
- const int `CPCJointRHElevator` = 20
- const int `CPCJointRHKnee` = 21
- const int `CPCSensorRHPaw` = 22

- const int [CPCSensorThermoSensor](#) = 23
- const int [CPCSensorBackSwitch](#) = 24
- const int [CPCSensorTailLeftSwitch](#) = 25
- const int [CPCSensorTailCenterSwitch](#) = 26
- const int [CPCSensorTailRightSwitch](#) = 27
- const int [CPCSensorAccelFB](#) = 28
- const int [CPCSensorAccelLR](#) = 29
- const int [CPCSensorAccelUD](#) = 30

Enumerations

- enum [LegOrder_t](#) { [LFrLegOrder](#) = 0, [RFrLegOrder](#), [LBkLegOrder](#), [RBkLegOrder](#) }
the ordering of legs
- enum [LegOffset_t](#) { [LFrLegOffset](#) = $\text{LegOffset} + \text{LFrLegOrder} * \text{JointsPerLeg}$, [RFrLegOffset](#) = $\text{LegOffset} + \text{RFrLegOrder} * \text{JointsPerLeg}$, [LBkLegOffset](#) = $\text{LegOffset} + \text{LBkLegOrder} * \text{JointsPerLeg}$, [RBkLegOffset](#) = $\text{LegOffset} + \text{RBkLegOrder} * \text{JointsPerLeg}$ }
The offsets of the individual legs.
- enum [REKOffset_t](#) { [RotatorOffset](#) = 0, [ElevatorOffset](#), [KneeOffset](#) }
The offsets within appendages (the legs) Note that the ordering matches the actual physical ordering of joints on the appendage (and not that of the head's [TPROffset_t](#)'s).
- enum [TPROffset_t](#) { [TiltOffset](#) = 0, [PanOffset](#), [RollOffset](#) }
The offsets of appendages with tilt (elevation), pan (heading), and roll joints (i.e. head) Note that the ordering matches the actual physical ordering of joints on the appendage (and not that of the leg's [REKOffset_t](#)'s).
- enum [LEDOffset_t](#) {
[FaceFrontLeftLEDOffset](#) = [LEDOffset](#), [FaceFrontRightLEDOffset](#), [FaceCenterLeftLEDOffset](#), [FaceCenterRightLEDOffset](#),
[FaceBackLeftLEDOffset](#), [FaceBackRightLEDOffset](#), [ModeLEDOffset](#), [BackLeft1LEDOffset](#),
[BackLeft2LEDOffset](#), [BackLeft3LEDOffset](#), [BackRight3LEDOffset](#), [BackRight2LEDOffset](#),
[BackRight1LEDOffset](#), [TailLeftLEDOffset](#), [TailCenterLEDOffset](#), [TailRightLEDOffset](#),
[FaceFrontBLEDOffset](#), [FaceFrontALEDOffset](#), [FaceFrontCLEDOffset](#), [RetractableHeadLEDOffset](#),

`BotLLEDOffset` = FaceFrontLeftLEDOffset, `BotRLEDOffset` = FaceFrontRightLEDOffset, `MidLLEDOffset` = FaceCenterLeftLEDOffset, `MidRLEDOffset` = FaceCenterRightLEDOffset,

`TopLLEDOffset` = FaceBackLeftLEDOffset, `TopRLEDOffset` = FaceBackRightLEDOffset, `TopBrLEDOffset` = ModeLEDOffset, `TIBluLEDOffset` = TailLeftLEDOffset,

`TIRedLEDOffset` = TailRightLEDOffset }

The offsets of the individual LEDs on the head and tail. Note that L/R are robot's POV. See also LEDBitMask.t.

- enum `MinMaxRange_t` { `MinRange`, `MaxRange` }

Defines the min and max index of entries in #outputRanges and #mechanicalLimits.

Variables

- const unsigned int `FrameTime` = 8
time between frames in the motion system (milliseconds)
- const unsigned int `NumFrames` = 4
the number of frames per buffer (don't forget also double buffered)
- const unsigned int `SlowFrameTime` = 128
time between frames for the ears (which move slower for some reason, don't want to mix with other outputs) (milliseconds)
- const unsigned int `NumSlowFrames` = 1
the number of frames per buffer being sent to ears (double buffered as well)
- const unsigned int `SoundBufferTime` = 32
the number of milliseconds per sound buffer... I'm not sure if this can be changed
- const unsigned `outputNameLen` = 9
The length of the strings used for each of the outputs in outputNames (doesn't include null term).
- const char *const `outputNames` [`NumOutputs`]
A name of uniform length for referring to joints - handy for posture files, etc.
- const char *const `PrimitiveName` [`NumOutputs`]
the joint identifier strings used to refer to specific joints in OPEN-R (but not needed for others)

- const float [DefaultPIDs](#) [[NumPIDJoints](#)][3]

This table holds the default PID values for each joint. [PIDMC](#).

- const float [MaxOutputSpeed](#) [[NumOutputs](#)]

These values are Sony's recommended maximum joint velocities, in rad/ms.

- const double [outputRanges](#) [[NumOutputs](#)][2]

This table holds the software limits of each of the outputs.

- const double [mechanicalLimits](#) [[NumOutputs](#)][2]

This table holds the mechanical limits of each of the outputs.

6.3.2 Typedef Documentation

6.3.2.1 typedef unsigned int [ERS220Info::LEDBitMask_t](#)

So you can be clear when you're referring to a LED bitmask.

Definition at line 177 of file [ERS220Info.h](#).

6.3.3 Enumeration Type Documentation

6.3.3.1 enum [ERS220Info::ButtonOffset_t](#)

holds offsets to different buttons in [WorldState::buttons](#)[]

Should be a straight mapping to the [ButtonSourceIDs](#)

Note that the chest (power) button is not a normal button. It kills power to the motors at a hardware level, and isn't sensed in the normal way. If you want to know when it is pressed (and you are about to shut down) see [PowerSourceID::PauseSID](#).

See also:

[WorldState::buttons](#)

[ButtonSourceID_t](#)

Enumeration values:

LFrPawOffset

RFrPawOffset

LBkPawOffset

RBkPawOffset

ChinButOffset

BackButOffset

HeadFrButOffset for the "antenna" - this is <.2 if pushed back all the way

HeadBkButOffset for the "antenna" - this is >.98 if pulled forward, <.2 if pushed back partly

TailLeftButOffset

TailCenterButOffset

TailRightButOffset

Definition at line 260 of file ERS220Info.h.

6.3.3.2 enum [ERS220Info::LEDOffset_t](#)

The offsets of the individual LEDs on the head and tail. Note that L/R are robot's POV. See also LEDBitMask_t.

Enumeration values:

FaceFrontLeftLEDOffset head face side light (front left - blue)

FaceFrontRightLEDOffset head face side light (front right - blue)

FaceCenterLeftLEDOffset head face side light (center left - blue)

FaceCenterRightLEDOffset head face side light (center right - blue)

FaceBackLeftLEDOffset head face side light (back left - red)

FaceBackRightLEDOffset head face side light (back right - red)

ModeLEDOffset mode indicator (back of the head - orange)

BackLeft1LEDOffset back multi-indicator (left #1 - blue)

BackLeft2LEDOffset back multi-indicator (left #2 - blue)

BackLeft3LEDOffset back multi-indicator (left #3 - blue)

BackRight3LEDOffset back multi-indicator (right #3 - blue)

BackRight2LEDOffset back multi-indicator (right #2 - blue)

BackRight1LEDOffset back multi-indicator (right #1 - blue)

TailLeftLEDOffset tail light (left - blue)

TailCenterLEDOffset tail light (center - red)

TailRightLEDOffset tail light (right - blue)

FaceFrontBLEDOffset face front light B (blue)

FaceFrontALEDOffset face front light A (blue)

FaceFrontCLEDOffset face front light C (red)

RetractableHeadLEDOffset retractable head light

BotLLEDOffset bottom left (red - sad) (ERS-210)
BotRLEDOffset bottom right (red - sad) (ERS-210)
MidLLEDOffset middle left (green - happy) (ERS-210)
MidRLEDOffset middle right (green - happy) (ERS-210)
TopLLEDOffset top left (red - angry) (ERS-210)
TopRLEDOffset top right (red - angry) (ERS-210)
TopBrLEDOffset top bar (green) (ERS-210)
TIBluLEDOffset blue tail light (ERS-210)
TIRedLEDOffset red tail light (ERS-210)

Definition at line 141 of file ERS220Info.h.

6.3.3.3 enum [ERS220Info::LegOffset_t](#)

The offsets of the individual legs.

Enumeration values:

LFrLegOffset beginning of left front leg
RFrLegOffset beginning of right front leg
LBkLegOffset beginning of left back leg
RBkLegOffset beginning of right back leg

Definition at line 119 of file ERS220Info.h.

6.3.3.4 enum [ERS220Info::LegOrder_t](#)

the ordering of legs

Enumeration values:

LFrLegOrder left front leg
RFrLegOrder right front leg
LBkLegOrder left back leg
RBkLegOrder right back leg

Definition at line 111 of file ERS220Info.h.

6.3.3.5 enum [ERS220Info::MinMaxRange_t](#)

Defines the min and max index of entries in [outputRanges](#) and [mechanicalLimits](#).

Enumeration values:

MinRange

MaxRange

Definition at line 480 of file ERS220Info.h.

6.3.3.6 enum [ERS220Info::REKOffset_t](#)

The offsets within appendages (the legs) Note that the ordering matches the actual physical ordering of joints on the appendage (and not that of the head's TPROffset_t's).

Enumeration values:

RotatorOffset moves leg forward or backward along body

ElevatorOffset moves leg toward or away from body

KneeOffset moves knee

Definition at line 127 of file ERS220Info.h.

6.3.3.7 enum [ERS220Info::SensorOffset_t](#)

holds offset to different sensor values in [WorldState::sensors\[\]](#)

See also:

[WorldState::sensors\[\]](#)

Enumeration values:

IRDistOffset in millimeters

BAccelOffset backward acceleration, in m/s^2 , negative if sitting on butt (positive for faceplant)

LAccelOffset acceleration to the robot's left, in m/s^2 , negative if lying on robot's left side

DAccelOffset downward acceleration, in m/s^2 , negative if standing up... be careful about the signs on all of these...

ThermoOffset in degrees Celcius

PowerRemainOffset percentage, 0-1

PowerThermoOffset degrees Celcius

PowerCapacityOffset milli-amp hours

PowerVoltageOffset volts

PowerCurrentOffset milli-amp negative values (maybe positive while charging?)

Definition at line 276 of file ERS220Info.h.

6.3.3.8 enum [ERS220Info::TPROffset_t](#)

The offsets of appendages with tilt (elevation), pan (heading), and roll joints (i.e. head) Note that the ordering matches the actual physical ordering of joints on the appendage (and not that of the leg's REKOffset.t's).

Enumeration values:

TiltOffset tilt/elevation (vertical)

PanOffset pan/heading (horizontal)

RollOffset roll (rotational)

Definition at line 134 of file ERS220Info.h.

6.3.4 Variable Documentation

6.3.4.1 const [LEDBitMask_t ERS220Info::AllLEDMask](#) = ~0

selects all of the leds

Definition at line 239 of file ERS220Info.h.

6.3.4.2 const [LEDBitMask_t ERS220Info::BackLEDMask](#)

Initial value:

```
BackLeft1LEDMask
| BackLeft2LEDMask
| BackLeft3LEDMask
| BackRight1LEDMask
| BackRight2LEDMask
| BackRight3LEDMask
```

LEDs on back.

Definition at line 227 of file ERS220Info.h.

6.3.4.3 `const LEDBitMask_t ERS220Info::BackLeft1LEDMask = 1<<(BackLeft1LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 185 of file ERS220Info.h.

6.3.4.4 `const LEDBitMask_t ERS220Info::BackLeft2LEDMask = 1<<(BackLeft2LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 186 of file ERS220Info.h.

6.3.4.5 `const LEDBitMask_t ERS220Info::BackLeft3LEDMask = 1<<(BackLeft3LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 187 of file ERS220Info.h.

6.3.4.6 `const LEDBitMask_t ERS220Info::BackRight1LEDMask = 1<<(BackRight1LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 190 of file ERS220Info.h.

6.3.4.7 `const LEDBitMask_t ERS220Info::BackRight2LEDMask = 1<<(BackRight2LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 189 of file ERS220Info.h.

6.3.4.8 `const LEDBitMask_t ERS220Info::BackRight3LEDMask = 1<<(BackRight3LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 188 of file ERS220Info.h.

6.3.4.9 `const unsigned ERS220Info::BinJointOffset = NumOutputs`

The beginning of the binary joints.

Definition at line 107 of file ERS220Info.h.

6.3.4.10 `const LEDBitMask_t ERS220Info::BotLLEDMask = 1<<(BotLLEDOffset-LEDOffset)`

bottom left (red - sad)

Definition at line 200 of file ERS220Info.h.

6.3.4.11 `const LEDBitMask_t ERS220Info::BotRLEDMask = 1<<(BotRLEDOffset-LEDOffset)`

bottom right (red - sad)

Definition at line 201 of file ERS220Info.h.

6.3.4.12 `const int ERS220Info::CPCJointLFElevator = 8 [static]`

Definition at line 539 of file ERS220Info.h.

6.3.4.13 `const int ERS220Info::CPCJointLFKnee = 9 [static]`

Definition at line 540 of file ERS220Info.h.

6.3.4.14 `const int ERS220Info::CPCJointLFRotator = 7 [static]`

Definition at line 538 of file ERS220Info.h.

6.3.4.15 `const int ERS220Info::CPCJointLHElevator = 12 [static]`

Definition at line 543 of file ERS220Info.h.

6.3.4.16 `const int ERS220Info::CPCJointLHKnee = 13 [static]`

Definition at line 544 of file ERS220Info.h.

6.3.4.17 `const int ERS220Info::CPCJointLHRotator = 11 [static]`

Definition at line 542 of file ERS220Info.h.

6.3.4.18 `const int ERS220Info::CPCJointNeckPan = 1` [static]

Definition at line 532 of file ERS220Info.h.

6.3.4.19 `const int ERS220Info::CPCJointNeckRoll = 2` [static]

Definition at line 533 of file ERS220Info.h.

6.3.4.20 `const int ERS220Info::CPCJointNeckTilt = 0` [static]

Definition at line 531 of file ERS220Info.h.

6.3.4.21 `const int ERS220Info::CPCJointRFElevator = 16` [static]

Definition at line 547 of file ERS220Info.h.

6.3.4.22 `const int ERS220Info::CPCJointRFKnee = 17` [static]

Definition at line 548 of file ERS220Info.h.

6.3.4.23 `const int ERS220Info::CPCJointRFRotator = 15` [static]

Definition at line 546 of file ERS220Info.h.

6.3.4.24 `const int ERS220Info::CPCJointRHElevator = 20` [static]

Definition at line 551 of file ERS220Info.h.

6.3.4.25 `const int ERS220Info::CPCJointRHKnee = 21` [static]

Definition at line 552 of file ERS220Info.h.

6.3.4.26 `const int ERS220Info::CPCJointRHRotator = 19` [static]

Definition at line 550 of file ERS220Info.h.

6.3.4.27 `const int ERS220Info::CPCSensorAccelFB = 28` [static]

Definition at line 559 of file ERS220Info.h.

6.3.4.28 `const int ERS220Info::CPCSensorAccelLR = 29` [static]

Definition at line 560 of file ERS220Info.h.

6.3.4.29 `const int ERS220Info::CPCSensorAccelUD = 30` [static]

Definition at line 561 of file ERS220Info.h.

6.3.4.30 `const int ERS220Info::CPCSensorBackSwitch = 24` [static]

Definition at line 555 of file ERS220Info.h.

6.3.4.31 `const int ERS220Info::CPCSensorChinSwitch = 6` [static]

Definition at line 537 of file ERS220Info.h.

6.3.4.32 `const int ERS220Info::CPCSensorHeadBackPressure = 4` [static]

Definition at line 535 of file ERS220Info.h.

6.3.4.33 `const int ERS220Info::CPCSensorHeadFrontPressure = 5` [static]

Definition at line 536 of file ERS220Info.h.

6.3.4.34 `const int ERS220Info::CPCSensorLFPaw = 10` [static]

Definition at line 541 of file ERS220Info.h.

6.3.4.35 `const int ERS220Info::CPCSensorLHPaw = 14` [static]

Definition at line 545 of file ERS220Info.h.

6.3.4.36 `const int ERS220Info::CPCSensorPSD = 3` [static]

Definition at line 534 of file ERS220Info.h.

6.3.4.37 `const int ERS220Info::CPCSensorRFPaw = 18` [static]

Definition at line 549 of file ERS220Info.h.

6.3.4.38 `const int ERS220Info::CPCSensorRHPaw = 22` [static]

Definition at line 553 of file ERS220Info.h.

6.3.4.39 `const int ERS220Info::CPCSensorTailCenterSwitch = 26` [static]

Definition at line 557 of file ERS220Info.h.

6.3.4.40 `const int ERS220Info::CPCSensorTailLeftSwitch = 25` [static]

Definition at line 556 of file ERS220Info.h.

6.3.4.41 `const int ERS220Info::CPCSensorTailRightSwitch = 27` [static]

Definition at line 558 of file ERS220Info.h.

6.3.4.42 `const int ERS220Info::CPCSensorThermoSensor = 23` [static]

Definition at line 554 of file ERS220Info.h.

6.3.4.43 `const float ERS220Info::DefaultPIDs[NumPIDJoints][3]`

Initial value:

```
{
    { 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
    { 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
    { 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },
    { 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
    { 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
    { 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },
    { 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
    { 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
    { 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },
    { 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
    { 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
    { 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },
    { 0x0A/(double)(1<<(16-0xE)), 0x08/(double)(1<<(16-0x2)), 0x0C/(double)(1<<(16-0xF)) },
    { 0x0D/(double)(1<<(16-0xE)), 0x08/(double)(1<<(16-0x2)), 0x0B/(double)(1<<(16-0xF)) },
    { 0x0A/(double)(1<<(16-0xE)), 0x08/(double)(1<<(16-0x2)), 0x0C/(double)(1<<(16-0xF)) }
}
```

This table holds the default PID values for each joint. [PIDMC](#).

Definition at line 417 of file ERS220Info.h.

6.3.4.44 `const LEDBitMask_t ERS220Info::FaceBackLeftLEDMask = 1<<(FaceBackLeftLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 182 of file ERS220Info.h.

6.3.4.45 `const LEDBitMask_t ERS220Info::FaceBackRightLEDMask = 1<<(FaceBackRightLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 183 of file ERS220Info.h.

6.3.4.46 `const LEDBitMask_t ERS220Info::FaceCenterLeftLEDMask = 1<<(FaceCenterLeftLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 180 of file ERS220Info.h.

6.3.4.47 `const LEDBitMask_t ERS220Info::FaceCenterRightLEDMask = 1<<(FaceCenterRightLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 181 of file ERS220Info.h.

6.3.4.48 `const LEDBitMask_t ERS220Info::FaceFrontALEDMask = 1<<(FaceFrontALEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 195 of file ERS220Info.h.

6.3.4.49 `const LEDBitMask_t ERS220Info::FaceFrontBLEDMask = 1<<(FaceFrontBLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 194 of file ERS220Info.h.

6.3.4.50 `const LEDBitMask_t ERS220Info::FaceFrontCLEDMask = 1<<(FaceFrontCLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 196 of file ERS220Info.h.

6.3.4.51 `const LEDBitMask_t ERS220Info::FaceFrontLeftLEDMask = 1<<(FaceFrontLeftLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 178 of file ERS220Info.h.

6.3.4.52 `const LEDBitMask_t ERS220Info::FaceFrontRightLEDMask = 1<<(FaceFrontRightLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 179 of file ERS220Info.h.

6.3.4.53 `const LEDBitMask_t ERS220Info::FaceLEDMask`

Initial value:

```
FaceFrontLeftLEDMask
| FaceFrontRightLEDMask
| FaceCenterLeftLEDMask
| FaceCenterRightLEDMask
| FaceBackLeftLEDMask
| FaceBackRightLEDMask
| FaceFrontALEDMask
| FaceFrontBLEDMask
| FaceFrontCLEDMask
| ModeLEDMask
```

LEDs for face.

Definition at line 211 of file ERS220Info.h.

6.3.4.54 `const unsigned int ERS220Info::FrameTime = 8`

time between frames in the motion system (milliseconds)

Definition at line 30 of file ERS220Info.h.

6.3.4.55 `const LEDBitMask_t ERS220Info::HeadLEDMask`**Initial value:**

```
FaceLEDMask
| RetractableHeadLEDMask
```

LEDs on head (face plus retractable light).

Definition at line 223 of file ERS220Info.h.

6.3.4.56 `const unsigned ERS220Info::HeadOffset = LegOffset+NumLegJoints`

the offset of the beginning of the head joints

Definition at line 103 of file ERS220Info.h.

6.3.4.57 `const bool ERS220Info::IsFastOutput[NumOutputs]`**Initial value:**

```
{
    true, true, true,
    true, true, true,
    true, true, true,
    true, true, true,
    true, true, true,

    true, true, true,
    true, true, true,
    true,
    true, true, true,
    true, true, true,
    true, true, true,
    true, true, true,
    true,

}
```

true for joints which can be updated every 32 ms (all but the ears on a 210)

Definition at line 44 of file ERS220Info.h.

6.3.4.58 `const bool ERS220Info::IsRealERS220[NumOutputs]`**Initial value:**

```
{  
  
    true, true, true,  
    true, true, true,  
    true, true, true,  
    true, true, true,  
    true, true, true,  
  
    true, true, true,  
    true, true, true,  
    true,  
    true, true, true,  
    true, true, true,  
    true, true, true,  
    true, true, true,  
    true,  
  
}
```

true for joints which can be updated every 32 ms (all but the ears on a 210)

Definition at line 64 of file ERS220Info.h.

6.3.4.59 const unsigned [ERS220Info::JointsPerLeg](#) = 3

The number of joints per leg.

Definition at line 83 of file ERS220Info.h.

6.3.4.60 const unsigned [ERS220Info::LEDOffset](#) = [PIDJointOffset](#) + [NumPIDJoints](#)

the offset of LEDs in [WorldState::outputs](#) and [MotionCommand](#) functions

Definition at line 105 of file ERS220Info.h.

6.3.4.61 const unsigned [ERS220Info::LegOffset](#) = [PIDJointOffset](#)

the offset of the beginning of the leg joints

Definition at line 102 of file ERS220Info.h.

6.3.4.62 const float [ERS220Info::MaxOutputSpeed](#)[[NumOutputs](#)]

Initial value:

```
{  
    2.8143434e-03,  
    2.4980025e-03,
```

```

2.8361600e-03,
2.8143434e-03,
2.4980025e-03,
2.8361600e-03,
2.8143434e-03,
2.4980025e-03,
2.8361600e-03,
2.8143434e-03,
2.4980025e-03,
2.8361600e-03,

2.1053034e-03,
3.0106930e-03,
3.0106930e-03,

0,0,0,
0,0,0,
0,
0,0,0,
0,0,0,
0,0,0,
0,0,0,
0
}

```

These values are Sony's recommended maximum joint velocities, in rad/ms.

a value ≤ 0 means infinite speed (e.g. LEDs)

Definition at line 444 of file ERS220Info.h.

6.3.4.63 `const double ERS220Info::mechanicalLimits[NumOutputs][2]`

Initial value:

```

{
  { RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },
  { RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },
  { RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },
  { RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },

  { RAD(-85),RAD(46) }, { RAD(-92.6),RAD(92.6) }, { RAD(-32),RAD(32) },

  {0,1},{0,1},{0,1},
  {0,1},{0,1},{0,1},
  {0,1},
  {0,1},{0,1},{0,1},
  {0,1},{0,1},{0,1},
  {0,1},{0,1},{0,1},
  {0,1},{0,1},{0,1},
  {0,1}
}

```

This table holds the mechanical limits of each of the outputs.

Definition at line 503 of file ERS220Info.h.

6.3.4.64 `const LEDBitMask_t ERS220Info::MidLLEDMask = 1<<(MidLLEDOffset-LEDOffset)`

middle left (green - happy)

Definition at line 202 of file ERS220Info.h.

6.3.4.65 `const LEDBitMask_t ERS220Info::MidRLEDMask = 1<<(MidRLEDOffset-LEDOffset)`

middle right (green - happy)

Definition at line 203 of file ERS220Info.h.

6.3.4.66 `const LEDBitMask_t ERS220Info::ModeLEDMask = 1<<(ModeLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 184 of file ERS220Info.h.

6.3.4.67 `const unsigned ERS220Info::NumBinJoints = 0`

The number of binary joints (210 has ears).

Definition at line 41 of file ERS220Info.h.

6.3.4.68 `const unsigned ERS220Info::NumButtons = 11`

the number of buttons that are available, see ButtonOffset.t

Definition at line 90 of file ERS220Info.h.

6.3.4.69 `const unsigned ERS220Info::NumEarJoints = 0`

The number of joints which control the ears (NOT per ear, is total).

Definition at line 89 of file ERS220Info.h.

6.3.4.70 `const unsigned int ERS220Info::NumFrames = 4`

the number of frames per buffer (don't forget also double buffered)

Definition at line 31 of file ERS220Info.h.

6.3.4.71 `const unsigned ERS220Info::NumHeadJoints = 3`

The number of joints in the neck.

Definition at line 86 of file ERS220Info.h.

6.3.4.72 `const unsigned ERS220Info::NumLEDs = 20`

The number LEDs which can be controlled.

Definition at line 40 of file ERS220Info.h.

6.3.4.73 `const unsigned ERS220Info::NumLegJoints = JointsPerLeg*NumLegs`

the TOTAL number of joints on ALL legs

Definition at line 85 of file ERS220Info.h.

6.3.4.74 `const unsigned ERS220Info::NumLegs = 4`

The number of legs.

Definition at line 84 of file ERS220Info.h.

6.3.4.75 `const unsigned ERS220Info::NumMouthJoints = 0`

the number of joints that control the mouth

Definition at line 88 of file ERS220Info.h.

6.3.4.76 `const unsigned ERS220Info::NumOutputs = NumPIDJoints +
NumBinJoints + NumLEDs`

the total number of outputs

Definition at line 42 of file ERS220Info.h.

6.3.4.77 `const unsigned ERS220Info::NumPIDJoints = 15`

The number of joints which use PID motion - everything.

Right now all binary joints are slow, but perhaps this won't always be the case... hence the IsFast/Slow bitmasks to select which type, in order to be more general

Definition at line 39 of file ERS220Info.h.

6.3.4.78 const unsigned [ERS220Info::NumSensors](#) = 1+3+1+5

1 dist, 3 accel, 1 thermo, 5 from power, see SensorOffset_t

Definition at line 91 of file ERS220Info.h.

6.3.4.79 const unsigned int [ERS220Info::NumSlowFrames](#) = 1

the number of frames per buffer being sent to ears (double buffered as well)

Definition at line 33 of file ERS220Info.h.

6.3.4.80 const unsigned [ERS220Info::NumTailJoints](#) = 0

The number of joints assigned to the tail.

Definition at line 87 of file ERS220Info.h.

6.3.4.81 const unsigned [ERS220Info::outputNameLen](#) = 9

The length of the strings used for each of the outputs in outputNames (doesn't include null term).

Definition at line 293 of file ERS220Info.h.

6.3.4.82 const char* const [ERS220Info::outputNames](#)[NumOutputs]

A name of uniform length for referring to joints - handy for posture files, etc.

Definition at line 295 of file ERS220Info.h.

6.3.4.83 const double [ERS220Info::outputRanges](#)[NumOutputs][2]

Initial value:

```
{
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },
```

```

    { RAD(-82),RAD(43) }, { RAD(-89.6),RAD(89.6) }, { RAD(-29),RAD(29) },
    { 0,1 }, { 0,1 }, { 0,1 },
    { 0,1 }, { 0,1 }, { 0,1 },
    { 0,1 },
    { 0,1 }, { 0,1 }, { 0,1 },
    { 0,1 }, { 0,1 }, { 0,1 },
    { 0,1 }, { 0,1 }, { 0,1 },
    { 0,1 }, { 0,1 }, { 0,1 },
    { 0,1 }
}

```

This table holds the software limits of each of the outputs.

Definition at line 483 of file ERS220Info.h.

6.3.4.84 `const unsigned ERS220Info::PIDJointOffset = 0`

The beginning of the PID Joints.

Definition at line 101 of file ERS220Info.h.

6.3.4.85 `const char* const ERS220Info::PrimitiveName[NumOutputs]`

Initial value:

```

{
    "PRM:/r2/c1-Joint2:j1",
    "PRM:/r2/c1/c2-Joint2:j2",
    "PRM:/r2/c1/c2/c3-Joint2:j3",
    "PRM:/r4/c1-Joint2:j1",
    "PRM:/r4/c1/c2-Joint2:j2",
    "PRM:/r4/c1/c2/c3-Joint2:j3",

    "PRM:/r3/c1-Joint2:j1",
    "PRM:/r3/c1/c2-Joint2:j2",
    "PRM:/r3/c1/c2/c3-Joint2:j3",
    "PRM:/r5/c1-Joint2:j1",
    "PRM:/r5/c1/c2-Joint2:j2",
    "PRM:/r5/c1/c2/c3-Joint2:j3",

    "PRM:/r1/c1-Joint2:j1",
    "PRM:/r1/c1/c2-Joint2:j2",
    "PRM:/r1/c1/c2/c3-Joint2:j3",

    "PRM:/r1/c1/c2/c3/l1-LED2:l1",
    "PRM:/r1/c1/c2/c3/l4-LED2:l4",
    "PRM:/r1/c1/c2/c3/l2-LED2:l2",
    "PRM:/r1/c1/c2/c3/l5-LED2:l5",
    "PRM:/r1/c1/c2/c3/l3-LED2:l3",
    "PRM:/r1/c1/c2/c3/l6-LED2:l6",
    "PRM:/r1/c1/c2/c3/l7-LED2:l7",
}

```

```

"PRM:/r6/11-LED2:11",
"PRM:/r6/12-LED2:12",
"PRM:/r6/13-LED2:13",
"PRM:/r6/14-LED2:14",
"PRM:/r6/15-LED2:15",
"PRM:/r6/16-LED2:16",

"PRM:/r6/19-LED2:19",
"PRM:/r6/17-LED2:17",
"PRM:/r6/18-LED2:18",

"PRM:/r1/c1/c2/c3/18-LED2:18",
"PRM:/r1/c1/c2/c3/19-LED2:19",
"PRM:/r1/c1/c2/c3/1a-LED2:1a",
"PRM:/r1/c1/c2/c3/1b-LED2:1b",
}

```

the joint identifier strings used to refer to specific joints in OPEN-R (but not needed for others)

Warning:

IMPORTANT!!!! DO NOT CHANGE THE ORDER OF ITEMS IN THIS TABLE!!!

The offset consts defined in this file correspond to this table and will make life easier if you feel the need to reorder things, but they aren't used perfect *everywhere*

In particular, assumptions are made that the pid joints will be in slots 0-numPIDJoints and that the fast outputs (ie NOT ears) will be in slots 0-NumFastOutputs

There may be other assumptions not noted here!!!

Note:

These entries DON'T correspond to the CPC index numbers defined in [World-State](#) (this only lists joints, and in a different order defined by OPEN-R, that one has sensors as well)

Definition at line 347 of file ERS220Info.h.

6.3.4.86 `const LEDBitMask_t ERS220Info::RetractableHeadLEDMask = 1<<(RetractableHeadLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 197 of file ERS220Info.h.

6.3.4.87 `const unsigned int ERS220Info::SlowFrameTime = 128`

time between frames for the ears (which move slower for some reason, don't want to mix with other outputs) (milliseconds)

Definition at line 32 of file ERS220Info.h.

6.3.4.88 `const unsigned int ERS220Info::SoundBufferTime = 32`

the number of milliseconds per sound buffer... I'm not sure if this can be changed

Definition at line 34 of file ERS220Info.h.

6.3.4.89 `const LEDBitMask_t ERS220Info::TailCenterLEDMask = 1<<(TailCenterLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 192 of file ERS220Info.h.

6.3.4.90 `const LEDBitMask_t ERS220Info::TailLEDMask`

Initial value:

```
TailLeftLEDMask
| TailCenterLEDMask
| TailRightLEDMask
```

LEDs for tail.

Definition at line 235 of file ERS220Info.h.

6.3.4.91 `const LEDBitMask_t ERS220Info::TailLeftLEDMask = 1<<(TailLeftLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 191 of file ERS220Info.h.

6.3.4.92 `const LEDBitMask_t ERS220Info::TailRightLEDMask = 1<<(TailRightLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 193 of file ERS220Info.h.

6.3.4.93 `const LEDBitMask_t ERS220Info::TIBluLEDMask = 1<<(TIBluLEDOffset-LEDOffset)`

blue tail light

Definition at line 208 of file ERS220Info.h.

6.3.4.94 `const LEDBitMask_t ERS220Info::TIRedLEDMask = 1<<(TIRedLEDOffset-LEDOffset)`

red tail light

Definition at line 207 of file ERS220Info.h.

6.3.4.95 `const LEDBitMask_t ERS220Info::TopBrLEDMask = 1<<(TopBrLEDOffset-LEDOffset)`

top bar (green)

Definition at line 206 of file ERS220Info.h.

6.3.4.96 `const LEDBitMask_t ERS220Info::TopLLEDMask = 1<<(TopLLEDOffset-LEDOffset)`

top left (red - angry)

Definition at line 204 of file ERS220Info.h.

6.3.4.97 `const LEDBitMask_t ERS220Info::TopRLEDMask = 1<<(TopRLEDOffset-LEDOffset)`

top right (red - angry)

Definition at line 205 of file ERS220Info.h.

6.4 ERS2xxInfo Namespace Reference

6.4.1 Detailed Description

Contains information about the ERS-2xx series of robot, such as number of joints, PID defaults, timing information, etc.

This is a compatability mode, which allows dual-booting of the same memory stick on both models without needing to recompile, at the expense of (a little) runtime speed.

LED Bitmasks

Bitmasks for use when specifying combinations of LEDs (see LEDEngine) Note that L/R are robot's POV

- typedef unsigned int [LEDBitMask_t](#)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceFrontLeftLEDMask](#) = 1<<(FaceFrontLeftLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceFrontRightLEDMask](#) = 1<<(FaceFrontRightLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceCenterLeftLEDMask](#) = 1<<(FaceCenterLeftLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceCenterRightLEDMask](#) = 1<<(FaceCenterRightLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceBackLeftLEDMask](#) = 1<<(FaceBackLeftLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [FaceBackRightLEDMask](#) = 1<<(FaceBackRightLEDOffset-LEDOffset)
So you can be clear when you're referring to a LED bitmask.
- const [LEDBitMask_t](#) [ModeLEDMask](#) = 1<<(ModeLEDOffset-LEDOffset)

So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t BackLeft1LEDMask = 1<<(BackLeft1LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t BackLeft2LEDMask = 1<<(BackLeft2LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t BackLeft3LEDMask = 1<<(BackLeft3LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t BackRight3LEDMask = 1<<(BackRight3LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t BackRight2LEDMask = 1<<(BackRight2LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t BackRight1LEDMask = 1<<(BackRight1LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t TailLeftLEDMask = 1<<(TailLeftLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t TailCenterLEDMask = 1<<(TailCenterLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t TailRightLEDMask = 1<<(TailRightLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t FaceFrontBLEDMask = 1<<(FaceFrontBLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- `const LEDBitMask_t FaceFrontALEDMask = 1<<(FaceFrontALEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- const `LEDBitMask_t FaceFrontCLEDMask` = `1<<(FaceFrontCLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- const `LEDBitMask_t RetractableHeadLEDMask` = `1<<(RetractableHeadLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

- const `LEDBitMask_t TIRedLEDMask` = `1<<(TIRedLEDOffset-LEDOffset)`
red tail light

- const `LEDBitMask_t TIBluLEDMask` = `1<<(TIBluLEDOffset-LEDOffset)`
blue tail light

- const `LEDBitMask_t BotLLEDMask` = `1<<(BotLLEDOffset-LEDOffset)`
bottom left (red - sad)

- const `LEDBitMask_t BotRLEDMask` = `1<<(BotRLEDOffset-LEDOffset)`
bottom right (red - sad)

- const `LEDBitMask_t MidLLEDMask` = `1<<(MidLLEDOffset-LEDOffset)`
middle left (green - happy)

- const `LEDBitMask_t MidRLEDMask` = `1<<(MidRLEDOffset-LEDOffset)`
middle right (green - happy)

- const `LEDBitMask_t TopLLEDMask` = `1<<(TopLLEDOffset-LEDOffset)`
top left (red - angry)

- const `LEDBitMask_t TopRLEDMask` = `1<<(TopRLEDOffset-LEDOffset)`
top right (red - angry)

- const `LEDBitMask_t TopBrLEDMask` = `1<<(TopBrLEDOffset-LEDOffset)`
top bar (green)

- const `LEDBitMask_t FaceLEDMask`
LEDs for face.

- const `LEDBitMask_t HeadLEDMask`
LEDs on head (face plus retractable light).

- const [LEDBitMask_t BackLEDMask](#)
LEDs on back.
- const [LEDBitMask_t TailLEDMask](#)
LEDs for tail.
- const [LEDBitMask_t AllLEDMask](#) = ~0
selects all of the leds

Input Offsets

The order in which inputs should be stored

- enum [ButtonOffset_t](#) {
[LFrPawOffset](#) = LFrLegOrder, [RFrPawOffset](#) = RFrLegOrder, [LBkPawOffset](#) =
LBkLegOrder, [RBkPawOffset](#) = RBkLegOrder,
[ChinButOffset](#) = 4, [BackButOffset](#), [HeadFrButOffset](#), [HeadBkButOffset](#),
[TailLeftButOffset](#), [TailCenterButOffset](#), [TailRightButOffset](#) }
holds offsets to different buttons in [WorldState::buttons\[\]](#)
- enum [SensorOffset_t](#) {
[IRDistOffset](#) = 0, [BAccelOffset](#), [LAccelOffset](#), [DAccelOffset](#),
[ThermoOffset](#), [PowerRemainOffset](#), [PowerThermoOffset](#), [PowerCapacity-](#)
[Offset](#),
[PowerVoltageOffset](#), [PowerCurrentOffset](#) }
holds offset to different sensor values in [WorldState::sensors\[\]](#)

Output Types Information

- const unsigned [NumPIDJoints](#) = 18
The number of joints which use PID motion - everything.
- const unsigned [NumLEDs](#) = 22
The number LEDs which can be controlled.
- const unsigned [NumBinJoints](#) = 2
The number of binary joints (210 has ears).

- const unsigned [NumOutputs](#) = [NumPIDJoints](#) + [NumBinJoints](#) + [NumLEDs](#)
the total number of outputs
- const bool [IsFastOutput](#) [[NumOutputs](#)]
true for joints which can be updated every 32 ms (all but the ears on a 210)
- const bool [IsRealERS210](#) [[NumOutputs](#)]
true for joints which can be updated every 32 ms (all but the ears on a 210)
- const bool [IsRealERS220](#) [[NumOutputs](#)]
true for joints which can be updated every 32 ms (all but the ears on a 210)
- const unsigned [JointsPerLeg](#) = 3
The number of joints per leg.
- const unsigned [NumLegs](#) = 4
The number of legs.
- const unsigned [NumLegJoints](#) = [JointsPerLeg](#)*[NumLegs](#)
the TOTAL number of joints on ALL legs
- const unsigned [NumHeadJoints](#) = 3
The number of joints in the neck.
- const unsigned [NumTailJoints](#) = 2
The number of joints assigned to the tail.
- const unsigned [NumMouthJoints](#) = 1
the number of joints that control the mouth
- const unsigned [NumEarJoints](#) = 2
The number of joints which control the ears (NOT per ear, is total).
- const unsigned [NumButtons](#) = 11
the number of buttons that are available, see [ButtonOffset_t](#)
- const unsigned [NumSensors](#) = 1+3+1+5
1 dist, 3 accel, 1 thermo, 5 from power, see [SensorOffset_t](#)

Output Offsets

Corresponds to entries in PrimitiveName, defined at the end of this file

- const unsigned [PIDJointOffset](#) = 0
The beginning of the PID Joints.
- const unsigned [LegOffset](#) = [PIDJointOffset](#)
the offset of the beginning of the leg joints
- const unsigned [HeadOffset](#) = [LegOffset](#)+[NumLegJoints](#)
the offset of the beginning of the head joints
- const unsigned [TailOffset](#) = [HeadOffset](#)+[NumHeadJoints](#)
the offset of the beginning of the tail joints
- const unsigned [MouthOffset](#) = [TailOffset](#)+[NumTailJoints](#)
the offset of the beginning of the mouth joint
- const unsigned [LEDOffset](#) = [PIDJointOffset](#) + [NumPIDJoints](#)
the offset of LEDs in [WorldState::outputs](#) and [MotionCommand](#) functions
- const unsigned [BinJointOffset](#) = [LEDOffset](#)+[NumLEDs](#)
The beginning of the binary joints.
- const unsigned [EarOffset](#) = [BinJointOffset](#)
the offset of the beginning of the ear joints - note that ears aren't sensed. They can be flicked by the environment and you won't know. Nor will they be flicked back

Enumerations

- enum [LegOrder_t](#) { [LFrLegOrder](#) = 0, [RFrLegOrder](#), [LBkLegOrder](#), [RBkLegOrder](#) }
the ordering of legs
- enum [LegOffset_t](#) { [LFrLegOffset](#) = [LegOffset](#)+[LFrLegOrder](#)*[JointsPerLeg](#), [RFrLegOffset](#) = [LegOffset](#)+[RFrLegOrder](#)*[JointsPerLeg](#), [LBkLegOffset](#) = [LegOffset](#)+[LBkLegOrder](#)*[JointsPerLeg](#), [RBkLegOffset](#) = [LegOffset](#)+[RBkLegOrder](#)*[JointsPerLeg](#) }
The offsets of the individual legs.
- enum [REKOffset_t](#) { [RotatorOffset](#) = 0, [ElevatorOffset](#), [KneeOffset](#) }

The offsets within appendages (the legs) Note that the ordering matches the actual physical ordering of joints on the appendage (and not that of the head's TPOffset_t's).

- enum `TPOffset_t` { `TiltOffset` = 0, `PanOffset`, `RollOffset` }

The offsets of appendages with tilt (elevation), pan (heading), and roll joints (i.e. head) Note that the ordering matches the actual physical ordering of joints on the appendage (and not that of the leg's REKOffset_t's).

- enum `LEDOffset_t` {
`FaceFrontLeftLEDOffset` = `LEDOffset`, `FaceFrontRightLEDOffset`, `FaceCenterLeftLEDOffset`, `FaceCenterRightLEDOffset`,
`FaceBackLeftLEDOffset`, `FaceBackRightLEDOffset`, `ModeLEDOffset`, `BackLeft1LEDOffset`,
`BackLeft2LEDOffset`, `BackLeft3LEDOffset`, `BackRight3LEDOffset`, `BackRight2LEDOffset`,
`BackRight1LEDOffset`, `TailLeftLEDOffset`, `TailCenterLEDOffset`, `TailRightLEDOffset`,
`FaceFrontBLEDOffset`, `FaceFrontALEDOffset`, `FaceFrontCLEDOffset`, `RetractableHeadLEDOffset`,
`TIBluLEDOffset`, `TIRedLEDOffset`, `BotLLEDOffset` = `FaceFrontLeftLEDOffset`, `BotRLEDOffset` = `FaceFrontRightLEDOffset`,
`MidLLEDOffset` = `FaceCenterLeftLEDOffset`, `MidRLEDOffset` = `FaceCenterRightLEDOffset`, `TopLLEDOffset` = `FaceBackLeftLEDOffset`, `TopRLEDOffset` = `FaceBackRightLEDOffset`,
`TopBrLEDOffset` = `ModeLEDOffset` }

The offsets of the individual LEDs on the head and tail. Note that L/R are robot's POV. See also LEDBitMask_t.

- enum `MinMaxRange_t` { `MinRange`, `MaxRange` }

Defines the min and max index of entries in #outputRanges and #mechanicalLimits.

Variables

- const unsigned int `FrameTime` = 8
time between frames in the motion system (milliseconds)
- const unsigned int `NumFrames` = 4
the number of frames per buffer (don't forget also double buffered)
- const unsigned int `SlowFrameTime` = 128

time between frames for the ears (which move slower for some reason, don't want to mix with other outputs) (milliseconds)

- const unsigned int [NumSlowFrames](#) = 1
the number of frames per buffer being sent to ears (double buffered as well)
- const unsigned int [SoundBufferTime](#) = 32
the number of milliseconds per sound buffer... I'm not sure if this can be changed
- const unsigned [outputNameLen](#) = 9
The length of the strings used for each of the outputs in outputNames (doesn't include null term).
- const char *const [outputNames](#) [[NumOutputs](#)]
A name of uniform length for referring to joints - handy for posture files, etc.
- const char *const [PrimitiveName](#) [[NumOutputs](#)]
the joint identifier strings used to refer to specific joints in OPEN-R (but not needed for others)
- const float [DefaultPIDs](#) [[NumPIDJoints](#)][3]
This table holds the default PID values for each joint. [PIDMC](#).
- const float [MaxOutputSpeed](#) [[NumOutputs](#)]
These values are Sony's recommended maximum joint velocities, in rad/ms.
- const double [outputRanges](#) [[NumOutputs](#)][2]
This table holds the software limits of each of the outputs.
- const double [mechanicalLimits](#) [[NumOutputs](#)][2]
This table holds the mechanical limits of each of the outputs.

6.4.2 Typedef Documentation

6.4.2.1 typedef unsigned int [ERS2xxInfo::LEDBitMask_t](#)

So you can be clear when you're referring to a LED bitmask.

Definition at line 209 of file ERS2xxInfo.h.

6.4.3 Enumeration Type Documentation

6.4.3.1 enum [ERS2xxInfo::ButtonOffset_t](#)

holds offsets to different buttons in [WorldState::buttons\[\]](#)

Should be a straight mapping to the ButtonSourceIDs

Note that the chest (power) button is not a normal button. It kills power to the motors at a hardware level, and isn't sensed in the normal way. If you want to know when it is pressed (and you are about to shut down) see [PowerSourceID::PauseSID](#).

See also:

[WorldState::buttons](#)

ButtonSourceID_t

Enumeration values:

LFrPawOffset

RFrPawOffset

LBkPawOffset

RBkPawOffset

ChinButOffset

BackButOffset

HeadFrButOffset [ERS210Info::HeadFrButOffset](#), [ERS220Info::HeadFrButOffset](#).

HeadBkButOffset [ERS210Info::HeadBkButOffset](#), [ERS220Info::HeadBkButOffset](#).

TailLeftButOffset

TailCenterButOffset

TailRightButOffset

Definition at line 295 of file ERS2xxInfo.h.

6.4.3.2 enum [ERS2xxInfo::LEDOffset_t](#)

The offsets of the individual LEDs on the head and tail. Note that L/R are robot's POV. See also [LEDBitMask_t](#).

Enumeration values:

FaceFrontLeftLEDOffset head face side light (front left - blue) (ERS-220)

FaceFrontRightLEDOffset head face side light (front right - blue) (ERS-220)

FaceCenterLeftLEDOffset head face side light (center left - blue) (ERS-220)

FaceCenterRightLEDOffset head face side light (center right - blue) (ERS-220)

FaceBackLeftLEDOffset head face side light (back left - red) (ERS-220)

FaceBackRightLEDOffset head face side light (back right - red) (ERS-220)

ModeLEDOffset mode indicator (back of the head - orange) (ERS-220)

BackLeft1LEDOffset back multi-indicator (left #1 - blue) (ERS-220)

BackLeft2LEDOffset back multi-indicator (left #2 - blue) (ERS-220)

BackLeft3LEDOffset back multi-indicator (left #3 - blue) (ERS-220)

BackRight3LEDOffset back multi-indicator (right #3 - blue) (ERS-220)

BackRight2LEDOffset back multi-indicator (right #2 - blue) (ERS-220)

BackRight1LEDOffset back multi-indicator (right #1 - blue) (ERS-220)

TailLeftLEDOffset tail light (left - blue) (ERS-220)

TailCenterLEDOffset tail light (center - red) (ERS-220)

TailRightLEDOffset tail light (right - blue) (ERS-220)

FaceFrontBLEDOffset face front light B (blue) (ERS-220)

FaceFrontALEDOffset face front light A (blue) (ERS-220)

FaceFrontCLEDOffset face front light C (red) (ERS-220)

RetractableHeadLEDOffset retractable head light (ERS-220)

TIBluLEDOffset blue tail light (ERS-210)

TIRedLEDOffset red tail light (ERS-210)

BotLLEDOffset bottom left (red - sad) (ERS-210)

BotRLEDOffset bottom right (red - sad) (ERS-210)

MidLLEDOffset middle left (green - happy) (ERS-210)

MidRLEDOffset middle right (green - happy) (ERS-210)

TopLLEDOffset top left (red - angry) (ERS-210)

TopRLEDOffset top right (red - angry) (ERS-210)

TopBrLEDOffset top bar (green) (ERS-210)

Definition at line 172 of file ERS2xxInfo.h.

6.4.3.3 enum [ERS2xxInfo::LegOffset_t](#)

The offsets of the individual legs.

Enumeration values:

LFrLegOffset beginning of left front leg

RFrLegOffset beginning of right front leg

LBkLegOffset beginning of left back leg

RBkLegOffset beginning of right back leg

Definition at line 150 of file ERS2xxInfo.h.

6.4.3.4 enum [ERS2xxInfo::LegOrder_t](#)

the ordering of legs

Enumeration values:

- LFrLegOrder** left front leg
- RFrLegOrder** right front leg
- LBkLegOrder** left back leg
- RBkLegOrder** right back leg

Definition at line 142 of file ERS2xxInfo.h.

6.4.3.5 enum [ERS2xxInfo::MinMaxRange_t](#)

Defines the min and max index of entries in [outputRanges](#) and [mechanicalLimits](#).

Enumeration values:

- MinRange**
- MaxRange**

Definition at line 545 of file ERS2xxInfo.h.

6.4.3.6 enum [ERS2xxInfo::REKOffset_t](#)

The offsets within appendages (the legs) Note that the ordering matches the actual physical ordering of joints on the appendage (and not that of the head's [TPROffset_t](#)'s).

Enumeration values:

- RotatorOffset** moves leg forward or backward along body
- ElevatorOffset** moves leg toward or away from body
- KneeOffset** moves knee

Definition at line 158 of file ERS2xxInfo.h.

6.4.3.7 enum [ERS2xxInfo::SensorOffset_t](#)

holds offset to different sensor values in [WorldState::sensors\[\]](#)

See also:

[WorldState::sensors\[\]](#)

Enumeration values:

- IRDistOffset** in millimeters
- BAccelOffset** backward acceleration, in m/s^2 , negative if sitting on butt (positive for faceplant)
- LAccelOffset** acceleration to the robot's left, in m/s^2 , negative if lying on robot's left side
- DAccelOffset** downward acceleration, in m/s^2 , negative if standing up... be careful about the signs on all of these...
- ThermoOffset** in degrees Celcius
- PowerRemainOffset** percentage, 0-1
- PowerThermoOffset** degrees Celcius
- PowerCapacityOffset** milli-amp hours
- PowerVoltageOffset** volts
- PowerCurrentOffset** milli-amp negative values (maybe positive while charging?)

Definition at line 311 of file ERS2xxInfo.h.

6.4.3.8 enum [ERS2xxInfo::TPROffset_t](#)

The offsets of appendages with tilt (elevation), pan (heading), and roll joints (i.e. head)
 Note that the ordering matches the actual physical ordering of joints on the appendage
 (and not that of the leg's REKOffset.t's).

Enumeration values:

- TiltOffset** tilt/elevation (vertical)
- PanOffset** pan/heading (horizontal)
- RollOffset** roll (rotational)

Definition at line 165 of file ERS2xxInfo.h.

6.4.4 Variable Documentation**6.4.4.1 const [LEDBitMask_t](#) [ERS2xxInfo::AllLEDMask](#) = ~0**

selects all of the leds

Definition at line 274 of file ERS2xxInfo.h.

6.4.4.2 `const LEDBitMask_t ERS2xxInfo::BackLEDMask`

Initial value:

```
BackLeft1LEDMask  
| BackLeft2LEDMask  
| BackLeft3LEDMask  
| BackRight1LEDMask  
| BackRight2LEDMask  
| BackRight3LEDMask
```

LEDs on back.

Definition at line 260 of file ERS2xxInfo.h.

6.4.4.3 `const LEDBitMask_t ERS2xxInfo::BackLeft1LEDMask = 1<<(BackLeft1LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 217 of file ERS2xxInfo.h.

6.4.4.4 `const LEDBitMask_t ERS2xxInfo::BackLeft2LEDMask = 1<<(BackLeft2LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 218 of file ERS2xxInfo.h.

6.4.4.5 `const LEDBitMask_t ERS2xxInfo::BackLeft3LEDMask = 1<<(BackLeft3LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 219 of file ERS2xxInfo.h.

6.4.4.6 `const LEDBitMask_t ERS2xxInfo::BackRight1LEDMask = 1<<(BackRight1LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 222 of file ERS2xxInfo.h.

6.4.4.7 `const LEDBitMask_t ERS2xxInfo::BackRight2LEDMask = 1<<(BackRight2LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 221 of file ERS2xxInfo.h.

6.4.4.8 `const LEDBitMask_t ERS2xxInfo::BackRight3LEDMask = 1<<(BackRight3LEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 220 of file ERS2xxInfo.h.

6.4.4.9 `const unsigned ERS2xxInfo::BinJointOffset = LEDOffset+NumLEDs`

The beginning of the binary joints.

Definition at line 137 of file ERS2xxInfo.h.

6.4.4.10 `const LEDBitMask_t ERS2xxInfo::BotLLEDMask = 1<<(BotLLEDOffset-LEDOffset)`

bottom left (red - sad)

Definition at line 235 of file ERS2xxInfo.h.

6.4.4.11 `const LEDBitMask_t ERS2xxInfo::BotRLEDMask = 1<<(BotRLEDOffset-LEDOffset)`

bottom right (red - sad)

Definition at line 236 of file ERS2xxInfo.h.

6.4.4.12 `const float ERS2xxInfo::DefaultPIDs[NumPIDJoints][3]`

Initial value:

```
{
  { 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
  { 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
  { 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },
  { 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
  { 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
  { 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },
  { 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
}
```

```

{ 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
{ 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },
{ 0x16/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x08/(double)(1<<(16-0xF)) },
{ 0x14/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x06/(double)(1<<(16-0xF)) },
{ 0x23/(double)(1<<(16-0xE)), 0x04/(double)(1<<(16-0x2)), 0x05/(double)(1<<(16-0xF)) },

{ 0x0A/(double)(1<<(16-0xE)), 0x08/(double)(1<<(16-0x2)), 0x0C/(double)(1<<(16-0xF)) },
{ 0x0D/(double)(1<<(16-0xE)), 0x08/(double)(1<<(16-0x2)), 0x0B/(double)(1<<(16-0xF)) },
{ 0x0A/(double)(1<<(16-0xE)), 0x08/(double)(1<<(16-0x2)), 0x0C/(double)(1<<(16-0xF)) },

{ 0x0A/(double)(1<<(16-0xE)), 0x00/(double)(1<<(16-0x2)), 0x18/(double)(1<<(16-0xF)) },
{ 0x07/(double)(1<<(16-0xE)), 0x00/(double)(1<<(16-0x2)), 0x11/(double)(1<<(16-0xF)) },

{ 0x0E/(double)(1<<(16-0xE)), 0x08/(double)(1<<(16-0x2)), 0x10/(double)(1<<(16-0xF)) }
}

```

This table holds the default PID values for each joint. [PIDMC](#).

Definition at line 474 of file ERS2xxInfo.h.

6.4.4.13 `const unsigned ERS2xxInfo::EarOffset = BinJointOffset`

the offset of the beginning of the ear joints - note that ears aren't sensed. They can be flicked by the environment and you won't know. Nor will they be flicked back

Definition at line 138 of file ERS2xxInfo.h.

6.4.4.14 `const LEDBitMask_t ERS2xxInfo::FaceBackLeftLEDMask = 1<<(FaceBackLeftLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 214 of file ERS2xxInfo.h.

6.4.4.15 `const LEDBitMask_t ERS2xxInfo::FaceBackRightLEDMask = 1<<(FaceBackRightLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 215 of file ERS2xxInfo.h.

6.4.4.16 `const LEDBitMask_t ERS2xxInfo::FaceCenterLeftLEDMask = 1<<(FaceCenterLeftLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 212 of file ERS2xxInfo.h.

6.4.4.17 `const LEDBitMask_t ERS2xxInfo::FaceCenterRightLEDMask = 1<<(FaceCenterRightLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 213 of file ERS2xxInfo.h.

6.4.4.18 `const LEDBitMask_t ERS2xxInfo::FaceFrontALEDMask = 1<<(FaceFrontALEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 227 of file ERS2xxInfo.h.

6.4.4.19 `const LEDBitMask_t ERS2xxInfo::FaceFrontBLEDMask = 1<<(FaceFrontBLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 226 of file ERS2xxInfo.h.

6.4.4.20 `const LEDBitMask_t ERS2xxInfo::FaceFrontCLEDMask = 1<<(FaceFrontCLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 228 of file ERS2xxInfo.h.

6.4.4.21 `const LEDBitMask_t ERS2xxInfo::FaceFrontLeftLEDMask = 1<<(FaceFrontLeftLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 210 of file ERS2xxInfo.h.

6.4.4.22 `const LEDBitMask_t ERS2xxInfo::FaceFrontRightLEDMask = 1<<(FaceFrontRightLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 211 of file ERS2xxInfo.h.

6.4.4.23 `const LEDBitMask_t ERS2xxInfo::FaceLEDMask`

Initial value:

```

FaceFrontLeftLEDMask
| FaceFrontRightLEDMask
| FaceCenterLeftLEDMask
| FaceCenterRightLEDMask
| FaceBackLeftLEDMask
| FaceBackRightLEDMask
| FaceFrontALEDMask
| FaceFrontBLEDMask
| FaceFrontCLEDMask
| ModeLEDMask

```

LEDs for face.

Definition at line 244 of file ERS2xxInfo.h.

6.4.4.24 `const unsigned int ERS2xxInfo::FrameTime = 8`

time between frames in the motion system (milliseconds)

Definition at line 26 of file ERS2xxInfo.h.

6.4.4.25 `const LEDBitMask_t ERS2xxInfo::HeadLEDMask`

Initial value:

```

FaceLEDMask
| RetractableHeadLEDMask

```

LEDs on head (face plus retractable light).

Definition at line 256 of file ERS2xxInfo.h.

6.4.4.26 `const unsigned ERS2xxInfo::HeadOffset = LegOffset+NumLegJoints`

the offset of the beginning of the head joints

Definition at line 131 of file ERS2xxInfo.h.

6.4.4.27 `const bool ERS2xxInfo::IsFastOutput[NumOutputs]`

Initial value:

```

{
    true, true, true,
    true, true, true,
    true, true, true,

```

```
    true, true, true,
    true, true, true,
    true, true,
    true,

    true, true, true,
    true, true, true,
    true,
    true, true, true,
    true, true, true,
    true, true, true,
    true, true, true,
    true,
    true, true,

    false, false
}
```

true for joints which can be updated every 32 ms (all but the ears on a 210)

Definition at line 40 of file ERS2xxInfo.h.

6.4.4.28 const bool [ERS2xxInfo::IsRealERS210](#)[NumOutputs]

Initial value:

```
{

    true, true, true,
    true, true, true,
    true, true, true,
    true, true, true,
    true, true, true,
    true, true,
    true,

    true, true, true,
    true, true, true,
    true,
    false, false, false,
    false, false, false,
    false, false, false,
    false, false, false,
    false,
    true, true,

    true, true
}
```

true for joints which can be updated every 32 ms (all but the ears on a 210)

Definition at line 64 of file ERS2xxInfo.h.

6.4.4.29 `const bool ERS2xxInfo::IsRealERS220[NumOutputs]`

Initial value:

```

{
    true, true, true,
    true, true, true,
    true, true, true,
    true, true, true,
    true, true, true,
    false, false,
    false,

    true, true, true,
    true, true, true,
    true,
    true, true, true,
    true, true, true,
    true, true, true,
    true, true, true,
    true,
    false, false,

    false, false
}

```

true for joints which can be updated every 32 ms (all but the ears on a 210)

Definition at line 88 of file [ERS2xxInfo.h](#).

6.4.4.30 `const unsigned ERS2xxInfo::JointsPerLeg = 3`

The number of joints per leg.

Definition at line 111 of file [ERS2xxInfo.h](#).

6.4.4.31 `const unsigned ERS2xxInfo::LEDOffset = PIDJointOffset + NumPIDJoints`

the offset of LEDs in [WorldState::outputs](#) and [MotionCommand](#) functions

Definition at line 135 of file [ERS2xxInfo.h](#).

6.4.4.32 `const unsigned ERS2xxInfo::LegOffset = PIDJointOffset`

the offset of the beginning of the leg joints

Definition at line 130 of file [ERS2xxInfo.h](#).

6.4.4.33 `const float ERS2xxInfo::MaxOutputSpeed[NumOutputs]`

These values are Sony's recommended maximum joint velocities, in rad/ms.

a value ≤ 0 means infinite speed (e.g. LEDs)

Definition at line 501 of file ERS2xxInfo.h.

6.4.4.34 `const double ERS2xxInfo::mechanicalLimits[NumOutputs][2]`

Initial value:

```
{
  { RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },
  { RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },
  { RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },
  { RAD(-120),RAD(120) }, { RAD(-14),RAD(92) }, { RAD(-30),RAD(150) },

  { RAD(-85),RAD(46) }, { RAD(-92.6),RAD(92.6) }, { RAD(-32),RAD(32) },

  { RAD(-25),RAD(25) }, { RAD(-25),RAD(25) },

  { RAD(-50),RAD(0) },

  { 0,1 }, { 0,1 }, { 0,1 },
  { 0,1 }, { 0,1 }, { 0,1 },
  { 0,1 },
  { 0,1 }, { 0,1 }, { 0,1 },
  { 0,1 }, { 0,1 }, { 0,1 },
  { 0,1 }, { 0,1 }, { 0,1 },
  { 0,1 }, { 0,1 }, { 0,1 },
  { 0,1 },
  { 0,1 }, { 0,1 },

  { 0,1 }, { 0,1 }
}
```

This table holds the mechanical limits of each of the outputs.

Definition at line 575 of file ERS2xxInfo.h.

6.4.4.35 `const LEDBitMask_t ERS2xxInfo::MidLLEDMask = 1<<(MidLLEDOffset-LEDOffset)`

middle left (green - happy)

Definition at line 237 of file ERS2xxInfo.h.

6.4.4.36 `const LEDBitMask_t ERS2xxInfo::MidRLEDMask = 1<<(MidRLEDOffset-LEDOffset)`

middle right (green - happy)

Definition at line 238 of file ERS2xxInfo.h.

6.4.4.37 `const LEDBitMask_t ERS2xxInfo::ModeLEDMask = 1<<(ModeLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 216 of file ERS2xxInfo.h.

6.4.4.38 `const unsigned ERS2xxInfo::MouthOffset = TailOffset+NumTailJoints`

the offset of the beginning of the mouth joint

Definition at line 133 of file ERS2xxInfo.h.

6.4.4.39 `const unsigned ERS2xxInfo::NumBinJoints = 2`

The number of binary joints (210 has ears).

Definition at line 37 of file ERS2xxInfo.h.

6.4.4.40 `const unsigned ERS2xxInfo::NumButtons = 11`

the number of buttons that are available, see ButtonOffset.t

Definition at line 118 of file ERS2xxInfo.h.

6.4.4.41 `const unsigned ERS2xxInfo::NumEarJoints = 2`

The number of joints which control the ears (NOT per ear, is total).

Definition at line 117 of file ERS2xxInfo.h.

6.4.4.42 `const unsigned int ERS2xxInfo::NumFrames = 4`

the number of frames per buffer (don't forget also double buffered)

Definition at line 27 of file ERS2xxInfo.h.

6.4.4.43 `const unsigned ERS2xxInfo::NumHeadJoints = 3`

The number of joints in the neck.

Definition at line 114 of file ERS2xxInfo.h.

6.4.4.44 `const unsigned ERS2xxInfo::NumLEDs = 22`

The number LEDs which can be controlled.

Definition at line 36 of file ERS2xxInfo.h.

6.4.4.45 `const unsigned ERS2xxInfo::NumLegJoints = JointsPerLeg*NumLegs`

the TOTAL number of joints on ALL legs

Definition at line 113 of file ERS2xxInfo.h.

6.4.4.46 `const unsigned ERS2xxInfo::NumLegs = 4`

The number of legs.

Definition at line 112 of file ERS2xxInfo.h.

6.4.4.47 `const unsigned ERS2xxInfo::NumMouthJoints = 1`

the number of joints that control the mouth

Definition at line 116 of file ERS2xxInfo.h.

6.4.4.48 `const unsigned ERS2xxInfo::NumOutputs = NumPIDJoints +
NumBinJoints + NumLEDs`

the total number of outputs

Definition at line 38 of file ERS2xxInfo.h.

6.4.4.49 `const unsigned ERS2xxInfo::NumPIDJoints = 18`

The number of joints which use PID motion - everything.

Right now all binary joints are slow, but perhaps this won't always be the case... hence the IsFast/Slow bitmasks to select which type, in order to be more general

Definition at line 35 of file ERS2xxInfo.h.

6.4.4.50 `const unsigned ERS2xxInfo::NumSensors = 1+3+1+5`

1 dist, 3 accel, 1 thermo, 5 from power, see SensorOffset.t

Definition at line 119 of file ERS2xxInfo.h.

6.4.4.51 `const unsigned int ERS2xxInfo::NumSlowFrames = 1`

the number of frames per buffer being sent to ears (double buffered as well)

Definition at line 29 of file ERS2xxInfo.h.

6.4.4.52 `const unsigned ERS2xxInfo::NumTailJoints = 2`

The number of joints assigned to the tail.

Definition at line 115 of file ERS2xxInfo.h.

6.4.4.53 `const unsigned ERS2xxInfo::outputNameLen = 9`

The length of the strings used for each of the outputs in outputNames (doesn't include null term).

Definition at line 328 of file ERS2xxInfo.h.

6.4.4.54 `const char* const ERS2xxInfo::outputNames[NumOutputs]`

A name of uniform length for referring to joints - handy for posture files, etc.

Definition at line 330 of file ERS2xxInfo.h.

6.4.4.55 `const double ERS2xxInfo::outputRanges[NumOutputs][2]`

Initial value:

```
{
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },
  { RAD(-117),RAD(117) }, { RAD(-11),RAD(89) }, { RAD(-27),RAD(147) },

  { RAD(-82),RAD(43) }, { RAD(-89.6),RAD(89.6) }, { RAD(-29),RAD(29) },

  { RAD(-22),RAD(22) }, { RAD(-22),RAD(22) },

  { RAD(-47),RAD(-3) },
```

```

    {0,1},{0,1},{0,1},
    {0,1},{0,1},{0,1},
    {0,1},
    {0,1},{0,1},{0,1},
    {0,1},{0,1},{0,1},
    {0,1},{0,1},{0,1},
    {0,1},{0,1},{0,1},
    {0,1},
    {0,1},{0,1},

    {0,1},{0,1}
}

```

This table holds the software limits of each of the outputs.

Definition at line 548 of file ERS2xxInfo.h.

6.4.4.56 const unsigned ERS2xxInfo::PIDJointOffset = 0

The beginning of the PID Joints.

Definition at line 129 of file ERS2xxInfo.h.

6.4.4.57 const char* const ERS2xxInfo::PrimitiveName[NumOutputs]

Initial value:

```

{
    "PRM:/r2/c1-Joint2:j1",
    "PRM:/r2/c1/c2-Joint2:j2",
    "PRM:/r2/c1/c2/c3-Joint2:j3",
    "PRM:/r4/c1-Joint2:j1",
    "PRM:/r4/c1/c2-Joint2:j2",
    "PRM:/r4/c1/c2/c3-Joint2:j3",

    "PRM:/r3/c1-Joint2:j1",
    "PRM:/r3/c1/c2-Joint2:j2",
    "PRM:/r3/c1/c2/c3-Joint2:j3",
    "PRM:/r5/c1-Joint2:j1",
    "PRM:/r5/c1/c2-Joint2:j2",
    "PRM:/r5/c1/c2/c3-Joint2:j3",

    "PRM:/r1/c1-Joint2:j1",
    "PRM:/r1/c1/c2-Joint2:j2",
    "PRM:/r1/c1/c2/c3-Joint2:j3",

    "PRM:/r6/c2-Joint2:j2",
    "PRM:/r6/c1-Joint2:j1",

    "PRM:/r1/c1/c2/c3/c4-Joint2:j4",

    "PRM:/r1/c1/c2/c3/l1-LED2:l1",
    "PRM:/r1/c1/c2/c3/l4-LED2:l4",
}

```

```

"PRM:/r1/c1/c2/c3/l2-LED2:l2",
"PRM:/r1/c1/c2/c3/l5-LED2:l5",
"PRM:/r1/c1/c2/c3/l3-LED2:l3",
"PRM:/r1/c1/c2/c3/l6-LED2:l6",
"PRM:/r1/c1/c2/c3/l7-LED2:l7",

"PRM:/r6/l1-LED2:l1",
"PRM:/r6/l2-LED2:l2",
"PRM:/r6/l3-LED2:l3",
"PRM:/r6/l4-LED2:l4",
"PRM:/r6/l5-LED2:l5",
"PRM:/r6/l6-LED2:l6",

"PRM:/r6/l9-LED2:l9",
"PRM:/r6/l7-LED2:l7",
"PRM:/r6/l8-LED2:l8",

"PRM:/r1/c1/c2/c3/l8-LED2:l8",
"PRM:/r1/c1/c2/c3/l9-LED2:l9",
"PRM:/r1/c1/c2/c3/la-LED2:la",
"PRM:/r1/c1/c2/c3/lb-LED2:lb",

"PRM:/r6/l2-LED2:l2",
"PRM:/r6/l1-LED2:l1",

"PRM:/r1/c1/c2/c3/e1-Joint3:j5",
"PRM:/r1/c1/c2/c3/e2-Joint3:j6"
}

```

the joint identifier strings used to refer to specific joints in OPEN-R (but not needed for others)

Warning:

IMPORTANT!!!! DO NOT CHANGE THE ORDER OF ITEMS IN THIS TABLE!!!

The offset consts defined in this file correspond to this table and will make life easier if you feel the need to reorder things, but they aren't used perfect *everywhere*

In particular, assumptions are made that the pid joints will be in slots 0-numPIDJoints and that the fast outputs (ie NOT ears) will be in slots 0-NumFastOutputs

There may be other assumptions not noted here!!!

Note:

These entries DON'T correspond to the CPC index numbers defined in [World-State](#) (this only lists joints, and in a different order defined by OPEN-R, that one has sensors as well

Definition at line 393 of file ERS2xxInfo.h.

6.4.4.58 `const LEDBitMask_t ERS2xxInfo::RetractableHeadLEDMask = 1<<(RetractableHeadLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 229 of file ERS2xxInfo.h.

6.4.4.59 `const unsigned int ERS2xxInfo::SlowFrameTime = 128`

time between frames for the ears (which move slower for some reason, don't want to mix with other outputs) (milliseconds)

Definition at line 28 of file ERS2xxInfo.h.

6.4.4.60 `const unsigned int ERS2xxInfo::SoundBufferTime = 32`

the number of milliseconds per sound buffer... I'm not sure if this can be changed

Definition at line 30 of file ERS2xxInfo.h.

6.4.4.61 `const LEDBitMask_t ERS2xxInfo::TailCenterLEDMask = 1<<(TailCenterLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 224 of file ERS2xxInfo.h.

6.4.4.62 `const LEDBitMask_t ERS2xxInfo::TailLEDMask`

Initial value:

```
TailLeftLEDMask
| TailCenterLEDMask
| TailRightLEDMask
| TlRedLEDMask
| TlBluLEDMask
```

LEDs for tail.

Definition at line 268 of file ERS2xxInfo.h.

6.4.4.63 `const LEDBitMask_t ERS2xxInfo::TailLeftLEDMask = 1<<(TailLeftLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 223 of file ERS2xxInfo.h.

6.4.4.64 `const unsigned ERS2xxInfo::TailOffset = HeadOffset+NumHeadJoints`

the offset of the beginning of the tail joints

Definition at line 132 of file ERS2xxInfo.h.

6.4.4.65 `const LEDBitMask_t ERS2xxInfo::TailRightLEDMask = 1<<(TailRightLEDOffset-LEDOffset)`

So you can be clear when you're referring to a LED bitmask.

Definition at line 225 of file ERS2xxInfo.h.

6.4.4.66 `const LEDBitMask_t ERS2xxInfo::TIBluLEDMask = 1<<(TIBluLEDOffset-LEDOffset)`

blue tail light

Definition at line 232 of file ERS2xxInfo.h.

6.4.4.67 `const LEDBitMask_t ERS2xxInfo::TIRedLEDMask = 1<<(TIRedLEDOffset-LEDOffset)`

red tail light

Definition at line 231 of file ERS2xxInfo.h.

6.4.4.68 `const LEDBitMask_t ERS2xxInfo::TopBrLEDMask = 1<<(TopBrLEDOffset-LEDOffset)`

top bar (green)

Definition at line 241 of file ERS2xxInfo.h.

6.4.4.69 `const LEDBitMask_t ERS2xxInfo::TopLLEDMask = 1<<(TopLLEDOffset-LEDOffset)`

top left (red - angry)

Definition at line 239 of file ERS2xxInfo.h.

6.4.4.70 `const LEDBitMask_t ERS2xxInfo::TopRLEDMask = 1<<(TopRLEDOffset-LEDOffset)`

top right (red - angry)

Definition at line 240 of file ERS2xxInfo.h.

6.5 GVector Namespace Reference

Compounds

- class [vector2d](#)
- class [vector3d](#)

Functions

- template<class num> num [dot](#) (const [vector3d](#)< num > a, const [vector3d](#)< num > b)
- template<class num> [vector3d](#)< num > [cross](#) (const [vector3d](#)< num > a, const [vector3d](#)< num > b)
- [VECTOR3D_EQUAL_BINARY_OPERATOR](#) (+=) VECTOR3D_EQUAL_BINARY_OPERATOR(-=) VECTOR3D_EQUAL_BINARY_OPERATOR(*=) VECTOR3D_EQUAL_BINARY_OPERATOR(/=)#define VECTOR3D_BINARY_OPERATOR(opr) VECTOR3D_BINARY_OPERATOR(+) VECTOR3D_BINARY_OPERATOR(-) VECTOR3D_BINARY_OPERATOR(*) VECTOR3D_BINARY_OPERATOR(/)#define VECTOR3D_SCALAR_OPERATOR(opr) VECTOR3D_SCALAR_OPERATOR(*) VECTOR3D_SCALAR_OPERATOR(/)#define VECTOR3D_EQUAL_SCALAR_OPERATOR(opr) VECTOR3D_EQUAL_SCALAR_OPERATOR(*=) VECTOR3D_EQUAL_SCALAR_OPERATOR(/=)#define VECTOR3D_LOGIC_OPERATOR(opr
- combine [VECTOR3D_LOGIC_OPERATOR](#) (==,&&) VECTOR3D_LOGIC_OPERATOR(!=
- combine [VECTOR3D_LOGIC_OPERATOR](#) (< ,&&) VECTOR3D_LOGIC_OPERATOR(>
- combine && [VECTOR3D_LOGIC_OPERATOR](#) (<=,&&) VECTOR3D_LOGIC_OPERATOR(>=
- template<class num> num [distance](#) (const [vector3d](#)< num > a, const [vector3d](#)< num > b)
- template<class num> num [sdistance](#) (const [vector3d](#)< num > a, const [vector3d](#)< num > b)
- template<class num> num [distance_to_line](#) (const [vector3d](#)< num > x0, const [vector3d](#)< num > x1, const [vector3d](#)< num > p)
- template<class num> num [dot](#) (const [vector2d](#)< num > a, const [vector2d](#)< num > b)
- [VECTOR2D_EQUAL_BINARY_OPERATOR](#) (+=) VECTOR2D_EQUAL_BINARY_OPERATOR(-=) VECTOR2D_EQUAL_BINARY_OPERATOR(*=) VECTOR2D_EQUAL_BINARY_OPERATOR(/=)#define VECTOR2D_BINARY_OPERATOR(opr) VECTOR2D_BINARY_OPERATOR(+) VECTOR2D_BINARY_OPERATOR(-) VECTOR2D_BINARY_OPERATOR(*)

```

VECTOR2D_BINARY_OPERATOR(/)#define      VECTOR2D_SCALAR_-
OPERATOR(opr) VECTOR2D_SCALAR_OPERATOR(*) VECTOR2D_-
SCALAR_OPERATOR(/)#define      VECTOR2D_EQUAL_SCALAR_-
OPERATOR(opr) VECTOR2D_EQUAL_SCALAR_OPERATOR(=) VEC-
TOR2D_EQUAL_SCALAR_OPERATOR(/)#define    VECTOR2D_LOGIC_-
OPERATOR(opr

```

- combine [VECTOR2D_LOGIC_OPERATOR](#) (==,&&) VECTOR2D_LOGIC_OPERATOR(!=
- combine [VECTOR2D_LOGIC_OPERATOR](#) (< ,&&) VECTOR2D_LOGIC_OPERATOR(>
- combine && [VECTOR2D_LOGIC_OPERATOR](#) (<=,&&) VECTOR2D_LOGIC_OPERATOR(>=
- template<class num> num [distance](#) (const [vector2d](#)< num > a, const [vector2d](#)< num > b)
- template<class num> num [sdistance](#) (const [vector2d](#)< num > a, const [vector2d](#)< num > b)
- template<class num> num [distance_to_line](#) (const [vector2d](#)< num > x0, const [vector2d](#)< num > x1, const [vector2d](#)< num > p)
- template<class num> num [offset_to_line](#) (const [vector2d](#)< num > x0, const [vector2d](#)< num > x1, const [vector2d](#)< num > p)
- template<class vector> vector [point_on_segment](#) (const vector x0, const vector x1, const vector p)
- template<class vector> bool [intersect_ray_plane](#) (const vector r0, const vector rd, const vector p0, const vector pn, vector &result)

6.5.1 Function Documentation

6.5.1.1 template<class num> [vector3d](#)<num> cross (const [vector3d](#)< num > a, const [vector3d](#)< num > b)

Definition at line 158 of file gvector.h.

References [GVector::vector3d< double >::x](#), [GVector::vector3d< double >::y](#), and [GVector::vector3d< double >::z](#).

6.5.1.2 template<class num> num distance (const [vector2d](#)< num > a, const [vector2d](#)< num > b)

Definition at line 561 of file gvector.h.

References [dx](#), [GVector::vector2d< num >::x](#), and [GVector::vector2d< num >::y](#).

6.5.1.3 `template<class num> num distance (const vector3d< num > a, const vector3d< num > b)`

Definition at line 308 of file gvector.h.

References `dx`, `GVector::vector3d< double >::x`, `GVector::vector3d< double >::y`, and `GVector::vector3d< double >::z`.

6.5.1.4 `template<class num> num distance_to_line (const vector2d< num > x0, const vector2d< num > x1, const vector2d< num > p)`

Definition at line 585 of file gvector.h.

References `distance()`, `GVector::vector2d< num >::x`, and `GVector::vector2d< num >::y`.

6.5.1.5 `template<class num> num distance_to_line (const vector3d< num > x0, const vector3d< num > x1, const vector3d< num > p)`

Definition at line 334 of file gvector.h.

References `distance()`, `GVector::vector3d< double >::x`, `GVector::vector3d< double >::y`, and `GVector::vector3d< double >::z`.

6.5.1.6 `template<class num> num dot (const vector2d< num > a, const vector2d< num > b)`

Definition at line 447 of file gvector.h.

References `GVector::vector2d< num >::x`, and `GVector::vector2d< num >::y`.

6.5.1.7 `template<class num> num dot (const vector3d< num > a, const vector3d< num > b)`

Definition at line 139 of file gvector.h.

References `GVector::vector3d< double >::x`, `GVector::vector3d< double >::y`, and `GVector::vector3d< double >::z`.

6.5.1.8 `template<class vector> bool intersect_ray_plane (const vector r0, const vector rd, const vector p0, const vector pn, vector & result)`

Definition at line 641 of file gvector.h.

6.5.1.9 `template<class num> num offset_to_line (const vector2d< num > x0,
const vector2d< num > x1, const vector2d< num > p)`

Definition at line 600 of file gvector.h.

References `GVector::vector2d< num >::dot()`, `GVector::vector2d< num >::set()`, `GVector::vector2d< num >::x`, and `GVector::vector2d< num >::y`.

6.5.1.10 `template<class vector> vector point_on_segment (const vector x0,
const vector x1, const vector p)`

Definition at line 618 of file gvector.h.

References `dot()`, and `dx`.

6.5.1.11 `template<class num> num sdistance (const vector2d< num > a, const
vector2d< num > b)`

Definition at line 573 of file gvector.h.

References `dx`, `GVector::vector2d< num >::x`, and `GVector::vector2d< num >::y`.

6.5.1.12 `template<class num> num sdistance (const vector3d< num > a, const
vector3d< num > b)`

Definition at line 321 of file gvector.h.

References `dx`, `GVector::vector3d< double >::x`, `GVector::vector3d< double >::y`, and `GVector::vector3d< double >::z`.

6.5.1.13 VECTOR2D_EQUAL_BINARY_OPERATOR (+)

6.5.1.14 combine && VECTOR2D_LOGIC_OPERATOR (<=, &&)

6.5.1.15 combine VECTOR2D_LOGIC_OPERATOR ()

6.5.1.16 combine VECTOR2D_LOGIC_OPERATOR (&&)

6.5.1.17 VECTOR3D_EQUAL_BINARY_OPERATOR (+)

6.5.1.18 combine && VECTOR3D_LOGIC_OPERATOR (<=, &&)

6.5.1.19 combine VECTOR3D_LOGIC_OPERATOR ()

6.5.1.20 combine VECTOR3D_LOGIC_OPERATOR (&&)

6.6 mathutils Namespace Reference

Functions

- `template<class num> num squareDistance (num x1, num ya, num x2, num yb)`
- `template<class num> num distance (num x1, num ya, num x2, num yb)`
- `template<class num> num limitRange (num n, num low, num high)`
- `template<class num> num squared (num n)`
- `template<class num> num abs.t (num n)`
- `template<class num> num log2t (num x)`
- `template<> float log2t (float x)`
- `template<> double log2t (double x)`

6.6.1 Function Documentation

6.6.1.1 `template<class num> num abs.t (num n)` [inline]

Definition at line 31 of file mathutils.h.

6.6.1.2 `template<class num> num distance (num x1, num ya, num x2, num yb)` [inline]

Definition at line 14 of file mathutils.h.

References `squareDistance()`.

6.6.1.3 `template<class num> num limitRange (num n, num low, num high)` [inline]

Definition at line 19 of file mathutils.h.

6.6.1.4 `template<> double log2t (double x)` [inline]

Definition at line 52 of file mathutils.h.

6.6.1.5 `template<> float log2t (float x)` [inline]

Definition at line 48 of file mathutils.h.

6.6.1.6 `template<class num> num log2t (num x)` [inline]

Definition at line 36 of file mathutils.h.

6.6.1.7 `template<class num> num squared (num n) [inline]`

Definition at line 26 of file mathutils.h.

6.6.1.8 `template<class num> num squareDistance (num x1, num ya, num x2,
num yb) [inline]`

Definition at line 9 of file mathutils.h.

6.7 PowerSourceID Namespace Reference

6.7.1 Detailed Description

holds source ID types for power events [EventBase](#) `PowerSourceID.t`

Enumerations

- enum `PowerSourceID.t` {
[PauseSID](#) = 0, [MotorPowerSID](#), [VibrationSID](#), [BatteryEmptySID](#),
[LowPowerWarnSID](#), [BatteryFullSID](#), [ExternalPowerSID](#), [ExternalPortSID](#),
[BatteryConnectSID](#), [BatteryInitSID](#), [DischargingSID](#), [ChargingSID](#),
[OverheatingSID](#), [PowerGoodSID](#), [ChargerStatusSID](#), [SuspendedSID](#),
[OverChargedSID](#), [TermDischargeSID](#), [TermChargeSID](#), [ErrorSID](#),
[StationConnectSID](#), [BatteryOverCurrentSID](#), [DataFromStationSID](#), [Register-UpdateSID](#),
[RTCSID](#), [SpecialModeSID](#), [BMNDebugModeSID](#), [PlungerSID](#),
[UpdatedSID](#), [NumPowerSIDs](#) }

holds source ID types for power events

6.7.2 Enumeration Type Documentation

6.7.2.1 enum `PowerSourceID::PowerSourceID.t`

holds source ID types for power events

Also serve as offsets into `::powerFlags[]` I've never seen a lot of these events thrown by the OS. NS means never-seen, which could simply be because i haven't put it in that situation (don't have a station-type power charger) or because the OS doesn't actually support sending that flag.

Under normal conditions, you'll see `MotorPowerSID`, `BatteryConnectSID`, `DischargingSID`, and `PowerGoodSID` always active with occasional `VibrationSID` and `UpdateSID`. When the chest button is pushed, `PauseSID` is activated and `MotorPowerSID` is deactivated.

The [BatteryMonitorBehavior](#) will give a warning once power begins getting low. The OS won't boot off a battery with less than 15% power remaining (which is when the `LowPowerWarnSID` is thrown)

Note:

there's not a one-to-one correspondance of the events from the OPENR power system... i map several OPENR events to fewer Tekkotsu events, check the name if you want to know the specific source (say if low battery is low current and/or low voltage) Status ETIDS are only generated when one of a related group goes on/off but others are still active

Enumeration values:

PauseSID the chest button was pushed (this is not a normal button, it kills power to the motors in hardware)

MotorPowerSID active while the motors have power

VibrationSID triggered when the OS decides a large acceleration has occurred, like falling down (or more specifically, hitting the ground afterward)

BatteryEmptySID battery is dead

LowPowerWarnSID triggered when `sensors[PowerRemainOffset] <= 0.15` (PowerGoodSID stays on)

BatteryFullSID battery is full

ExternalPowerSID receiving power from an external source (such as AC cable, may or may not include the "station", see StationConnectSID)

ExternalPortSID an external power source is plugged in (does not imply current is flowing however)

BatteryConnectSID a battery is plugged in

BatteryInitSID ? NS

DischargingSID using power from the battery (although still stays on after hooked up to external power)

ChargingSID you used to be able to charge while running, tho that has changed in more recent versions of OPEN-R. In any case, I never saw this even when it did work.

OverheatingSID in case the robot starts getting too hot NS

PowerGoodSID there is power, either from external or battery

ChargerStatusSID ? NS

SuspendedSID ? NS

OverChargedSID in case the charger screws up somehow (?) NS

TermDischargeSID end of battery (?) NS

TermChargeSID end of charging (?) NS

ErrorSID general power error NS

StationConnectSID connected to a station NS

BatteryOverCurrentSID similar to OverChargedSID (?) NS

DataFromStationSID ? NS

RegisterUpdateSID ? NS

RTCSID ? NS

SpecialModeSID ? NS

BMNDebugModeSID ? NS

PlungerSID I think this is in reference to having a memorystick (?) NS.

UpdatedSID sent as last event after processing a frame

NumPowerSIDs

Definition at line 79 of file WorldState.h.

6.8 RobotInfo Namespace Reference

6.8.1 Detailed Description

Contains information about the robot, such as number of joints, PID defaults, timing information, etc.

6.9 SensorSourceID Namespace Reference

6.9.1 Detailed Description

holds source ID types for sensor events [EventBase](#) [SensorSourceID.t](#)

Enumerations

- enum [SensorSourceID.t](#) { [UpdatedSID](#) }
holds source ID types for sensor events

6.9.2 Enumeration Type Documentation

6.9.2.1 enum [SensorSourceID::SensorSourceID.t](#)

holds source ID types for sensor events

Enumeration values:

UpdatedSID sends status event as last event after processing a frame

Definition at line 49 of file WorldState.h.

6.10 SocketNS Namespace Reference

Enumerations

- enum `TransportType_t` { `SOCK_STREAM` = 0, `SOCK_DGRAM` }
Specifies transport type. TCP is usually a good idea.
- enum `ConnectionState` {
`CONNECTION_CLOSED`, `CONNECTION_CONNECTING`, `CONNECTION_CONNECTED`, `CONNECTION_LISTENING`,
`CONNECTION_CLOSING`, `CONNECTION_ERROR` }
Internal TCP/UDP Connection State.
- enum `FlushType_t` { `FLUSH_NONBLOCKING` = 0, `FLUSH_BLOCKING` }
Chooses between blocking and non-blocking [Wireless](#) Input, Output. Blocking wireless output from the main process will affect the performance of the Aibo, and should only be used for debugging purposes.

6.10.1 Enumeration Type Documentation

6.10.1.1 enum `SocketNS::ConnectionState`

Internal TCP/UDP Connection State.

Enumeration values:

`CONNECTION_CLOSED`
`CONNECTION_CONNECTING`
`CONNECTION_CONNECTED`
`CONNECTION_LISTENING`
`CONNECTION_CLOSING`
`CONNECTION_ERROR`

Definition at line 16 of file `Socket.h`.

6.10.1.2 enum `SocketNS::FlushType_t`

Chooses between blocking and non-blocking [Wireless](#) Input, Output. Blocking wireless output from the main process will affect the performance of the Aibo, and should only be used for debugging purposes.

Enumeration values:

FLUSH_NONBLOCKING Writes and Reads return immediately, and are processed by another process, so Main can continue to run. Non-blocking reads require specifying a callback function to handle data received.

FLUSH_BLOCKING Blocking writes are a good idea for debugging - a blocking write will be transmitted before execution continues to the next statement. Blocking reads should be avoided, since they'll cause a significant slow down in the main process.

Definition at line 26 of file Socket.h.

6.10.1.3 enum [SocketNS::TransportType_t](#)

Specifies transport type. TCP is usually a good idea.

Enumeration values:

SOCK_STREAM TCP: guaranteed delivery, higher overhead.

SOCK_DGRAM UDP: no guarantees, low overhead.

Definition at line 10 of file Socket.h.

6.11 std Namespace Reference

6.12 VisionEventNS Namespace Reference

6.12.1 Detailed Description

contains source IDs for the objects we can recognize

Enumerations

- enum [VisionSourceID_t](#) {
 [RedBallSID](#) = 0, [PinkBallSID](#), [HandSID](#), [ThumbsupSID](#),
 [ThingSID](#), [MarkersSID](#) }

contains source IDs for the objects we can recognize

6.12.2 Enumeration Type Documentation

6.12.2.1 enum [VisionEventNS::VisionSourceID_t](#)

contains source IDs for the objects we can recognize

Enumeration values:

RedBallSID a red bean bag ball

PinkBallSID the plastic pink ball Aibos ship with

HandSID Optimized for Ethan's hand, a pasty white thing.

ThumbsupSID Recognizes when Ethan is giving a thumbs up.

ThingSID Other stuff.

MarkersSID never actually generated: data directly reported to WorldModel

Definition at line 10 of file VisionEvent.h.

6.13 VisionInterface Namespace Reference

Compounds

- struct [ObjectInfo](#)
- struct [VObject](#)

Functions

- void [WriteThresholdImage](#) ([Vision](#) *vision, char *filename)
- int [SetThreshold](#) ([Vision](#) *vision, int threshold_id)
- void [SendRawImage](#) ([Vision](#) *vision)
- void [SendRLEImage](#) ([Vision](#) *vision)

Variables

- const double [HorzFOV](#) = 58.0 * M_PI / 180.0
- const double [VertFOV](#) = 48.0 * M_PI / 180.0
- const uchar [OFF_EDGE_LEFT](#) = 1
- const uchar [OFF_EDGE_RIGHT](#) = 2
- const uchar [OFF_EDGE_TOP](#) = 4
- const uchar [OFF_EDGE_BOTTOM](#) = 8
- const int [MARKER_GOG](#) = 0
- const int [MARKER_GOP](#) = 1
- const int [MARKER_GPG](#) = 2
- const int [MARKER_GPO](#) = 3
- const int [MARKER_OGO](#) = 4
- const int [MARKER_OGP](#) = 5
- const int [MARKER_OPG](#) = 6
- const int [MARKER_OPO](#) = 7
- const int [MARKER_PGO](#) = 8
- const int [MARKER_PGP](#) = 9
- const int [MARKER_POG](#) = 10
- const int [MARKER_POP](#) = 11
- const int [RBALL](#) = 12
- const int [PBALL](#) = 13
- const int [HAND](#) = 14
- const int [THING](#) = 15
- const int [NUM_MARKERS](#) = 12
- const int [NUM_VISION_OBJECTS](#) = [NUM_MARKERS](#) + 4

6.13.1 Function Documentation

6.13.1.1 void SendRawImage ([Vision](#) * *vision*)

6.13.1.2 void SendRLEImage ([Vision](#) * *vision*)

6.13.1.3 int VisionInterface::SetThreshold ([Vision](#) * *vision*, int *threshold_id*)

Definition at line 1134 of file Vision.cc.

References Vision::setThreshold(), and vision.

6.13.1.4 void VisionInterface::WriteThresholdImage ([Vision](#) * *vision*, char * *filename*)

Definition at line 1130 of file Vision.cc.

References Vision::saveThresholdImage(), and vision.

6.13.2 Variable Documentation

6.13.2.1 const int [VisionInterface::HAND](#) = 14

Definition at line 71 of file VisionInterface.h.

6.13.2.2 const double [VisionInterface::HorzFOV](#) = 58.0 * M_PI / 180.0
[static]

Definition at line 38 of file VisionInterface.h.

6.13.2.3 const int [VisionInterface::MARKER_GOG](#) = 0

Definition at line 57 of file VisionInterface.h.

6.13.2.4 const int [VisionInterface::MARKER_GOP](#) = 1

Definition at line 58 of file VisionInterface.h.

6.13.2.5 const int [VisionInterface::MARKER_GPG](#) = 2

Definition at line 59 of file VisionInterface.h.

6.13.2.6 `const int VisionInterface::MARKER_GPO = 3`

Definition at line 60 of file VisionInterface.h.

6.13.2.7 `const int VisionInterface::MARKER_OGO = 4`

Definition at line 61 of file VisionInterface.h.

6.13.2.8 `const int VisionInterface::MARKER_OGP = 5`

Definition at line 62 of file VisionInterface.h.

6.13.2.9 `const int VisionInterface::MARKER_OPG = 6`

Definition at line 63 of file VisionInterface.h.

6.13.2.10 `const int VisionInterface::MARKER_OPO = 7`

Definition at line 64 of file VisionInterface.h.

6.13.2.11 `const int VisionInterface::MARKER_PGO = 8`

Definition at line 65 of file VisionInterface.h.

6.13.2.12 `const int VisionInterface::MARKER_PGP = 9`

Definition at line 66 of file VisionInterface.h.

6.13.2.13 `const int VisionInterface::MARKER_POG = 10`

Definition at line 67 of file VisionInterface.h.

6.13.2.14 `const int VisionInterface::MARKER_POP = 11`

Definition at line 68 of file VisionInterface.h.

6.13.2.15 `const int VisionInterface::NUM_MARKERS = 12`

Definition at line 78 of file VisionInterface.h.

6.13.2.16 `const int VisionInterface::NUM_VISION_OBJECTS =
NUM_MARKERS + 4`

Definition at line 80 of file VisionInterface.h.

6.13.2.17 `const uchar VisionInterface::OFF_EDGE_BOTTOM = 8`

Definition at line 45 of file VisionInterface.h.

6.13.2.18 `const uchar VisionInterface::OFF_EDGE_LEFT = 1`

Definition at line 42 of file VisionInterface.h.

6.13.2.19 `const uchar VisionInterface::OFF_EDGE_RIGHT = 2`

Definition at line 43 of file VisionInterface.h.

6.13.2.20 `const uchar VisionInterface::OFF_EDGE_TOP = 4`

Definition at line 44 of file VisionInterface.h.

6.13.2.21 `const int VisionInterface::PBALL = 13`

Definition at line 70 of file VisionInterface.h.

6.13.2.22 `const int VisionInterface::RBALL = 12`

Definition at line 69 of file VisionInterface.h.

6.13.2.23 `const int VisionInterface::THING = 15`

Definition at line 72 of file VisionInterface.h.

6.13.2.24 `const double VisionInterface::VertFOV = 48.0 * M_PI / 180.0
[static]`

Definition at line 39 of file VisionInterface.h.

6.14 WM2Kludge Namespace Reference

6.14.1 Detailed Description

Symbols for kludges that alter the behavior of [WorldModel2](#) mapping.

Various assumptions (kludges) you can enable or disable with `enableKludge` and `disableKludge` to make the behavior of the maps appear better

Variables

- `const unsigned int IgnoreZLessThanZero = 1`
Don't incorporate measurements of items situated under the ground plane.
- `const unsigned int IgnoreGreenItems = 2`
Don't incorporate measurements of green items in the map.
- `const unsigned int LazyFastSLAM = 4`
Delay passing of movement information to FastSLAM.

6.14.2 Variable Documentation

6.14.2.1 `const unsigned int WM2Kludge::IgnoreGreenItems = 2`

Don't incorporate measurements of green items in the map.

Our environment has green matting that uniquely indicates that we're looking at the ground. Since we can already guess that the ground is flat, we don't need to waste time on measurements of it.

Definition at line 50 of file `WorldModel2.h`.

6.14.2.2 `const unsigned int WM2Kludge::IgnoreZLessThanZero = 1`

Don't incorporate measurements of items situated under the ground plane.

If your environment has no pits in it, there will be nothing that has a z value smaller than 0. Measurements which have such z values are surely noise and can be ignored.

Definition at line 44 of file `WorldModel2.h`.

6.14.2.3 `const unsigned int WM2Kludge::LazyFastSLAM = 4`

Delay passing of movement information to FastSLAM.

With lots of particles, FastSLAM takes forever to do motion resamples—too long for the robot to remain responsive if it's just doing a course correction while en route. With this kludge enabled, motion resampling will wait until the AIBO is completely stopped; then all the motions will be applied en masse. This will still have an impact on state machine timing, but at least it keeps AIBO from running into a wall. Be warned: you will not know where you are with LazyFastSLAM enabled until you stop moving and FastSLAM has had an opportunity to catch up with itself!

Definition at line 61 of file WorldModel2.h.

Chapter 7

Tekkotsu Class Documentation

7.1 `_afsLandmarkLoc` Struct Reference

```
#include <afsParticle.h>
```

7.1.1 Detailed Description

This structure is used within `afsParticle` structures to encode information about landmark location

Definition at line 33 of file `afsParticle.h`.

Public Types

- enum { `PRIMING`, `PRIMED` }

Public Attributes

- enum `_afsLandmarkLoc:: { ... } state`
- `afsLastObservation priming`
- struct {
 double `x`
 double `y`
} `mean`
- struct {
 double `x`
 double `xy`

```

        double y
    } variance

```

7.1.2 Member Enumeration Documentation

7.1.2.1 anonymous enum

This indicates whether the landmark has been found yet (PRIMED if so, PRIMING if not).

Enumeration values:

PRIMING

PRIMED

Definition at line 36 of file afsParticle.h.

7.1.3 Member Data Documentation

7.1.3.1 struct { ... } [_afsLandmarkLoc::mean](#)

Mean of Gaussian landmark location estimate

7.1.3.2 [afsLastObservation](#) [_afsLandmarkLoc::priming](#)

This element is used when the initial location of the landmark must be triangulated.

Definition at line 40 of file afsParticle.h.

7.1.3.3 enum { ... } [_afsLandmarkLoc::state](#)

This indicates whether the landmark has been found yet (PRIMED if so, PRIMING if not).

7.1.3.4 struct { ... } [_afsLandmarkLoc::variance](#)

Variance of Gaussian landmark location estimate

7.1.3.5 double [_afsLandmarkLoc::x](#)

x value

Definition at line 50 of file afsParticle.h.

7.1.3.6 `double _afsLandmarkLoc::xy`

TSS_TODO.

Definition at line 51 of file `afsParticle.h`.

7.1.3.7 `double _afsLandmarkLoc::y`

y value

Definition at line 52 of file `afsParticle.h`.

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

- [afsParticle.h](#)

7.2 `_afsLastObservation` Struct Reference

```
#include <afsParticle.h>
```

7.2.1 Detailed Description

This structure is used within `afsLandmarkLoc` during the initialization phase, when it is necessary to triangulate the location of the landmark before using regular FastSLAM techniques.

Definition at line 24 of file `afsParticle.h`.

Public Attributes

- `int empty`
TSS.TODO.
- `double x`
x value
- `double y`
y value
- `double theta`
theta value

7.2.2 Member Data Documentation

7.2.2.1 `int _afsLastObservation::empty`

TSS.TODO.

Definition at line 25 of file `afsParticle.h`.

7.2.2.2 `double _afsLastObservation::theta`

theta value

Definition at line 28 of file `afsParticle.h`.

7.2.2.3 `double _afsLastObservation::x`

x value

Definition at line 26 of file `afsParticle.h`.

7.2.2.4 `double _afsLastObservation::y`

y value

Definition at line 27 of file `afsParticle.h`.

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

- [afsParticle.h](#)

7.3 `_afsParticle` Struct Reference

```
#include <afsParticle.h>
```

7.3.1 Detailed Description

This structure contains data for each particle used by the particle filter. Since we have a fixed number of landmarks, we simply fix the number of `afsLandmarkLoc` structures inside.

Definition at line 66 of file `afsParticle.h`.

Public Attributes

- `afsPose` `pose`
- `afsLandmarkLoc` `landmarks` [AFS_NUM_LANDMARKS]
- `int` `gotweight`
- `double` `weight`

7.3.2 Member Data Documentation

7.3.2.1 `int _afsParticle::gotweight`

These two variables reflect the weight assigned to this particle from the last sensor measurement. While the particle is still triangulating, there is no weight calculated and `gotweight` will be false. Otherwise, `gotweight` is true and the new weight is placed into `weight`

Definition at line 79 of file `afsParticle.h`.

7.3.2.2 `afsLandmarkLoc _afsParticle::landmarks`[AFS_NUM_LANDMARKS]

The estimated poses of all landmarks in the map

Definition at line 71 of file `afsParticle.h`.

7.3.2.3 `afsPose _afsParticle::pose`

The robot pose represented by this particle

Definition at line 68 of file `afsParticle.h`.

7.3.2.4 `double _afsParticle::weight`

These two variables reflect the weight assigned to this particle from the last sensor measurement. While the particle is still triangulating, there is no weight calculated and `gotweight` will be false. Otherwise, `gotweight` is true and the new weight is placed into `weight`

Definition at line 80 of file `afsParticle.h`.

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

- [afsParticle.h](#)

7.4 `_afsPose` Struct Reference

```
#include <afsParticle.h>
```

7.4.1 Detailed Description

This structure contains a robot pose. Pretty simple.

Definition at line 15 of file `afsParticle.h`.

Public Attributes

- double `x`
x value
- double `y`
y value
- double `theta`
theta value

7.4.2 Member Data Documentation

7.4.2.1 double `_afsPose::theta`

theta value

Definition at line 18 of file `afsParticle.h`.

7.4.2.2 double `_afsPose::x`

x value

Definition at line 16 of file `afsParticle.h`.

7.4.2.3 double `_afsPose::y`

y value

Definition at line 17 of file `afsParticle.h`.

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

- [afsParticle.h](#)

7.5 `_afsRRP` Struct Reference

```
#include <afsFindBestPose.h>
```

7.5.1 Detailed Description

A structure that contains the relative radial position of a landmark (the robot is at the origin).

Definition at line 18 of file `afsFindBestPose.h`.

Public Attributes

- double `theta`
angle
- double `distance`
distance

7.5.2 Member Data Documentation

7.5.2.1 double `_afsRRP::distance`

`distance`

Definition at line 20 of file `afsFindBestPose.h`.

7.5.2.2 double `_afsRRP::theta`

`angle`

Definition at line 19 of file `afsFindBestPose.h`.

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

- `afsFindBestPose.h`

7.6 `_afsXY` Struct Reference

```
#include <afsFindBestPose.h>
```

7.6.1 Detailed Description

A structure that simply contains the x,y coordinates of landmarks.

Definition at line 24 of file `afsFindBestPose.h`.

Public Attributes

- double `x`
distance along x
- double `y`
distance along y

7.6.2 Member Data Documentation

7.6.2.1 double `_afsXY::x`

distance along x

Definition at line 25 of file `afsFindBestPose.h`.

7.6.2.2 double `_afsXY::y`

distance along y

Definition at line 26 of file `afsFindBestPose.h`.

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

- `afsFindBestPose.h`

7.7 `_dm_cell` Struct Reference

```
#include <almStructures.h>
```

7.7.1 Detailed Description

A cell for the spherical depth map.

Definition at line 25 of file `almStructures.h`.

Public Attributes

- float `depth`
depth in millimeters
- float `confidence`
0 to 1 confidence value in cell data
- `colortype` `color`
the camera.
- int `cluster`
K-means uncertainty clustering. (Internal use only).

7.7.2 Member Data Documentation

7.7.2.1 int `_dm_cell::cluster`

K-means uncertainty clustering. (Internal use only).

Definition at line 30 of file `almStructures.h`.

7.7.2.2 `colortype` `_dm_cell::color`

the camera.

The color of the object in this cell as seen from

Definition at line 28 of file `almStructures.h`.

7.7.2.3 float `_dm_cell::confidence`

0 to 1 confidence value in cell data

Definition at line 27 of file `almStructures.h`.

7.7.2.4 float `_dm_cell::depth`

depth in millimeters

Definition at line 26 of file `almStructures.h`.

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

- `almStructures.h`

7.8 `_FastSLAM_update` Struct Reference

```
#include <WorldModel2.h>
```

7.8.1 Detailed Description

Structure for keeping track of FastSLAM system updates for eventual evaluation. See the documentation on the LazyFastSLAM kludge to see why we need to keep this data around at times.

Definition at line 91 of file WorldModel2.h.

Public Types

- enum { [MOTION](#), [LANDMARK](#) }

Public Attributes

- enum `_FastSLAM_update:: { ... }` [type](#)
the type of record
- double [dx](#)
x distance
- double [dy](#)
y distance
- double [da](#)
angular distance
- long [time](#)
timestamp

7.8.2 Member Enumeration Documentation

7.8.2.1 anonymous enum

Enumeration values:

MOTION

LANDMARK

Definition at line 92 of file WorldModel2.h.

7.8.3 Member Data Documentation

7.8.3.1 `double _FastSLAM_update::da`

angular distance

Definition at line 95 of file `WorldModel2.h`.

7.8.3.2 `double _FastSLAM_update::dx`

x distance

Definition at line 93 of file `WorldModel2.h`.

7.8.3.3 `double _FastSLAM_update::dy`

y distance

Definition at line 94 of file `WorldModel2.h`.

7.8.3.4 `long _FastSLAM_update::time`

timestamp

Definition at line 96 of file `WorldModel2.h`.

7.8.3.5 `enum { ... } _FastSLAM_update::type`

the type of record

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

- [WorldModel2.h](#)

7.9 `_hm_cell` Struct Reference

```
#include <almStructures.h>
```

7.9.1 Detailed Description

A cell for the horizontal height map.

Definition at line 34 of file `almStructures.h`.

Public Attributes

- float `height`
height in millimeters; ground is 0
- float `trav`
0 to 1 estimate of cell traversability (1=yes)
- float `confidence`
0 to 1 confidence value in cell data
- `colortype` `color`
the camera.
- int `cluster`
K-means uncertainty clustering. (Internal use only).

7.9.2 Member Data Documentation

7.9.2.1 int `_hm_cell::cluster`

K-means uncertainty clustering. (Internal use only).

Definition at line 40 of file `almStructures.h`.

7.9.2.2 `colortype` `_hm_cell::color`

the camera.

The color of the object in this cell as seen from

Definition at line 38 of file `almStructures.h`.

7.9.2.3 float `_hm_cell::confidence`

0 to 1 confidence value in cell data

Definition at line 37 of file `almStructures.h`.

7.9.2.4 float `_hm_cell::height`

height in millimeters; ground is 0

Definition at line 35 of file `almStructures.h`.

7.9.2.5 float `_hm_cell::trav`

0 to 1 estimate of cell traversability (1=yes)

Definition at line 36 of file `almStructures.h`.

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

- [almStructures.h](#)

7.10 AGM Class Reference

```
#include <agmMain.h>
```

Static Public Member Functions

- void [init](#) (void)
- void [carryOver](#) (double x, double y, [hm_cell](#) &cell)
- void [decay](#) ()
- void [dump](#) ([hmPicker](#) &p, std::ostream &out)

Static Private Member Functions

- void [genRequests](#) ([MRvector](#) &requests)
- [hm_cell](#) * [getGM](#) ()

Friends

- class [WorldModel2](#)

7.10.1 Member Function Documentation

7.10.1.1 void AGM::carryOver (double x, double y, [hm_cell](#) & cell) [static]

Definition at line 54 of file agmMain.cc.

References [_hm_cell::confidence](#), [GM](#), and [xy2gm_index\(\)](#).

7.10.1.2 void AGM::decay () [static]

Definition at line 67 of file agmMain.cc.

References [ALM_GM_TAX](#), [_hm_cell::confidence](#), [GM](#), and [GM_CELL_COUNT](#).

7.10.1.3 void AGM::dump ([hmPicker](#) & p, std::ostream & out) [static]

Definition at line 147 of file agmMain.cc.

References [AGM_H_SIZE](#), [AGM_V_SIZE](#), and [GM](#).

7.10.1.4 `void AGM::genRequests (MRvector & requests)` [static, private]

Definition at line 76 of file agmMain.cc.

References AGM_BOTTOM, AGM_LEFT, AGM_NUMCLUSTERS, AGM_RIGHT, AGM_TOP, AM_KMEANS_ITERATIONS, [_hm_cell::cluster](#), [_hm_cell::confidence](#), GM, GM_CELL_COUNT, [gm_index2xy\(\)](#), [MotionRequest::GO_TO](#), [MotionRequest::type](#), and [MotionRequest::xy](#).

7.10.1.5 `hm_cell * AGM::getGM ()` [static, private]

Definition at line 160 of file agmMain.cc.

References GM.

7.10.1.6 `void AGM::init (void)` [static]

Definition at line 33 of file agmMain.cc.

References [_hm_cell::cluster](#), [_hm_cell::color](#), COLOR_GREEN, [_hm_cell::confidence](#), GM, GM_CELL_COUNT, [_hm_cell::height](#), and [_hm_cell::trav](#).

7.10.2 Friends And Related Function Documentation

7.10.2.1 `friend class WorldModel2` [friend]

Definition at line 21 of file agmMain.h.

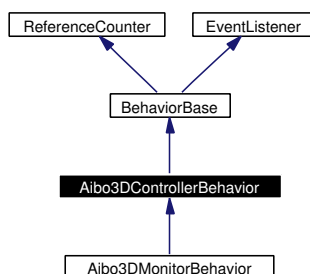
The documentation for this class was generated from the following files:

- [agmMain.h](#)
- [agmMain.cc](#)

7.11 Aibo3DControllerBehavior Class Reference

```
#include <Aibo3DControllerBehavior.h>
```

Inheritance diagram for Aibo3DControllerBehavior:



7.11.1 Detailed Description

Listens to aibo3d control commands coming in from the command port.

Definition at line 23 of file Aibo3DControllerBehavior.h.

Public Member Functions

- [Aibo3DControllerBehavior](#) ()
constructor
- virtual [~Aibo3DControllerBehavior](#) ()
destructor
- int [registerData](#) (char *buf, int bytes)
processes input from the GUI
- virtual void [DoStart](#) ()
By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.
- virtual void [DoStop](#) ()
By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).
- virtual std::string [getGUIType](#) () const

- virtual unsigned int [getPort](#) () const
- virtual std::string [getName](#) () const

returns name of behavior

Static Public Member Functions

- std::string [getClassDescription](#) ()

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- [MotionManager::MC_ID](#) [rcontrol_id](#)

remote controller motion command's id

- [Socket](#) * [cmdsock](#)

The input command stream socket.

- float [val](#) [NumPIDJoints]

the value to use for each of the PID joints

- char * [fbuf](#)

alias to val

- unsigned int [pos](#)

a counter to know when we've gotten 4 frames

Private Member Functions

- [Aibo3DControllerBehavior](#) (const [Aibo3DControllerBehavior](#) &)

don't call

- [Aibo3DControllerBehavior](#) operator= (const [Aibo3DControllerBehavior](#) &)

don't call

7.11.2 Constructor & Destructor Documentation

7.11.2.1 `Aibo3DControllerBehavior::Aibo3DControllerBehavior (const Aibo3DControllerBehavior &) [private]`

don't call

7.11.2.2 `Aibo3DControllerBehavior::Aibo3DControllerBehavior () [inline]`

constructor

Definition at line 40 of file `Aibo3DControllerBehavior.h`.

References `aibo3dControllerBehavior`, `cmdsock`, `fbuf`, `pos`, `rcontrol_id`, `Socket::SOCK_STREAM`, `val`, and `wireless`.

7.11.2.3 `virtual Aibo3DControllerBehavior::~~Aibo3DControllerBehavior () [inline, virtual]`

destructor

Definition at line 47 of file `Aibo3DControllerBehavior.h`.

References `aibo3dControllerBehavior`.

7.11.3 Member Function Documentation

7.11.3.1 `virtual void Aibo3DControllerBehavior::DoStart () [inline, virtual]`

By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.

Reimplemented from [BehaviorBase](#).

Definition at line 69 of file `Aibo3DControllerBehavior.h`.

References `MotionManager::addMotion()`, `Config::main_config::aibo3d_port`, `aibo3dcontrollercmd_callback()`, `cmdsock`, `config`, `BehaviorBase::DoStart()`, `getGUIType()`, `getPort()`, `Wireless::listen()`, `Controller::loadGUI()`, `Config::main`, `motman`, `rcontrol_id`, `Wireless::setReceiver()`, `Socket::sock`, and `wireless`.

7.11.3.2 `virtual void Aibo3DControllerBehavior::DoStop ()` [`inline`, `virtual`]

By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your `DoStop()`, not beginning (it might delete this).

Reimplemented from [BehaviorBase](#).

Definition at line 86 of file `Aibo3DControllerBehavior.h`.

References `Wireless::close()`, `Controller::closeGUI()`, `cmdsock`, `BehaviorBase::DoStop()`, `getGUIType()`, `motman`, `rcontrol_id`, `MotionManager::removeMotion()`, and `wireless`.

7.11.3.3 `std::string Aibo3DControllerBehavior::getClassDescription ()` [`inline`, `static`]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Reimplemented in [Aibo3DMonitorBehavior](#).

Definition at line 100 of file `Aibo3DControllerBehavior.h`.

References `Config::main_config::aibo3d_port`, `config`, and `Config::main`.

7.11.3.4 `virtual std::string Aibo3DControllerBehavior::getGUIType () const` [`inline`, `virtual`]

Reimplemented in [Aibo3DMonitorBehavior](#).

Definition at line 96 of file `Aibo3DControllerBehavior.h`.

7.11.3.5 `virtual std::string Aibo3DControllerBehavior::getName () const` [`inline`, `virtual`]

returns name of behavior

Implements [BehaviorBase](#).

Reimplemented in [Aibo3DMonitorBehavior](#).

Definition at line 99 of file `Aibo3DControllerBehavior.h`.

7.11.3.6 `virtual unsigned int Aibo3DControllerBehavior::getPort () const`
[inline, virtual]

Reimplemented in [Aibo3DMonitorBehavior](#).

Definition at line 97 of file `Aibo3DControllerBehavior.h`.

References `Config::main_config::aibo3d_port`, `config`, and `Config::main`.

7.11.3.7 [Aibo3DControllerBehavior](#) `Aibo3DControllerBehavior::operator=`
(const [Aibo3DControllerBehavior](#) &) [private]

don't call

7.11.3.8 `int Aibo3DControllerBehavior::registerData (char * buf, int bytes)`
[inline]

processes input from the GUI

Definition at line 50 of file `Aibo3DControllerBehavior.h`.

References `MotionManager::checkinMotion()`, `MotionManager::checkoutMotion()`, `RemoteControllerMC::cmds`, `fbuf`, `motman`, `ERS210Info::NumPIDJoints`, `pos`, `rcontrol_id`, `RemoteControllerMC::setDirty()`, and `val`.

7.11.4 Member Data Documentation

7.11.4.1 [Socket*](#) [Aibo3DControllerBehavior::cmdsock](#) [protected]

The input command stream socket.

Definition at line 28 of file `Aibo3DControllerBehavior.h`.

7.11.4.2 `char*` [Aibo3DControllerBehavior::fbuf](#) [protected]

alias to `val`

Definition at line 31 of file `Aibo3DControllerBehavior.h`.

7.11.4.3 `unsigned int` [Aibo3DControllerBehavior::pos](#) [protected]

a counter to know when we've gotten 4 frames

Definition at line 32 of file `Aibo3DControllerBehavior.h`.

7.11.4.4 [MotionManager::MC_ID Aibo3DControllerBehavior::rcontrol_id](#) [protected]

remote controller motion command's id

Definition at line 25 of file Aibo3DControllerBehavior.h.

7.11.4.5 **float** [Aibo3DControllerBehavior::val](#)[NumPIDJoints] [protected]

the value to use for each of the PID joints

Definition at line 30 of file Aibo3DControllerBehavior.h.

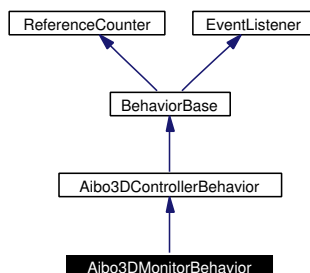
The documentation for this class was generated from the following file:

- [Aibo3DControllerBehavior.h](#)

7.12 Aibo3DMonitorBehavior Class Reference

```
#include <Aibo3DMonitorBehavior.h>
```

Inheritance diagram for Aibo3DMonitorBehavior:



7.12.1 Detailed Description

Sends current pose to Aibo3D GUI, ignores incoming commands.

Definition at line 6 of file Aibo3DMonitorBehavior.h.

Public Member Functions

- [Aibo3DMonitorBehavior](#) ()
- virtual std::string [getGUIType](#) () const
- virtual unsigned int [getPort](#) () const
- virtual std::string [getName](#) () const

returns name of behavior

Static Public Member Functions

- std::string [getClassDescription](#) ()

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

7.12.2 Constructor & Destructor Documentation

7.12.2.1 Aibo3DMonitorBehavior::Aibo3DMonitorBehavior () [inline]

Definition at line 8 of file Aibo3DMonitorBehavior.h.

7.12.3 Member Function Documentation

7.12.3.1 std::string Aibo3DMonitorBehavior::getClassDescription () [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [Aibo3DControllerBehavior](#).

Definition at line 14 of file Aibo3DMonitorBehavior.h.

References `Config::main_config::aibo3d_port`, `config`, and `Config::main`.

7.12.3.2 virtual std::string Aibo3DMonitorBehavior::getGUIType () const [inline, virtual]

Reimplemented from [Aibo3DControllerBehavior](#).

Definition at line 10 of file Aibo3DMonitorBehavior.h.

7.12.3.3 virtual std::string Aibo3DMonitorBehavior::getName () const [inline, virtual]

returns name of behavior

Reimplemented from [Aibo3DControllerBehavior](#).

Definition at line 13 of file Aibo3DMonitorBehavior.h.

7.12.3.4 virtual unsigned int Aibo3DMonitorBehavior::getPort () const [inline, virtual]

Reimplemented from [Aibo3DControllerBehavior](#).

Definition at line 11 of file Aibo3DMonitorBehavior.h.

References `Config::main_config::aibo3d_port`, `config`, and `Config::main`.

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

- [Aibo3DMonitorBehavior.h](#)

7.13 ALM Class Reference

```
#include <almMain.h>
```

Static Public Member Functions

- void [init](#) (void)
- void [move](#) (double [dx](#), double dy, double [da](#), unsigned int [time](#))
- void [registerDepth](#) (double depth, double tilt, double pan)
- void [registerDepth](#) (double depth, double tilt, double pan, unsigned int kludges)
- void [registerGround](#) ()
- void [stampHM](#) ([afsPose](#) &pose)
- void [dumpDM](#) ([dmPicker](#) &p, std::ostream &out)
- void [dumpHM](#) ([hmPicker](#) &p, std::ostream &out)

Static Private Member Functions

- void [nukeAndPaveCurrentMap](#) (void)
- void [genRequests](#) ([MRvector](#) &requests)
- [dm_cell](#) * [getDM](#) ()
- [hm_cell](#) * [getHM](#) ()

Friends

- class [WorldModel2](#)

7.13.1 Member Function Documentation

7.13.1.1 void [ALM::dumpDM](#) ([dmPicker](#) &*p*, std::ostream &*out*) [static]

Definition at line 680 of file [almMain.cc](#).

References [ALM_DM_H_SIZE](#), [ALM_DM_V_SIZE](#), and [DM](#).

7.13.1.2 void [ALM::dumpHM](#) ([hmPicker](#) &*p*, std::ostream &*out*) [static]

Definition at line 690 of file [almMain.cc](#).

References [ALM_HM_SIZE](#), and [HM](#).

7.13.1.3 `void ALM::genRequests (MRvector & requests)` [static, private]

Definition at line 520 of file almMain.cc.

References ALM_DM_BOTTOM, ALM_DM_LEFT, ALM_DM_NUMCLUSTERS, ALM_DM_RIGHT, ALM_DM_TOP, ALM_HM_NUMCLUSTERS, ALM_HM_RADIUS, ALM_HM_SIZE, AM_KMEANS_ITERATIONS, MotionRequest::azalt, _hm_cell::cluster, _dm_cell::cluster, _hm_cell::confidence, _dm_cell::confidence, DM, DM_CELL_COUNT, dm_index2angles(), HM, HM_CELL_COUNT, hm_index2xy(), MotionRequest::LOOK_AT, MotionRequest::LOOK_DOWN_AT, MotionRequest::type, and MotionRequest::xy.

7.13.1.4 `dm_cell * ALM::getDM ()` [static, private]

Definition at line 704 of file almMain.cc.

References DM.

7.13.1.5 `hm_cell * ALM::getHM ()` [static, private]

Definition at line 705 of file almMain.cc.

References HM.

7.13.1.6 `void ALM::init (void)` [static]

Definition at line 60 of file almMain.cc.

References DM, DMs, HM, HMs, and nukeAndPaveCurrentMap().

7.13.1.7 `void ALM::move (double dx, double dy, double da, unsigned int time)` [static]

Definition at line 76 of file almMain.cc.

References ALM_DM_TAX, ALM_HM_TAX, angles2dm_index(), _dm_cell::confidence, _hm_cell::confidence, da, _dm_cell::depth, DM, DM_CELL_COUNT, dm_index2angles(), DMs, dx, HM, HM_CELL_COUNT, hm_index2xy(), HMs, IROORDIST, neck_range2xyz(), nukeAndPaveCurrentMap(), time, WalkMotionModel(), xy2hm_index(), and xyz2neck_range().

7.13.1.8 `void ALM::nukeAndPaveCurrentMap (void)` [static, private]

Definition at line 205 of file almMain.cc.

References AIBO_TILT_PIVOT_HEIGHT, `_hm_cell::color`, `_dm_cell::color`, COLOR_BLUE, COLOR_GREEN, `_hm_cell::confidence`, `_dm_cell::confidence`, `_dm_cell::depth`, DM, DM_CELL_COUNT, `dm_index2angles()`, `_hm_cell::height`, HM, HM_CELL_COUNT, IROORDIST, and `_hm_cell::trav`.

7.13.1.9 void ALM::registerDepth (double *depth*, double *tilt*, double *pan*, unsigned int *kludges*) [static]

Definition at line 267 of file `almMain.cc`.

References AIBO_IR_CAL_MULTIPLIER, AIBO_IR_CAL_OFFSET, AIBO_MAX_BUMP, AIBO_MIN_CLEARANCE, ALM_DM_H_SIZE, ALM_IR_SPLAT_STDDEV, `angles2dm_index()`, `_hm_cell::color`, `_dm_cell::color`, COLOR_GREEN, `colortype`, `_hm_cell::confidence`, `_dm_cell::confidence`, `_dm_cell::depth`, DM, DM_CELL_COUNT, `dm_index2angles()`, `dx`, `head_range2xyz()`, `_hm_cell::height`, `Vision::height`, HM, IROORDIST, `neck_range2xyz()`, `Vision::num_runs`, `Vision::rmap`, SQRT_2_PI, `_hm_cell::trav`, `vision`, `Vision::width`, `xy2hm_index()`, and `xyz2neck_range()`.

7.13.1.10 void ALM::registerDepth (double *depth*, double *tilt*, double *pan*) [static]

Definition at line 259 of file `almMain.cc`.

7.13.1.11 void ALM::registerGround () [static]

Definition at line 404 of file `almMain.cc`.

References AIBO_CAM_H_FOV, AIBO_CAM_V_FOV, AIBO_NECK_HEIGHT, AIBO_TILT_PIVOT_HEIGHT, ALM_GPA_CONFIDENCE, `angles2dm_index()`, `_dm_cell::color`, `_hm_cell::color`, COLOR_GREEN, `_dm_cell::confidence`, `_hm_cell::confidence`, `_dm_cell::depth`, DM, `dx`, `_hm_cell::height`, `Vision::height`, HM, `Vision::num_runs`, `Vision::rmap`, `_hm_cell::trav`, `vision`, `Vision::width`, `xy2hm_index()`, and `xyz2neck_range()`.

7.13.1.12 void ALM::stampHM (afspose &pose) [static]

Definition at line 488 of file `almMain.cc`.

References ALM_HM_RADIUS, AGM::carryOver(), HM, HM_CELL_COUNT, `hm_index2xy()`, `_afspose::theta`, `_afspose::x`, and `_afspose::y`.

7.13.2 Friends And Related Function Documentation

7.13.2.1 friend class [WorldModel2](#) [`friend`]

Definition at line 24 of file `almMain.h`.

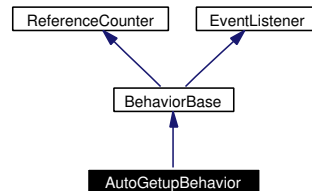
The documentation for this class was generated from the following files:

- [almMain.h](#)
- [almMain.cc](#)

7.14 AutoGetupBehavior Class Reference

```
#include <AutoGetupBehavior.h>
```

Inheritance diagram for AutoGetupBehavior:



7.14.1 Detailed Description

a little background behavior to keep the robot on its feet

Definition at line 15 of file AutoGetupBehavior.h.

Public Member Functions

- [AutoGetupBehavior](#) ()
constructor
- virtual [~AutoGetupBehavior](#) ()
destructor
- virtual void [DoStart](#) ()
Listens for the [SensorSourceID::UpdatedSID](#).
- virtual void [DoStop](#) ()
Stops listening for events.
- virtual void [processEvent](#) (const [EventBase](#) &event)
Run appropriate motion script if the robot falls over.
- virtual std::string [getName](#) () const
Identifies the behavior in menus and such.

Static Public Member Functions

- std::string [getClassDescription](#) ()

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- float [back](#)

exponential average of backwards accel

- float [side](#)

exponential average of sideways accel

- float [gamma](#)

default 0.9, gamma parameter for exponential average of above

- float [sensitivity](#)

*default 0.85*0.85, squared threshold to consider having fallen over, use values 0-1*

- bool [waiting](#)

true while we're waiting to hear from completion of [MotionSequence](#), won't try again until this is cleared

7.14.2 Constructor & Destructor Documentation

7.14.2.1 `AutoGetupBehavior::AutoGetupBehavior () [inline]`

constructor

Definition at line 18 of file AutoGetupBehavior.h.

References [back](#), [gamma](#), [sensitivity](#), [side](#), and [waiting](#).

7.14.2.2 `virtual AutoGetupBehavior::~~AutoGetupBehavior () [inline, virtual]`

destructor

Definition at line 20 of file AutoGetupBehavior.h.

7.14.3 Member Function Documentation

7.14.3.1 `virtual void AutoGetupBehavior::DoStart () [inline, virtual]`

Listens for the [SensorSourceID::UpdatedSID](#).

Reimplemented from [BehaviorBase](#).

Definition at line 23 of file AutoGetupBehavior.h.

References [EventRouter::addListener\(\)](#), [BehaviorBase::DoStart\(\)](#), [erouter](#), and [EventBase::sensorEGID](#).

7.14.3.2 `virtual void AutoGetupBehavior::DoStop () [inline, virtual]`

Stops listening for events.

Reimplemented from [BehaviorBase](#).

Definition at line 28 of file AutoGetupBehavior.h.

References [BehaviorBase::DoStop\(\)](#), [erouter](#), and [EventRouter::forgetListener\(\)](#).

7.14.3.3 `std::string AutoGetupBehavior::getClassDescription () [inline, static]`

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 62 of file AutoGetupBehavior.h.

7.14.3.4 `virtual std::string AutoGetupBehavior::getName () const [inline, virtual]`

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 61 of file AutoGetupBehavior.h.

7.14.3.5 `virtual void AutoGetupBehavior::processEvent (const EventBase & event) [inline, virtual]`

Run appropriate motion script if the robot falls over.

Reimplemented from [BehaviorBase](#).

Definition at line 33 of file AutoGetupBehavior.h.

References [EventRouter::addListener\(\)](#), [MotionManager::addMotion\(\)](#), [ERS210Info::BAccelOffset](#), [back](#), [config](#), [EventBase::deactivateETID](#), [erouter](#), [WorldState::g](#), [gamma](#), [EventBase::getGeneratorID\(\)](#), [MotionManager::kHigh-Priority](#), [ERS210Info::LAccelOffset](#), [Config::motion_config::makePath\(\)](#), [MotionManager::MC_ID](#), [Config::motion](#), [motman](#), [EventBase::motmanEGID](#), [SoundManager::PlayFile\(\)](#), [EventRouter::removeListener\(\)](#), [sensitivity](#), [WorldState::sensors](#), [side](#), [sndman](#), [state](#), and [waiting](#).

7.14.4 Member Data Documentation

7.14.4.1 float [AutoGetupBehavior::back](#) [protected]

exponential average of backwards accel

Definition at line 65 of file AutoGetupBehavior.h.

7.14.4.2 float [AutoGetupBehavior::gamma](#) [protected]

default 0.9, gamma parameter for exponential average of above

Definition at line 67 of file AutoGetupBehavior.h.

7.14.4.3 float [AutoGetupBehavior::sensitivity](#) [protected]

default 0.85*0.85, squared threshold to consider having fallen over, use values 0-1

Definition at line 68 of file AutoGetupBehavior.h.

7.14.4.4 float [AutoGetupBehavior::side](#) [protected]

exponential average of sideways accel

Definition at line 66 of file AutoGetupBehavior.h.

7.14.4.5 bool [AutoGetupBehavior::waiting](#) [protected]

true while we're waiting to hear from completion of [MotionSequence](#), won't try again until this is cleared

Definition at line 69 of file AutoGetupBehavior.h.

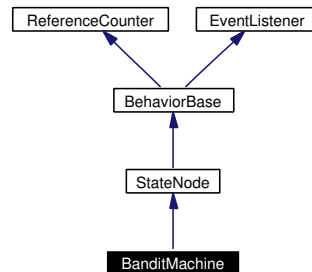
The documentation for this class was generated from the following file:

- [AutoGetupBehavior.h](#)

7.15 BanditMachine Class Reference

```
#include <BanditMachine.h>
```

Inheritance diagram for BanditMachine:



7.15.1 Detailed Description

Plays K-armed bandit.

Definition at line 19 of file BanditMachine.h.

Public Member Functions

- [BanditMachine](#) ()
constructor
- [BanditMachine](#) (const char *n, [StateNode](#) *p=NULL)
constructor
- virtual [~BanditMachine](#) ()
destructor
- virtual void [setup](#) ()
This is called by [DoStart\(\)](#) when you should setup the network.
- virtual void [DoStart](#) ()
Transitions should call this when you are entering the state, so it can enable its transitions.
- virtual void [DoStop](#) ()
Transitions should call this when you are leaving the state, so it can disable its transitions.

Static Public Member Functions

- `std::string getClassDescription ()`
Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- `StareAtBallBehavior * stare`
active as long as we're in this state so it keeps an eye on the ball
- `StateNode * start`
used to start off by lying down before we start pressing buttons
- `MotionManager::MC_ID liedown`
a [MotionSequence](#) which will move the dog into a lying down posture
- `karmedbanditExp3_1 bandit`
algorithm to use in the k-armed bandit problem

Private Member Functions

- `BanditMachine (const BanditMachine &node)`
don't call this
- `BanditMachine operator= (const BanditMachine &node)`
don't call this

7.15.2 Constructor & Destructor Documentation

7.15.2.1 `BanditMachine::BanditMachine \(\) [inline]`

constructor

Definition at line 22 of file `BanditMachine.h`.

References `ReferenceCounter::AddReference()`, `bandit`, `liedown`, `stare`, and `start`.

7.15.2.2 **BanditMachine::BanditMachine** (const char * *n*, [StateNode](#) * *p* = NULL) [inline]

constructor

Definition at line 29 of file BanditMachine.h.

References [ReferenceCounter::AddReference\(\)](#), [bandit](#), [liedown](#), [stare](#), and [start](#).

7.15.2.3 **virtual BanditMachine::~~BanditMachine** () [inline, virtual]

destructor

Definition at line 36 of file BanditMachine.h.

References [ReferenceCounter::RemoveReference\(\)](#), and [stare](#).

7.15.2.4 **BanditMachine::BanditMachine** (const [BanditMachine](#) & *node*) [private]

don't call this

7.15.3 Member Function Documentation

7.15.3.1 **virtual void BanditMachine::DoStart** () [inline, virtual]

Transitions should call this when you are entering the state, so it can enable its transitions.

Reimplemented from [StateNode](#).

Definition at line 58 of file BanditMachine.h.

References [MotionManager::addMotion\(\)](#), [StareAtBallBehavior::DoStart\(\)](#), [StateNode::DoStart\(\)](#), [ERS210Info::KneeOffset](#), [ERS210Info::LFrLegOffset](#), [liedown](#), [motman](#), [ERS210Info::RFrLegOffset](#), [ERS210Info::RotatorOffset](#), [stare](#), and [start](#).

7.15.3.2 **virtual void BanditMachine::DoStop** () [inline, virtual]

Transitions should call this when you are leaving the state, so it can disable its transitions.

Reimplemented from [StateNode](#).

Definition at line 70 of file BanditMachine.h.

References [StateNode::DoStop\(\)](#), [StareAtBallBehavior::DoStop\(\)](#), [liedown](#), [motman](#), [MotionManager::removeMotion\(\)](#), and [stare](#).

7.15.3.3 `std::string BanditMachine::getClassDescription ()` [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 40 of file `BanditMachine.h`.

7.15.3.4 `BanditMachine BanditMachine::operator= (const BanditMachine & node)` [private]

don't call this

7.15.3.5 `virtual void BanditMachine::setup ()` [inline, virtual]

This is called by [DoStart\(\)](#) when you should setup the network.

Reimplemented from [StateNode](#).

Definition at line 42 of file `BanditMachine.h`.

References `StateNode::addNode()`, `StateNode::addTransition()`, `bandit`, `ERS210Info::KneeOffset`, `ERS210Info::LFrLegOffset`, `WorldState::pidduties`, `ERS210Info::RFrLegOffset`, `ERS210Info::RotatorOffset`, `EventBase::sensorEGID`, `StateNode::setup()`, `start`, `state`, and `EventBase::statusETID`.

7.15.4 Member Data Documentation

7.15.4.1 `karmedbanditExp3.1 BanditMachine::bandit` [protected]

algorithm to use in the k-armed bandit problem

Definition at line 204 of file `BanditMachine.h`.

7.15.4.2 `MotionManager::MC_ID BanditMachine::liedown` [protected]

a [MotionSequence](#) which will move the dog into a lying down posture

Definition at line 203 of file `BanditMachine.h`.

7.15.4.3 `StareAtBallBehavior* BanditMachine::stare` [protected]

active as long as we're in this state so it keeps an eye on the ball

Definition at line 201 of file BanditMachine.h.

7.15.4.4 `StateNode*` `BanditMachine::start` [protected]

used to start off by lying down before we start pressing buttons

Definition at line 202 of file BanditMachine.h.

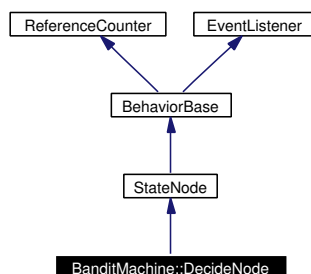
The documentation for this class was generated from the following file:

- [BanditMachine.h](#)

7.16 BanditMachine::DecideNode Class Reference

```
#include <BanditMachine.h>
```

Inheritance diagram for BanditMachine::DecideNode:



7.16.1 Detailed Description

uses one of the algorithms in [karmedbandit.h](#) to decide which paw to press next

Definition at line 121 of file BanditMachine.h.

Public Member Functions

- [DecideNode](#) (const char *n, [StateNode](#) *p, [karmedbanditExp3_1](#) &bandito, [StateNode](#) *left, [StateNode](#) *right)
constructor
- virtual void [DoStart](#) ()
Transitions should call this when you are entering the state, so it can enable its transitions.

Protected Attributes

- [karmedbanditExp3_1](#) & b
the class implementing the k-armed bandit algorithm
- [StateNode](#) * l
the node to go to if the left paw is chosen
- [StateNode](#) * r

the node to go to if the right paw is chosen

Private Member Functions

- [DecideNode](#) (const [DecideNode](#) &node)
don't call this
- [DecideNode operator=](#) (const [DecideNode](#) &node)
don't call this

7.16.2 Constructor & Destructor Documentation

7.16.2.1 [BanditMachine::DecideNode::DecideNode](#) (const char * *n*, [StateNode](#) * *p*, [karmedbanditExp3_1](#) & *bandito*, [StateNode](#) * *left*, [StateNode](#) * *right*) [inline]

constructor

Parameters:

- n* name of the node
- p* the parent node
- bandito* the decision making algorithm to use (look in [karmedbandit.h](#))
- left* the [PressNode](#) to go to if the left paw is chosen
- right* the [PressNode](#) to go to if the right paw is chosen

Definition at line 130 of file [BanditMachine.h](#).

References [b](#), [l](#), and [r](#).

7.16.2.2 [BanditMachine::DecideNode::DecideNode](#) (const [DecideNode](#) & *node*) [private]

don't call this

7.16.3 Member Function Documentation

7.16.3.1 [virtual void BanditMachine::DecideNode::DoStart](#) () [inline, virtual]

Transitions should call this when you are entering the state, so it can enable its transitions.

Reimplemented from [StateNode](#).

Definition at line 133 of file BanditMachine.h.

References [b](#), [karmedbanditExp3_1::decide\(\)](#), [StateNode::DoStart\(\)](#), [StateNode::DoStop\(\)](#), [l](#), and [r](#).

7.16.3.2 [DecideNode](#) [BanditMachine::DecideNode::operator= \(const \[DecideNode\]\(#\) & node\)](#) [private]

don't call this

7.16.4 Member Data Documentation

7.16.4.1 [karmedbanditExp3_1& BanditMachine::DecideNode::b](#) [protected]

the class implementing the k-armed bandit algorithm

Definition at line 145 of file BanditMachine.h.

7.16.4.2 [StateNode* BanditMachine::DecideNode::l](#) [protected]

the node to go to if the left paw is chosen

Definition at line 146 of file BanditMachine.h.

7.16.4.3 [StateNode* BanditMachine::DecideNode::r](#) [protected]

the node to go to if the right paw is chosen

Definition at line 147 of file BanditMachine.h.

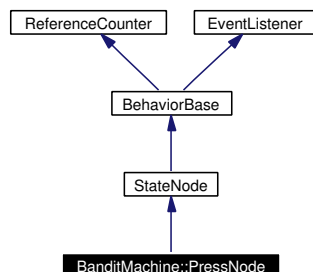
The documentation for this class was generated from the following file:

- [BanditMachine.h](#)

7.17 BanditMachine::PressNode Class Reference

```
#include <BanditMachine.h>
```

Inheritance diagram for BanditMachine::PressNode:



7.17.1 Detailed Description

This node is used to move a paw down using a [MotionSequenceMC](#).

Definition at line 78 of file `BanditMachine.h`.

Public Member Functions

- [PressNode](#) (const char *n, [StateNode](#) *p, unsigned int idx)
constructor
- virtual [~PressNode](#) ()
destructor
- virtual void [DoStart](#) ()
Transitions should call this when you are entering the state, so it can enable its transitions.
- virtual void [DoStop](#) ()
Transitions should call this when you are leaving the state, so it can disable its transitions.

Protected Attributes

- [MotionManager::MC_ID](#) `press_id`

the MC_ID of the [MotionSequenceMC](#) being used to do the press

- unsigned int [index](#)
the joint index of the paw to move

7.17.2 Constructor & Destructor Documentation

7.17.2.1 **BanditMachine::PressNode::PressNode** (const char * *n*, [StateNode](#) * *p*, unsigned int *idx*) [inline]

constructor

Parameters:

- n* name of the node
- p* the parent node
- idx* the joint index of the paw to move

Definition at line 85 of file BanditMachine.h.

References [MotionManager::addMotion\(\)](#), [index](#), [MotionManager::kStdPriority](#), [ERS210Info::MinRange](#), [motman](#), [ERS210Info::outputRanges](#), and [press_id](#).

7.17.2.2 **virtual BanditMachine::PressNode::~PressNode** () [inline, virtual]

destructor

Definition at line 98 of file BanditMachine.h.

References [motman](#), [press_id](#), and [MotionManager::removeMotion\(\)](#).

7.17.3 Member Function Documentation

7.17.3.1 **virtual void BanditMachine::PressNode::DoStart** () [inline, virtual]

Transitions should call this when you are entering the state, so it can enable its transitions.

Reimplemented from [StateNode](#).

Definition at line 101 of file BanditMachine.h.

References [StateNode::DoStart\(\)](#), [index](#), and [press_id](#).

7.17.3.2 **virtual void BanditMachine::PressNode::DoStop ()** [inline, virtual]

Transitions should call this when you are leaving the state, so it can disable its transitions.

Reimplemented from [StateNode](#).

Definition at line 108 of file BanditMachine.h.

References [StateNode::DoStop\(\)](#), and [press_id](#).

7.17.4 Member Data Documentation

7.17.4.1 **unsigned int BanditMachine::PressNode::index** [protected]

the joint index of the paw to move

Definition at line 117 of file BanditMachine.h.

7.17.4.2 **MotionManager::MC_ID BanditMachine::PressNode::press_id** [protected]

the MC_ID of the [MotionSequenceMC](#) being used to do the press

Definition at line 116 of file BanditMachine.h.

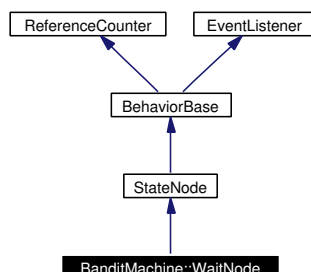
The documentation for this class was generated from the following file:

- [BanditMachine.h](#)

7.18 BanditMachine::WaitNode Class Reference

```
#include <BanditMachine.h>
```

Inheritance diagram for BanditMachine::WaitNode:



7.18.1 Detailed Description

Waits to see if a reward is received, lights up LEDs to let the user know.

Definition at line 154 of file BanditMachine.h.

Public Member Functions

- [WaitNode](#) (const char *n, [StateNode](#) *p, [karmedbanditExp3_1](#) &bandito)
constructor
- virtual [~WaitNode](#) ()
destructor
- virtual void [DoStart](#) ()
Transitions should call this when you are entering the state, so it can enable its transitions.
- virtual void [DoStop](#) ()
Transitions should call this when you are leaving the state, so it can disable its transitions.
- virtual void [processEvent](#) (const [EventBase](#) &event)
Doesn't do anything, supplied here so you don't have to (since events will probably be going to Transition's, not here).

Protected Attributes

- [karmedbanditExp3_1](#) & [b](#)
the class implimenting a k-armed bandit algorithm to pass the reward back to
- bool [reward](#)
true if a reward was received
- [MotionManager::MC_ID](#) [leds_id](#)
MC_ID of a [LedMC](#).

7.18.2 Constructor & Destructor Documentation

7.18.2.1 **BanditMachine::WaitNode::WaitNode** (const char * *n*, [StateNode](#) * *p*, [karmedbanditExp3_1](#) & *bandito*) [inline]

constructor

Definition at line 161 of file [BanditMachine.h](#).

References [MotionManager::addMotion\(\)](#), [b](#), [leds_id](#), [motman](#), and [reward](#).

7.18.2.2 **virtual BanditMachine::WaitNode::~~WaitNode** () [inline, virtual]

destructor

Definition at line 167 of file [BanditMachine.h](#).

References [leds_id](#), [motman](#), and [MotionManager::removeMotion\(\)](#).

7.18.3 Member Function Documentation

7.18.3.1 **virtual void BanditMachine::WaitNode::DoStart** () [inline, virtual]

Transitions should call this when you are entering the state, so it can enable its transitions.

Reimplemented from [StateNode](#).

Definition at line 170 of file [BanditMachine.h](#).

References [EventRouter::addListener\(\)](#), [EventRouter::addTimer\(\)](#), [ERS210Info::BotLLEDMask](#), [ERS210Info::BotRLEDMask](#), [StateNode::DoStart\(\)](#), [erouter](#), [leds_id](#), and [EventBase::visionEGID](#).

7.18.3.2 **virtual void BanditMachine::WaitNode::DoStop ()** [inline, virtual]

Transitions should call this when you are leaving the state, so it can disable its transitions.

Reimplemented from [StateNode](#).

Definition at line 177 of file BanditMachine.h.

References [b](#), [StateNode::DoStop\(\)](#), [erouter](#), [EventRouter::forgetListener\(\)](#), [reward](#), and [karmedbanditExp3_1::reward\(\)](#).

7.18.3.3 **virtual void BanditMachine::WaitNode::processEvent (const [EventBase](#) & event)** [inline, virtual]

Doesn't do anything, supplied here so you don't have to (since events will probably be going to Transition's, not here).

Reimplemented from [StateNode](#).

Definition at line 184 of file BanditMachine.h.

References [erouter](#), [EventRouter::forgetListener\(\)](#), [EventBase::getGeneratorID\(\)](#), [leds_id](#), [ERS210Info::MidLLEDMask](#), [ERS210Info::MidRLEDMask](#), [SoundManager::PlayFile\(\)](#), [reward](#), [sndman](#), and [EventBase::timerEGID](#).

7.18.4 Member Data Documentation

7.18.4.1 **[karmedbanditExp3_1](#) & [BanditMachine::WaitNode::b](#)** [protected]

the class implimenting a k-armed bandit algorithm to pass the reward back to

Definition at line 196 of file BanditMachine.h.

7.18.4.2 **[MotionManager::MC_ID](#) [BanditMachine::WaitNode::leds_id](#)** [protected]

MC_ID of a [LedMC](#).

Definition at line 198 of file BanditMachine.h.

7.18.4.3 **bool [BanditMachine::WaitNode::reward](#)** [protected]

true if a reward was received

Definition at line 197 of file BanditMachine.h.

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

- [BanditMachine.h](#)

7.19 `basic_iNetStream< charT, traits >` Class Template Reference

```
#include <ionetstream.h>
```

Inheritance diagram for `basic_iNetStream< charT, traits >`:



```
template<class charT, class traits> class basic_iNetStream< charT, traits >
```

Public Types

- typedef `charT` [char_type](#)
- typedef `traits::int_type` [int_type](#)
- typedef `traits::pos_type` [pos_type](#)
- typedef `traits::off_type` [off_type](#)
- typedef `traits` [traits_type](#)

Public Member Functions

- [basic_iNetStream](#) ()
- [basic_iNetStream](#) (const `IPAddr::ipnum_t` &host, const `IPAddr::ipport_t` port)
- [basic_iNetStream](#) (const `IPAddr::const_ipname_t` &host, const `IPAddr::ipport_t` port)
- [basic_iNetStream](#) (const `IPAddr` &addr)
- [basic_iNetStream](#) (size_t buf_in_size, size_t buf_out_size)
- [basic_netbuf](#)< `charT`, `traits` > * [rdbuf](#) () const
- bool [open](#) (const `IPAddr` &addr)
- bool [open](#) (const `IPAddr::const_ipname_t` &str)
- bool [open](#) (const `IPAddr::ipnum_t` &addr, const `IPAddr::ipport_t` &aPort)
- bool [open](#) (const `IPAddr::const_ipname_t` &ahost, unsigned int aPort)
- bool [is_open](#) () const
- void [close](#) ()
- void [setEcho](#) (bool val=true)
- bool [getEcho](#) ()

Private Attributes

- [basic_netbuf](#)< charT, traits > [nb](#)

7.19.1 Member Typedef Documentation

7.19.1.1 `template<class charT, class traits> typedef charT basic_iNetStream< charT, traits >::char_type`

Definition at line 103 of file ionetstream.h.

7.19.1.2 `template<class charT, class traits> typedef traits::int_type basic_iNetStream< charT, traits >::int_type`

Definition at line 104 of file ionetstream.h.

7.19.1.3 `template<class charT, class traits> typedef traits::off_type basic_iNetStream< charT, traits >::off_type`

Definition at line 106 of file ionetstream.h.

7.19.1.4 `template<class charT, class traits> typedef traits::pos_type basic_iNetStream< charT, traits >::pos_type`

Definition at line 105 of file ionetstream.h.

7.19.1.5 `template<class charT, class traits> typedef traits basic_iNetStream< charT, traits >::traits_type`

Definition at line 107 of file ionetstream.h.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 `template<class charT, class traits> basic_iNetStream< charT, traits >::basic_iNetStream () [inline]`

Definition at line 109 of file ionetstream.h.

References [basic_iNetStream](#)< charT, traits >::nb.

7.19.2.2 `template<class charT, class traits> basic_iNetStream< charT, traits >::basic_iNetStream (const IPAddr::ipnum_t & host, const IPAddr::ipport_t port) [inline]`

Definition at line 110 of file ionetstream.h.

References `basic_iNetStream< charT, traits >::nb`.

7.19.2.3 `template<class charT, class traits> basic_iNetStream< charT, traits >::basic_iNetStream (const IPAddr::const_ipname_t & host, const IPAddr::ipport_t port) [inline]`

Definition at line 111 of file ionetstream.h.

References `basic_iNetStream< charT, traits >::nb`.

7.19.2.4 `template<class charT, class traits> basic_iNetStream< charT, traits >::basic_iNetStream (const IPAddr & addr) [inline]`

Definition at line 112 of file ionetstream.h.

References `basic_iNetStream< charT, traits >::nb`.

7.19.2.5 `template<class charT, class traits> basic_iNetStream< charT, traits >::basic_iNetStream (size_t buf_in_size, size_t buf_out_size) [inline]`

Definition at line 113 of file ionetstream.h.

References `basic_iNetStream< charT, traits >::nb`.

7.19.3 Member Function Documentation

7.19.3.1 `template<class charT, class traits> void basic_iNetStream< charT, traits >::close () [inline]`

Definition at line 127 of file ionetstream.h.

References `basic_iNetStream< charT, traits >::nb`.

7.19.3.2 `template<class charT, class traits> bool basic_iNetStream< charT, traits >::getEcho () [inline]`

Definition at line 129 of file ionetstream.h.

References `basic_iNetStream< charT, traits >::nb`.

7.19.3.3 `template<class charT, class traits> bool basic_iNetStream< charT, traits >::is_open () const` [inline]

Definition at line 121 of file ionetstream.h.

References basic_iNetStream< charT, traits >::nb.

7.19.3.4 `template<class charT, class traits> bool basic_iNetStream< charT, traits >::open (const IPAddr::const_ipname_t & ahost, unsigned int aPort)` [inline]

Definition at line 120 of file ionetstream.h.

References basic_iNetStream< charT, traits >::nb.

7.19.3.5 `template<class charT, class traits> bool basic_iNetStream< charT, traits >::open (const IPAddr::ipnum_t & addr, const IPAddr::ipport_t & aPort)` [inline]

Definition at line 119 of file ionetstream.h.

References basic_iNetStream< charT, traits >::nb.

7.19.3.6 `template<class charT, class traits> bool basic_iNetStream< charT, traits >::open (const IPAddr::const_ipname_t & str)` [inline]

Definition at line 118 of file ionetstream.h.

References basic_iNetStream< charT, traits >::nb.

7.19.3.7 `template<class charT, class traits> bool basic_iNetStream< charT, traits >::open (const IPAddr & addr)` [inline]

Definition at line 117 of file ionetstream.h.

References basic_iNetStream< charT, traits >::nb.

7.19.3.8 `template<class charT, class traits> basic_netbuf<charT, traits>* basic_iNetStream< charT, traits >::rddbuf () const` [inline]

Definition at line 115 of file ionetstream.h.

References basic_iNetStream< charT, traits >::nb.

7.19.3.9 `template<class charT, class traits> void basic_iNetStream< charT, traits >::setEcho (bool val = true) [inline]`

Definition at line 128 of file ionetstream.h.

References `basic_iNetStream< charT, traits >::nb`.

7.19.4 Member Data Documentation

7.19.4.1 `template<class charT, class traits> basic_netbuf<charT, traits> basic_iNetStream< charT, traits >::nb [private]`

Definition at line 131 of file ionetstream.h.

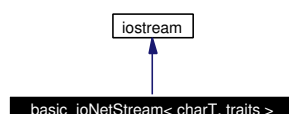
The documentation for this class was generated from the following file:

- [ionetstream.h](#)

7.20 basic_ioNetStream< charT, traits > Class Template Reference

```
#include <ionetstream.h>
```

Inheritance diagram for basic_ioNetStream< charT, traits >:



```
template<class charT, class traits> class basic_ioNetStream< charT, traits >
```

Public Types

- typedef charT [char_type](#)
- typedef traits::int_type [int_type](#)
- typedef traits::pos_type [pos_type](#)
- typedef traits::off_type [off_type](#)
- typedef traits [traits_type](#)

Public Member Functions

- [basic_ioNetStream](#) ()
- [basic_ioNetStream](#) (const IPaddr &addr)
- [basic_ioNetStream](#) (const IPaddr::ipnum_t &host, const IPaddr::ipport_t port)
- [basic_ioNetStream](#) (const IPaddr::const_ipname_t &host, const IPaddr::ipport_t port)
- [basic_ioNetStream](#) (size_t buf_in_size, size_t buf_out_size)
- [basic_netbuf](#)< charT, traits > * [rdbuf](#) () const
- bool [open](#) (const IPaddr &addr)
- bool [open](#) (const IPaddr::const_ipname_t &str)
- bool [open](#) (const IPaddr::ipnum_t &addr, const IPaddr::ipport_t &aPort)
- bool [open](#) (const IPaddr::const_ipname_t &ahost, unsigned int aPort)
- bool [is_open](#) ()
- void [close](#) ()
- void [setEcho](#) (bool val=true)
- bool [getEcho](#) ()

Private Attributes

- `basic_netbuf`< charT, traits > `nb`

7.20.1 Member Typedef Documentation

7.20.1.1 `template<class charT, class traits> typedef charT basic_ioNetStream< charT, traits >::char_type`

Definition at line 177 of file `ionetstream.h`.

7.20.1.2 `template<class charT, class traits> typedef traits::int_type basic_ioNetStream< charT, traits >::int_type`

Definition at line 178 of file `ionetstream.h`.

7.20.1.3 `template<class charT, class traits> typedef traits::off_type basic_ioNetStream< charT, traits >::off_type`

Definition at line 180 of file `ionetstream.h`.

7.20.1.4 `template<class charT, class traits> typedef traits::pos_type basic_ioNetStream< charT, traits >::pos_type`

Definition at line 179 of file `ionetstream.h`.

7.20.1.5 `template<class charT, class traits> typedef traits basic_ioNetStream< charT, traits >::traits_type`

Definition at line 181 of file `ionetstream.h`.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 `template<class charT, class traits> basic_ioNetStream< charT, traits >::basic_ioNetStream () [inline]`

Definition at line 183 of file `ionetstream.h`.

References `basic_ioNetStream`< charT, traits >::`nb`.

7.20.2.2 `template<class charT, class traits> basic_ioNetStream< charT, traits >::basic_ioNetStream (const IPaddr & addr) [inline]`

Definition at line 184 of file ionetstream.h.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.2.3 `template<class charT, class traits> basic_ioNetStream< charT, traits >::basic_ioNetStream (const IPaddr::ipnum_t & host, const IPaddr::ipport_t port) [inline]`

Definition at line 185 of file ionetstream.h.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.2.4 `template<class charT, class traits> basic_ioNetStream< charT, traits >::basic_ioNetStream (const IPaddr::const_ipname_t & host, const IPaddr::ipport_t port) [inline]`

Definition at line 186 of file ionetstream.h.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.2.5 `template<class charT, class traits> basic_ioNetStream< charT, traits >::basic_ioNetStream (size_t buf_in_size, size_t buf_out_size) [inline]`

Definition at line 187 of file ionetstream.h.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.3 Member Function Documentation

7.20.3.1 `template<class charT, class traits> void basic_ioNetStream< charT, traits >::close () [inline]`

Definition at line 201 of file ionetstream.h.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.3.2 `template<class charT, class traits> bool basic_ioNetStream< charT, traits >::getEcho () [inline]`

Definition at line 203 of file ionetstream.h.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.3.3 `template<class charT, class traits> bool basic_ioNetStream< charT, traits >::is_open () [inline]`

Definition at line 195 of file `ionetstream.h`.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.3.4 `template<class charT, class traits> bool basic_ioNetStream< charT, traits >::open (const IPAddr::const_ipname_t & ahost, unsigned int aPort) [inline]`

Definition at line 194 of file `ionetstream.h`.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.3.5 `template<class charT, class traits> bool basic_ioNetStream< charT, traits >::open (const IPAddr::ipnum_t & addr, const IPAddr::ipport_t & aPort) [inline]`

Definition at line 193 of file `ionetstream.h`.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.3.6 `template<class charT, class traits> bool basic_ioNetStream< charT, traits >::open (const IPAddr::const_ipname_t & str) [inline]`

Definition at line 192 of file `ionetstream.h`.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.3.7 `template<class charT, class traits> bool basic_ioNetStream< charT, traits >::open (const IPAddr & addr) [inline]`

Definition at line 191 of file `ionetstream.h`.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.3.8 `template<class charT, class traits> basic_netbuf<charT, traits>* basic_ioNetStream< charT, traits >::rdbuf () const [inline]`

Definition at line 189 of file `ionetstream.h`.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.3.9 `template<class charT, class traits> void basic_ioNetStream< charT, traits >::setEcho (bool val = true) [inline]`

Definition at line 202 of file `ionetstream.h`.

References `basic_ioNetStream< charT, traits >::nb`.

7.20.4 Member Data Documentation

7.20.4.1 `template<class charT, class traits> basic_netbuf<charT, traits> basic_ioNetStream< charT, traits >::nb [private]`

Definition at line 205 of file `ionetstream.h`.

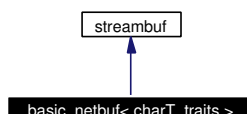
The documentation for this class was generated from the following file:

- [ionetstream.h](#)

7.21 `basic_netbuf< charT, traits >` Class Template Reference

```
#include <ionetstream.h>
```

Inheritance diagram for `basic_netbuf< charT, traits >`:



```
template<class charT, class traits> class basic_netbuf< charT, traits >
```

Public Types

- typedef charT [char_type](#)
- typedef traits::int_type [int_type](#)
- typedef traits::pos_type [pos_type](#)
- typedef traits::off_type [off_type](#)
- typedef traits [traits_type](#)

Public Member Functions

- [basic_netbuf](#) ()
- [basic_netbuf](#) (const IPaddr &addr)
- [basic_netbuf](#) (const IPaddr::ipnum_t &host, const IPaddr::ipport_t port)
- [basic_netbuf](#) (const IPaddr::const_ipname_t &host, const IPaddr::ipport_t port)
- [basic_netbuf](#) (size_t buf_in_size, size_t buf_out_size)
- virtual [~basic_netbuf](#) ()
- virtual bool [open](#) (const IPaddr &addr)
- virtual bool [open](#) (const IPaddr::const_ipname_t &str)
- virtual bool [open](#) (const IPaddr::ipnum_t &addr, const IPaddr::ipport_t &aPort)
- virtual bool [open](#) (const IPaddr::const_ipname_t &ahost, unsigned int aPort)
- virtual bool [is_open](#) () const
- virtual void [close](#) ()
- virtual void [setEcho](#) (bool val=true)
- virtual bool [getEcho](#) ()
- virtual void [in_sync](#) ()
- virtual void [out_flush](#) ()

Protected Member Functions

- void [Init](#) ()
- void [Init](#) (size_t insize, size_t outsize)
- virtual streamsize [showmanyc](#) ()
- virtual [int_type](#) [underflow](#) ()
- virtual [int_type](#) [uflow](#) ()
- virtual [int_type](#) [overflow](#) ([int_type](#) c=traits::eof())
- virtual int [sync](#) ()

Static Protected Member Functions

- void [printBuffer](#) (const char *buf, int buflen, const char *header)

Protected Attributes

- charT * [buf_in](#)
- charT * [buf_out](#)
- bool [using_buf_in](#)
- bool [using_buf_out](#)
- int [sock](#)
- bool [is_echoing](#)

Static Protected Attributes

- const size_t [def_buf_in_size](#) = 1<<8
- const size_t [def_buf_out_size](#) = 1<<12

7.21.1 Member Typedef Documentation

7.21.1.1 `template<class charT, class traits> typedef charT basic_netbuf< charT, traits >::char_type`

Definition at line 26 of file ionetstream.h.

7.21.1.2 `template<class charT, class traits> typedef traits::int_type basic_netbuf< charT, traits >::int_type`

Definition at line 27 of file ionetstream.h.

7.21.1.3 `template<class charT, class traits> typedef traits::off_type
basic_netbuf< charT, traits >::off_type`

Definition at line 29 of file ionetstream.h.

7.21.1.4 `template<class charT, class traits> typedef traits::pos_type
basic_netbuf< charT, traits >::pos_type`

Definition at line 28 of file ionetstream.h.

7.21.1.5 `template<class charT, class traits> typedef traits basic_netbuf<
charT, traits >::traits_type`

Definition at line 30 of file ionetstream.h.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 `template<class charT, class traits> basic_netbuf< charT, traits
>::basic_netbuf ()`

Definition at line 234 of file ionetstream.h.

References `basic_netbuf< charT, traits >::Init()`, and `INVALID_SOCKET`.

7.21.2.2 `template<class charT, class traits> basic_netbuf< charT, traits
>::basic_netbuf (const IPAddr & addr)`

Definition at line 240 of file ionetstream.h.

References `basic_netbuf< charT, traits >::Init()`, `INVALID_SOCKET`, and `basic_netbuf< charT, traits >::open()`.

7.21.2.3 `template<class charT, class traits> basic_netbuf< charT, traits
>::basic_netbuf (const IPAddr::ipnum_t & host, const IPAddr::ipport_t
port)`

Definition at line 248 of file ionetstream.h.

References `basic_netbuf< charT, traits >::Init()`, `INVALID_SOCKET`, and `basic_netbuf< charT, traits >::open()`.

7.21.2.4 `template<class charT, class traits> basic_netbuf< charT, traits
>::basic_netbuf (const IPAddr::const_ipname_t & host, const
IPAddr::ipport_t port)`

Definition at line 256 of file `ionetstream.h`.

References `basic_netbuf< charT, traits >::Init()`, `INVALID_SOCKET`, and `basic_netbuf< charT, traits >::open()`.

7.21.2.5 `template<class charT, class traits> basic_netbuf< charT, traits
>::basic_netbuf (size_t buf_in_size, size_t buf_out_size)`

Definition at line 264 of file `ionetstream.h`.

References `basic_netbuf< charT, traits >::Init()`, and `INVALID_SOCKET`.

7.21.2.6 `template<class charT, class traits> basic_netbuf< charT, traits
>::~basic_netbuf () [virtual]`

Definition at line 271 of file `ionetstream.h`.

References `basic_netbuf< charT, traits >::buf_in`, `basic_netbuf< charT, traits >::buf_out`, `basic_netbuf< charT, traits >::using_buf_in`, and `basic_netbuf< charT, traits >::using_buf_out`.

7.21.3 Member Function Documentation

7.21.3.1 `template<class charT, class traits> void basic_netbuf< charT, traits
>::close () [virtual]`

Definition at line 347 of file `ionetstream.h`.

References `INVALID_SOCKET`, `basic_netbuf< charT, traits >::is_open()`, and `basic_netbuf< charT, traits >::sock`.

7.21.3.2 `template<class charT, class traits> virtual bool basic_netbuf< charT,
traits >::getEcho () [inline, virtual]`

Definition at line 51 of file `ionetstream.h`.

References `basic_netbuf< charT, traits >::is_echoing`.

7.21.3.3 `template<class charT, class traits> void basic_netbuf< charT, traits >::in_sync () [virtual]`

Definition at line 377 of file `ionetstream.h`.

References `basic_netbuf< charT, traits >::close()`, `basic_netbuf< charT, traits >::is_escaping`, `basic_netbuf< charT, traits >::is_open()`, `basic_netbuf< charT, traits >::print_Buffer()`, and `basic_netbuf< charT, traits >::sock`.

7.21.3.4 `template<class charT, class traits> void basic_netbuf< charT, traits >::Init (size_t insize, size_t outsize) [protected]`

Definition at line 280 of file `ionetstream.h`.

References `basic_netbuf< charT, traits >::buf_in`, `basic_netbuf< charT, traits >::buf_out`, `basic_netbuf< charT, traits >::using_buf_in`, and `basic_netbuf< charT, traits >::using_buf_out`.

7.21.3.5 `template<class charT, class traits> void basic_netbuf< charT, traits >::Init () [inline, protected]`

Definition at line 54 of file `ionetstream.h`.

References `basic_netbuf< charT, traits >::def_buf_in_size`, and `basic_netbuf< charT, traits >::def_buf_out_size`.

7.21.3.6 `template<class charT, class traits> virtual bool basic_netbuf< charT, traits >::is_open () const [inline, virtual]`

Definition at line 47 of file `ionetstream.h`.

References `INVALID_SOCKET`, and `basic_netbuf< charT, traits >::sock`.

7.21.3.7 `template<class charT, class traits> virtual bool basic_netbuf< charT, traits >::open (const IPAddr::const_ipname_t & ahost, unsigned int aPort) [inline, virtual]`

Definition at line 46 of file `ionetstream.h`.

References `basic_netbuf< charT, traits >::open()`.

7.21.3.8 `template<class charT, class traits> bool basic_netbuf< charT, traits >::open (const IPAddr::ipnum_t & addr, const IPAddr::ipport_t & aPort) [virtual]`

Definition at line 318 of file ionetstream.h.

References INVALID_SOCKET, basic_netbuf< charT, traits >::sock, and SocketNS::SOCK_STREAM.

7.21.3.9 `template<class charT, class traits> bool basic_netbuf< charT, traits >::open (const IPAddr::const_ipname_t & str) [virtual]`

Definition at line 301 of file ionetstream.h.

References basic_netbuf< charT, traits >::open().

7.21.3.10 `template<class charT, class traits> bool basic_netbuf< charT, traits >::open (const IPAddr & addr) [virtual]`

Definition at line 293 of file ionetstream.h.

7.21.3.11 `template<class charT, class traits> void basic_netbuf< charT, traits >::out_flush () [virtual]`

Definition at line 406 of file ionetstream.h.

References basic_netbuf< charT, traits >::close(), basic_netbuf< charT, traits >::is_escaping(), basic_netbuf< charT, traits >::is_open(), basic_netbuf< charT, traits >::printBuffer(), and basic_netbuf< charT, traits >::sock.

7.21.3.12 `template<class charT, class traits> basic_netbuf< charT, traits >::int_type basic_netbuf< charT, traits >::overflow (int_type c = traits::eof()) [inline, protected, virtual]`

Definition at line 468 of file ionetstream.h.

References basic_netbuf< charT, traits >::is_open(), and basic_netbuf< charT, traits >::out_flush().

7.21.3.13 `template<class charT, class traits> void basic_netbuf< charT, traits >::printBuffer (const char * buf, int buflen, const char * header) [static, protected]`

Definition at line 361 of file ionetstream.h.

7.21.3.14 `template<class charT, class traits> virtual void basic_netbuf< charT, traits >::setEcho (bool val = true) [inline, virtual]`

Definition at line 50 of file ionetstream.h.

References `basic_netbuf< charT, traits >::is_echoing`.

7.21.3.15 `template<class charT, class traits> streamsize basic_netbuf< charT, traits >::showmanyc () [inline, protected, virtual]`

Definition at line 437 of file ionetstream.h.

7.21.3.16 `template<class charT, class traits> int basic_netbuf< charT, traits >::sync () [inline, protected, virtual]`

Definition at line 477 of file ionetstream.h.

References `basic_netbuf< charT, traits >::in_sync()`, `basic_netbuf< charT, traits >::is_open()`, and `basic_netbuf< charT, traits >::out_flush()`.

7.21.3.17 `template<class charT, class traits> basic_netbuf< charT, traits >::int_type basic_netbuf< charT, traits >::uflow () [inline, protected, virtual]`

Definition at line 455 of file ionetstream.h.

References `basic_netbuf< charT, traits >::in_sync()`, and `basic_netbuf< charT, traits >::int_type`.

7.21.3.18 `template<class charT, class traits> basic_netbuf< charT, traits >::int_type basic_netbuf< charT, traits >::underflow () [inline, protected, virtual]`

Definition at line 443 of file ionetstream.h.

References `basic_netbuf< charT, traits >::in_sync()`.

7.21.4 Member Data Documentation

7.21.4.1 `template<class charT, class traits> charT* basic_netbuf< charT, traits >::buf_in [protected]`

Definition at line 85 of file ionetstream.h.

7.21.4.2 `template<class charT, class traits> charT * basic_netbuf< charT, traits >::buf_out [protected]`

Definition at line 85 of file `ionetstream.h`.

7.21.4.3 `template<class charT, class traits> const size_t basic_netbuf< charT, traits >::def_buf_in_size = 1<<8 [static, protected]`

Definition at line 93 of file `ionetstream.h`.

7.21.4.4 `template<class charT, class traits> const size_t basic_netbuf< charT, traits >::def_buf_out_size = 1<<12 [static, protected]`

Definition at line 95 of file `ionetstream.h`.

7.21.4.5 `template<class charT, class traits> bool basic_netbuf< charT, traits >::is_echoing [protected]`

Definition at line 90 of file `ionetstream.h`.

7.21.4.6 `template<class charT, class traits> int basic_netbuf< charT, traits >::sock [protected]`

Definition at line 89 of file `ionetstream.h`.

7.21.4.7 `template<class charT, class traits> bool basic_netbuf< charT, traits >::using_buf_in [protected]`

Definition at line 86 of file `ionetstream.h`.

7.21.4.8 `template<class charT, class traits> bool basic_netbuf< charT, traits >::using_buf_out [protected]`

Definition at line 86 of file `ionetstream.h`.

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

- [ionetstream.h](#)

7.22 `basic_oNetStream< charT, traits >` Class Template Reference

```
#include <ionetstream.h>
```

Inheritance diagram for `basic_oNetStream< charT, traits >`:



```
template<class charT, class traits> class basic_oNetStream< charT, traits >
```

Public Types

- typedef `charT` [char_type](#)
- typedef `traits::int_type` [int_type](#)
- typedef `traits::pos_type` [pos_type](#)
- typedef `traits::off_type` [off_type](#)
- typedef `traits` [traits_type](#)

Public Member Functions

- [basic_oNetStream](#) ()
- [basic_oNetStream](#) (const `IPAddr::ipnum_t` &host, const `IPAddr::ipport_t` port)
- [basic_oNetStream](#) (const `IPAddr::const_ipname_t` &host, const `IPAddr::ipport_t` port)
- [basic_oNetStream](#) (const `IPAddr` &addr)
- [basic_oNetStream](#) (size_t buf_in_size, size_t buf_out_size)
- [basic_netbuf](#)< `charT`, `traits` > * [rdbuf](#) () const
- bool [open](#) (const `IPAddr` &addr)
- bool [open](#) (const `IPAddr::const_ipname_t` &str)
- bool [open](#) (const `IPAddr::ipnum_t` &addr, const `IPAddr::ipport_t` &aPort)
- bool [open](#) (const `IPAddr::const_ipname_t` &ahost, unsigned int aPort)
- bool [is_open](#) () const
- void [close](#) ()
- void [setEcho](#) (bool val=true)
- bool [getEcho](#) ()

Private Attributes

- [basic_netbuf](#)< charT, traits > [nb](#)

7.22.1 Member Typedef Documentation

7.22.1.1 `template<class charT, class traits> typedef charT basic_oNetStream< charT, traits >::char_type`

Definition at line 140 of file `ionetstream.h`.

7.22.1.2 `template<class charT, class traits> typedef traits::int_type basic_oNetStream< charT, traits >::int_type`

Definition at line 141 of file `ionetstream.h`.

7.22.1.3 `template<class charT, class traits> typedef traits::off_type basic_oNetStream< charT, traits >::off_type`

Definition at line 143 of file `ionetstream.h`.

7.22.1.4 `template<class charT, class traits> typedef traits::pos_type basic_oNetStream< charT, traits >::pos_type`

Definition at line 142 of file `ionetstream.h`.

7.22.1.5 `template<class charT, class traits> typedef traits basic_oNetStream< charT, traits >::traits_type`

Definition at line 144 of file `ionetstream.h`.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 `template<class charT, class traits> basic_oNetStream< charT, traits >::basic_oNetStream() [inline]`

Definition at line 146 of file `ionetstream.h`.

References `basic_oNetStream< charT, traits >::nb`.

7.22.2.2 `template<class charT, class traits> basic_oNetStream< charT, traits >::basic_oNetStream (const IPAddr::ipnum.t & host, const IPAddr::ipport.t port) [inline]`

Definition at line 147 of file ionetstream.h.

References `basic_oNetStream< charT, traits >::nb`.

7.22.2.3 `template<class charT, class traits> basic_oNetStream< charT, traits >::basic_oNetStream (const IPAddr::const_ipname.t & host, const IPAddr::ipport.t port) [inline]`

Definition at line 148 of file ionetstream.h.

References `basic_oNetStream< charT, traits >::nb`.

7.22.2.4 `template<class charT, class traits> basic_oNetStream< charT, traits >::basic_oNetStream (const IPAddr & addr) [inline]`

Definition at line 149 of file ionetstream.h.

References `basic_oNetStream< charT, traits >::nb`.

7.22.2.5 `template<class charT, class traits> basic_oNetStream< charT, traits >::basic_oNetStream (size.t buf_in_size, size.t buf_out_size) [inline]`

Definition at line 150 of file ionetstream.h.

References `basic_oNetStream< charT, traits >::nb`.

7.22.3 Member Function Documentation

7.22.3.1 `template<class charT, class traits> void basic_oNetStream< charT, traits >::close () [inline]`

Definition at line 164 of file ionetstream.h.

References `basic_oNetStream< charT, traits >::nb`.

7.22.3.2 `template<class charT, class traits> bool basic_oNetStream< charT, traits >::getEcho () [inline]`

Definition at line 166 of file ionetstream.h.

References basic_oNetStream< charT, traits >::nb.

7.22.3.3 `template<class charT, class traits> bool basic_oNetStream< charT, traits >::is_open () const [inline]`

Definition at line 158 of file ionetstream.h.

References basic_oNetStream< charT, traits >::nb.

7.22.3.4 `template<class charT, class traits> bool basic_oNetStream< charT, traits >::open (const IPAddr::const_ipname_t & ahost, unsigned int aPort) [inline]`

Definition at line 157 of file ionetstream.h.

References basic_oNetStream< charT, traits >::nb.

7.22.3.5 `template<class charT, class traits> bool basic_oNetStream< charT, traits >::open (const IPAddr::ipnum_t & addr, const IPAddr::ipport_t & aPort) [inline]`

Definition at line 156 of file ionetstream.h.

References basic_oNetStream< charT, traits >::nb.

7.22.3.6 `template<class charT, class traits> bool basic_oNetStream< charT, traits >::open (const IPAddr::const_ipname_t & str) [inline]`

Definition at line 155 of file ionetstream.h.

References basic_oNetStream< charT, traits >::nb.

7.22.3.7 `template<class charT, class traits> bool basic_oNetStream< charT, traits >::open (const IPAddr & addr) [inline]`

Definition at line 154 of file ionetstream.h.

References basic_oNetStream< charT, traits >::nb.

7.22.3.8 `template<class charT, class traits> basic_netbuf<charT, traits>* basic_oNetStream< charT, traits >::rdbuf () const [inline]`

Definition at line 152 of file ionetstream.h.

References basic_oNetStream< charT, traits >::nb.

7.22.3.9 `template<class charT, class traits> void basic_oNetStream< charT, traits >::setEcho (bool val = true) [inline]`

Definition at line 165 of file ionetstream.h.

References `basic_oNetStream< charT, traits >::nb`.

7.22.4 Member Data Documentation

7.22.4.1 `template<class charT, class traits> basic_netbuf<charT, traits> basic_oNetStream< charT, traits >::nb [private]`

Definition at line 168 of file ionetstream.h.

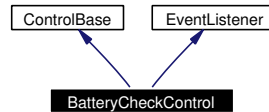
The documentation for this class was generated from the following file:

- [ionetstream.h](#)

7.23 BatteryCheckControl Class Reference

```
#include <BatteryCheckControl.h>
```

Inheritance diagram for BatteryCheckControl:



7.23.1 Detailed Description

when activated, this will print a battery report to stdout and light up LEDs to specify power level

The LEDs use the [LedEngine::displayPercent\(\)](#) function, with minor/major style. This means the left column (viewing the dog head on) will show the overall power level, and the right column will show the level within the tick lit up in the left column. The more geeky among you may prefer to think of this as a two digit base 5 display.

This gives you pretty precise visual feedback as to remaining power (perhaps more than you really need, but it's as much a demo as a useful tool)

This is implemented as a Control instead of a Behavior on the assumption you wouldn't want to leave this running while you were doing other things (ie not in e-stop). But it definitely blurs the line between the two.

Definition at line 24 of file BatteryCheckControl.h.

Public Member Functions

- [BatteryCheckControl](#) ()
Constructor.
- virtual [~BatteryCheckControl](#) ()
Destructor.
- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#) display, [Socket](#) *gui)
Prints a report to stdio and lights up the face to show battery level.
- virtual void [pause](#) ()
stops listening for power events and sets display to invalid

- virtual void [refresh](#) ()
calls [report\(\)](#)
- virtual void [deactivate](#) ()
stops listening for power events and sets display to invalid
- virtual void [processEvent](#) (const [EventBase](#) &event)
calls [refresh\(\)](#) to redisplay with new information if it's not a vibration event
- virtual [ControlBase](#) * [doSelect](#) ()
when the user has trigger an "open selection" - default is to return the hilighted control
- void [report](#) ()
redisplay text to cout and refresh LED values

7.23.2 Constructor & Destructor Documentation

7.23.2.1 [BatteryCheckControl::BatteryCheckControl](#) () [inline]

Constructor.

Definition at line 28 of file [BatteryCheckControl.h](#).

7.23.2.2 virtual [BatteryCheckControl::~~BatteryCheckControl](#) () [inline, virtual]

Destructor.

Definition at line 31 of file [BatteryCheckControl.h](#).

7.23.3 Member Function Documentation

7.23.3.1 virtual [ControlBase](#)* [BatteryCheckControl::activate](#) ([MotionManager::MC_ID](#) *display*, [Socket](#) * *gui*) [inline, virtual]

Prints a report to stdio and lights up the face to show battery level.

keeps running until deactivated - will listen for power events and continue to update display

Reimplemented from [ControlBase](#).

Definition at line 35 of file BatteryCheckControl.h.

References [ControlBase::activate\(\)](#), [EventRouter::addListener\(\)](#), [erouter](#), and [EventBase::powerEGID](#).

7.23.3.2 **virtual void BatteryCheckControl::deactivate ()** [inline, virtual]

stops listening for power events and sets display to invalid

Reimplemented from [ControlBase](#).

Definition at line 63 of file BatteryCheckControl.h.

References [ControlBase::display_id](#), [erouter](#), [EventRouter::forgetListener\(\)](#), and [MotionManager::invalid_MC_ID](#).

7.23.3.3 **virtual [ControlBase](#)* BatteryCheckControl::doSelect ()** [inline, virtual]

when the user has trigger an "open selection" - default is to return the hilighted control

Reimplemented from [ControlBase](#).

Definition at line 72 of file BatteryCheckControl.h.

7.23.3.4 **virtual void BatteryCheckControl::pause ()** [inline, virtual]

stops listening for power events and sets display to invalid

Reimplemented from [ControlBase](#).

Definition at line 41 of file BatteryCheckControl.h.

References [ControlBase::display_id](#), [erouter](#), [EventRouter::forgetListener\(\)](#), and [MotionManager::invalid_MC_ID](#).

7.23.3.5 **virtual void BatteryCheckControl::processEvent (const [EventBase](#) & event)** [inline, virtual]

calls [refresh\(\)](#) to redisplay with new information if it's not a vibration event

Implements [EventListener](#).

Definition at line 68 of file BatteryCheckControl.h.

References [EventBase::getSourceID\(\)](#), and [refresh\(\)](#).

7.23.3.6 virtual void BatteryCheckControl::refresh () [inline, virtual]

calls [report\(\)](#)

Reimplemented from [ControlBase](#).

Definition at line 46 of file BatteryCheckControl.h.

References [ControlBase::getName\(\)](#), [ControlBase::gui_comm](#), [Wireless::is-Connected\(\)](#), [ERS210Info::PowerRemainOffset](#), [report\(\)](#), [WorldState::sensors](#), [Socket::sock](#), [state](#), [wireless](#), and [Socket::write\(\)](#).

7.23.3.7 void BatteryCheckControl::report () [inline]

redisplay text to cout and refresh LED values

Definition at line 76 of file BatteryCheckControl.h.

References [ControlBase::display_id](#), [MotionManager::invalid_MC_ID](#), [Led-Engine::major](#), [ERS210Info::PowerCapacityOffset](#), [ERS210Info::Power-CurrentOffset](#), [WorldState::powerFlags](#), [ERS210Info::PowerRemainOffset](#), [ERS210Info::PowerThermoOffset](#), [ERS210Info::PowerVoltageOffset](#), [World-
State::sensors](#), and [state](#).

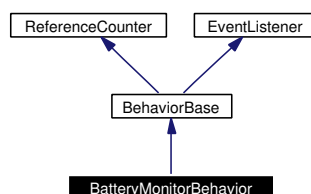
The documentation for this class was generated from the following file:

- [BatteryCheckControl.h](#)

7.24 BatteryMonitorBehavior Class Reference

```
#include <BatteryMonitorBehavior.h>
```

Inheritance diagram for BatteryMonitorBehavior:



7.24.1 Detailed Description

A background behavior which will monitor the power level and flip the ears when appropriate on a 210, or blink the headlight if a 220.

Think of this as a simple example class. For exercise, try using a [MotionSequenceMC](#) instead of switching the ears back manually using a [PostureMC](#)

Definition at line 19 of file BatteryMonitorBehavior.h.

Public Member Functions

- [BatteryMonitorBehavior](#) ()
constructor
- virtual [~BatteryMonitorBehavior](#) ()
destructor
- virtual void [DoStart](#) ()
Listens for the [PowerSourceID::LowPowerWarnSID](#).
- virtual void [DoStop](#) ()
Stops listening for events.
- virtual void [processEvent](#) (const [EventBase](#) &event)
Adds a [BatteryMonitorMC](#) to motman if power goes low.
- virtual std::string [getName](#) () const
Identifies the behavior in menus and such.

Static Public Member Functions

- `std::string getClassDescription ()`
Gives a short description of what this class of behaviors does... you should override this (but don't have to).
- `bool shouldWarn ()`
returns true if the warning should be active (power remaining less than high_power_p, no external power, but also checks that a power update has been received)

Static Public Attributes

- `const unsigned int max_t = 10000`
max time between ear flips when at "high power" mark
- `const unsigned int high_power_p = 20`
percent of 100 which is point at which to begin warning
- `const unsigned int no_power_p = 14`
percent of 100 at which power will fail (approximate!)

Protected Member Functions

- `void startWarning ()`
adds a pose and a timer to get the ears flipping
- `void stopWarning ()`
removes pose, in case battery magically charges
- `unsigned int calcFlipDelay ()`
makes the ears flip more rapidly as power declines. Flips back and forth once every 15 seconds at 15%, down to flipping constantly at 5%.
- `void setFlipper (bool set)`
sets the ears on a 210 or the headlight on a 220 - true toggles current, false clears

Protected Attributes

- [PostureMC * pose](#)
if we are currently warning of low battery, holds a pose, NULL otherwise
- [MotionManager::MC_ID pose_id](#)
id of pose if we are currently warning, [MotionManager::invalid_MC_ID](#) otherwise
- [MotionManager::MC_ID led_id](#)
id of [LedMC](#) if we are currently warning, [MotionManager::invalid_MC_ID](#) otherwise

Private Member Functions

- [BatteryMonitorBehavior](#) (const [BatteryMonitorBehavior](#) &)
don't copy behaviors
- [BatteryMonitorBehavior](#) operator= (const [BatteryMonitorBehavior](#) &)
don't assign behaviors

7.24.2 Constructor & Destructor Documentation

7.24.2.1 [BatteryMonitorBehavior::BatteryMonitorBehavior \(\)](#) [inline]

constructor

Definition at line 26 of file [BatteryMonitorBehavior.h](#).

References [led_id](#), [pose](#), and [pose_id](#).

7.24.2.2 [virtual BatteryMonitorBehavior::~~BatteryMonitorBehavior \(\)](#) [inline, virtual]

destructor

Definition at line 28 of file [BatteryMonitorBehavior.h](#).

7.24.2.3 [BatteryMonitorBehavior::BatteryMonitorBehavior \(const \[BatteryMonitorBehavior\]\(#\) &\)](#) [private]

don't copy behaviors

7.24.3 Member Function Documentation

7.24.3.1 `unsigned int BatteryMonitorBehavior::calcFlipDelay ()` [`inline`, `protected`]

makes the ears flip more rapidly as power declines. Flips back and forth once every 15 seconds at 15%, down to flipping constantly at 5%.

Definition at line 124 of file `BatteryMonitorBehavior.h`.

References `high_power_p`, `max_t`, `no_power_p`, `ERS210Info::PowerRemainOffset`, `WorldState::sensors`, and `state`.

7.24.3.2 `virtual void BatteryMonitorBehavior::DoStart ()` [`inline`, `virtual`]

Listens for the [PowerSourceID::LowPowerWarnSID](#).

Reimplemented from [BehaviorBase](#).

Definition at line 31 of file `BatteryMonitorBehavior.h`.

References `EventRouter::addListener()`, `BehaviorBase::DoStart()`, `erouter`, `EventBase::powerEGID`, `processEvent()`, `shouldWarn()`, and `EventBase::statusETID`.

7.24.3.3 `virtual void BatteryMonitorBehavior::DoStop ()` [`inline`, `virtual`]

Stops listening for events.

Reimplemented from [BehaviorBase](#).

Definition at line 42 of file `BatteryMonitorBehavior.h`.

References `BehaviorBase::DoStop()`, `erouter`, `EventRouter::forgetListener()`, `pose`, and `stopWarning()`.

7.24.3.4 `std::string BatteryMonitorBehavior::getClassDescription ()` [`inline`, `static`]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 96 of file `BatteryMonitorBehavior.h`.

7.24.3.5 `virtual std::string BatteryMonitorBehavior::getName () const` [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 95 of file BatteryMonitorBehavior.h.

7.24.3.6 `BatteryMonitorBehavior BatteryMonitorBehavior::operator= (const BatteryMonitorBehavior &) [private]`

don't assign behaviors

7.24.3.7 `virtual void BatteryMonitorBehavior::processEvent (const EventBase & event) [inline, virtual]`

Adds a BatteryMonitorMC to motman if power goes low.

Reimplemented from [BehaviorBase](#).

Definition at line 49 of file BatteryMonitorBehavior.h.

References [EventRouter::addTimer\(\)](#), [ASSERTRET](#), [calcFlipDelay\(\)](#), [erouter](#), [ERS210Info::FrameTime](#), [EventBase::getGeneratorID\(\)](#), [EventBase::getName\(\)](#), [EventBase::getSourceID\(\)](#), [MotionManager::kEmergencyPriority](#), [MotionManager::kIgnoredPriority](#), [led_id](#), [LedEngine::major](#), [motman](#), [ERS210Info::NumFrames](#), [pose](#), [EventBase::powerEGID](#), [ERS210Info::PowerRemainOffset](#), [WorldState::sensors](#), [setFlipper\(\)](#), [MotionManager::setPriority\(\)](#), [shouldWarn\(\)](#), [startWarning\(\)](#), [state](#), [stopWarning\(\)](#), and [EventBase::timerEGID](#).

7.24.3.8 `void BatteryMonitorBehavior::setFlipper (bool set) [inline, protected]`

sets the ears on a 210 or the headlight on a 220 - true toggles current, false clears

Definition at line 133 of file BatteryMonitorBehavior.h.

References [WorldState::ERS210Mask](#), [WorldState::ERS220Mask](#), [WorldState::outputs](#), [pose](#), [WorldState::robotDesign](#), [PostureMC::setOutputCmd\(\)](#), and [state](#).

7.24.3.9 `bool BatteryMonitorBehavior::shouldWarn () [inline, static]`

returns true if the warning should be active (power remaining less than `high_power_p`, no external power, but also checks that a power update has been received)

Definition at line 99 of file BatteryMonitorBehavior.h.

References `high_power_p`, `WorldState::powerFlags`, `ERS210Info::PowerRemainOffset`, `WorldState::sensors`, and `state`.

7.24.3.10 `void BatteryMonitorBehavior::startWarning ()` [inline, protected]

adds a pose and a timer to get the ears flipping

Definition at line 103 of file BatteryMonitorBehavior.h.

References `MotionManager::addMotion()`, `EventRouter::addTimer()`, `erouter`, `MotionManager::kEmergencyPriority`, `led_id`, `LedEngine::major`, `motman`, `MotionManager::peekMotion()`, `pose`, `pose_id`, `ERS210Info::PowerRemainOffset`, `WorldState::sensors`, `setFlipper()`, and `state`.

7.24.3.11 `void BatteryMonitorBehavior::stopWarning ()` [inline, protected]

removes pose, in case battery magically charges

Definition at line 114 of file BatteryMonitorBehavior.h.

References `erouter`, `MotionManager::invalid_MC_ID`, `led_id`, `motman`, `pose`, `pose_id`, `MotionManager::removeMotion()`, and `EventRouter::removeTimer()`.

7.24.4 Member Data Documentation

7.24.4.1 `const unsigned int BatteryMonitorBehavior::high_power_p = 20` [static]

percent of 100 which is point at which to begin warning

Definition at line 22 of file BatteryMonitorBehavior.h.

7.24.4.2 `MotionManager::MC_ID BatteryMonitorBehavior::led_id` [protected]

id of `LedMC` if we are currently warning, `MotionManager::invalid_MC_ID` otherwise

Definition at line 142 of file BatteryMonitorBehavior.h.

7.24.4.3 `const unsigned int BatteryMonitorBehavior::max_t = 10000`
[static]

max time between ear flips when at "high power" mark

Definition at line 21 of file BatteryMonitorBehavior.h.

7.24.4.4 `const unsigned int BatteryMonitorBehavior::no_power_p = 14`
[static]

percent of 100 at which power will fail (approximate!)

Definition at line 23 of file BatteryMonitorBehavior.h.

7.24.4.5 `PostureMC* BatteryMonitorBehavior::pose` [protected]

if we are currently warning of low battery, holds a pose, NULL otherwise

Definition at line 140 of file BatteryMonitorBehavior.h.

7.24.4.6 `MotionManager::MC_ID BatteryMonitorBehavior::pose_id`
[protected]

id of pose if we are currently warning, `MotionManager::invalid_MC_ID` otherwise

Definition at line 141 of file BatteryMonitorBehavior.h.

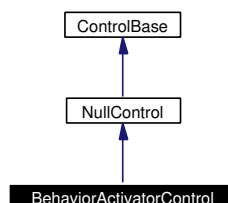
The documentation for this class was generated from the following file:

- [BatteryMonitorBehavior.h](#)

7.25 BehaviorActivatorControl Class Reference

```
#include <BehaviorActivatorControl.h>
```

Inheritance diagram for BehaviorActivatorControl:



7.25.1 Detailed Description

Upon activation, will start, stop, or toggle a behavior.

Definition at line 8 of file BehaviorActivatorControl.h.

Public Types

- enum [Mode_t](#) { [start](#), [stop](#), [toggle](#) }
lets you tell it what action to perform

Public Member Functions

- virtual [~BehaviorActivatorControl](#) ()
destructor
- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#) disp_id, [Socket](#) *gui)
performs the action denoted by [mode](#)
- [BehaviorActivatorControl](#) ([BehaviorBase](#) *behave, [Mode_t](#) m=toggle)
constructors
- [BehaviorActivatorControl](#) (const std::string &n, [BehaviorBase](#) *behave, [Mode_t](#) m=toggle)
constructors

- [BehaviorActivatorControl](#) (const std::string &n, const std::string &d, [BehaviorBase](#) *behave, [Mode_t](#) m=toggle)

constructors

Protected Member Functions

- void [init](#) ()

adds to target's reference counter

Protected Attributes

- [BehaviorBase](#) * [target](#)

The behavior to activate/deactivate.

- [Mode_t](#) [mode](#)

the mode this control is in

Private Member Functions

- [BehaviorActivatorControl](#) (const [BehaviorActivatorControl](#) &)

don't copy this class

- [BehaviorActivatorControl](#) operator= (const [BehaviorActivatorControl](#) &)

don't assign this class

7.25.2 Member Enumeration Documentation

7.25.2.1 enum [BehaviorActivatorControl::Mode_t](#)

lets you tell it what action to perform

Enumeration values:

start

stop

toggle

Definition at line 11 of file BehaviorActivatorControl.h.

7.25.3 Constructor & Destructor Documentation

7.25.3.1 BehaviorActivatorControl::BehaviorActivatorControl ([BehaviorBase](#) * *behave*, [Mode.t](#) *m* = toggle) [inline]

constructors

Definition at line 15 of file BehaviorActivatorControl.h.

References [init\(\)](#), [mode](#), [start](#), [target](#), and [toggle](#).

7.25.3.2 BehaviorActivatorControl::BehaviorActivatorControl (const [std::string](#) & *n*, [BehaviorBase](#) * *behave*, [Mode.t](#) *m* = toggle) [inline]

constructors

Definition at line 16 of file BehaviorActivatorControl.h.

References [init\(\)](#), [mode](#), [start](#), [target](#), and [toggle](#).

7.25.3.3 BehaviorActivatorControl::BehaviorActivatorControl (const [std::string](#) & *n*, const [std::string](#) & *d*, [BehaviorBase](#) * *behave*, [Mode.t](#) *m* = toggle) [inline]

constructors

Definition at line 17 of file BehaviorActivatorControl.h.

References [init\(\)](#), [mode](#), and [target](#).

7.25.3.4 virtual BehaviorActivatorControl::~~BehaviorActivatorControl () [inline, virtual]

destructor

Definition at line 21 of file BehaviorActivatorControl.h.

References [ReferenceCounter::RemoveReference\(\)](#), and [target](#).

7.25.3.5 BehaviorActivatorControl::BehaviorActivatorControl (const [BehaviorActivatorControl](#) &) [private]

don't copy this class

7.25.4 Member Function Documentation

7.25.4.1 `virtual ControlBase* BehaviorActivatorControl::activate(MotionManager::MC_ID disp_id, Socket * gui)` [inline, virtual]

performs the action denoted by [mode](#)

Reimplemented from [NullControl](#).

Definition at line 24 of file BehaviorActivatorControl.h.

References [NullControl::activate\(\)](#), [BehaviorBase::DoStart\(\)](#), [BehaviorBase::DoStop\(\)](#), [BehaviorBase::isActive\(\)](#), [mode](#), [start](#), [stop](#), [target](#), and [toggle](#).

7.25.4.2 `void BehaviorActivatorControl::init()` [inline, protected]

adds to target's reference counter

Definition at line 48 of file BehaviorActivatorControl.h.

References [ReferenceCounter::AddReference\(\)](#), and [target](#).

7.25.4.3 `BehaviorActivatorControl BehaviorActivatorControl::operator=(const BehaviorActivatorControl &)` [private]

don't assign this class

7.25.5 Member Data Documentation

7.25.5.1 `Mode_t BehaviorActivatorControl::mode` [protected]

the mode this control is in

Definition at line 53 of file BehaviorActivatorControl.h.

7.25.5.2 `BehaviorBase* BehaviorActivatorControl::target` [protected]

The behavior to activate/deactivate.

Definition at line 52 of file BehaviorActivatorControl.h.

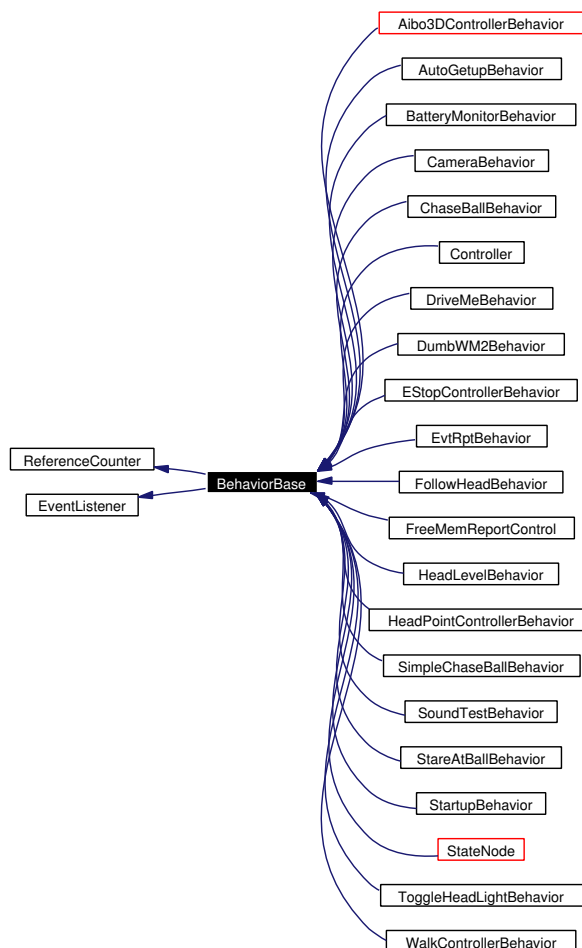
The documentation for this class was generated from the following file:

- [BehaviorActivatorControl.h](#)

7.26 BehaviorBase Class Reference

```
#include <BehaviorBase.h>
```

Inheritance diagram for BehaviorBase:



7.26.1 Detailed Description

The basis from which all other Behaviors should inherit.

Makes use of [ReferenceCounter](#) so that behaviors can automatically delete themselves if wanted

Make sure your own DoStart and DoStop call [BehaviorBase::DoStart](#) (or Stop) to allow

this behavior... otherwise you'll get memory leaks

Definition at line 12 of file BehaviorBase.h.

Public Member Functions

- [BehaviorBase](#) ()
constructor
- virtual [~BehaviorBase](#) ()
destructor - if is active when deleted, will call [DoStop\(\)](#) first
- virtual void [DoStart](#) ()
By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.
- virtual void [DoStop](#) ()
By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).
- virtual void [processEvent](#) (const [EventBase](#) &)
Allows you to get away with not supplying a [processEvent\(\)](#) function for the [Event-Listener](#) interface. By default, does nothing.
- virtual std::string [getName](#) () const=0
Identifies the behavior in menus and such.
- virtual std::string [getDescription](#) () const
Gives a short description of what this particular instantiation does (in case a more specific description is needed).
- virtual bool [isActive](#) () const
Returns true if the behavior is currently running.

Static Public Member Functions

- std::string [getClassDescription](#) ()
Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- bool [started](#)

true when the behavior is active

7.26.2 Constructor & Destructor Documentation

7.26.2.1 BehaviorBase::BehaviorBase () [inline]

constructor

Definition at line 15 of file BehaviorBase.h.

References [started](#).

7.26.2.2 virtual BehaviorBase::~BehaviorBase () [inline, virtual]

destructor - if is active when deleted, will call [DoStop\(\)](#) first

Definition at line 17 of file BehaviorBase.h.

References [DoStop\(\)](#), [ReferenceCounter::SetAutoDelete\(\)](#), and [started](#).

7.26.3 Member Function Documentation

7.26.3.1 virtual void BehaviorBase::DoStart () [inline, virtual]

By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.

Reimplemented in [Controller](#), [FreeMemReportControl](#), [Aibo3DControllerBehavior](#), [AutoGetupBehavior](#), [BanditMachine](#), [BanditMachine::PressNode](#), [BanditMachine::DecideNode](#), [BanditMachine::WaitNode](#), [BatteryMonitorBehavior](#), [CameraBehavior](#), [ChaseBallBehavior](#), [DriveMeBehavior](#), [DumbWM2Behavior](#), [EStopControllerBehavior](#), [EvtRptBehavior](#), [FollowHeadBehavior](#), [HeadLevelBehavior](#), [HeadPointControllerBehavior](#), [SimpleChaseBallBehavior](#), [SoundTestBehavior](#), [StareAtBallBehavior](#), [ToggleHeadLightBehavior](#), [WalkControllerBehavior](#), [WalkToTargetMachine](#), [WorldModel2Behavior](#), [WorldModel2Behavior::WalkNode](#), [WorldModel2Behavior::GawkNode](#), [WorldModel2Behavior::WaitNode](#), [OutputNode](#), [StateNode](#), and [StartupBehavior](#).

Definition at line 19 of file BehaviorBase.h.

References [ReferenceCounter::AddReference\(\)](#), and [started](#).

7.26.3.2 virtual void BehaviorBase::DoStop () [inline, virtual]

By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).

Reimplemented in [Controller](#), [FreeMemReportControl](#), [Aibo3DControllerBehavior](#), [AutoGetupBehavior](#), [BanditMachine](#), [BanditMachine::PressNode](#), [BanditMachine::WaitNode](#), [BatteryMonitorBehavior](#), [CameraBehavior](#), [ChaseBallBehavior](#), [DriveMeBehavior](#), [DumbWM2Behavior](#), [EStopControllerBehavior](#), [EvtRptBehavior](#), [FollowHeadBehavior](#), [HeadLevelBehavior](#), [HeadPointControllerBehavior](#), [SimpleChaseBallBehavior](#), [SoundTestBehavior](#), [StareAtBallBehavior](#), [ToggleHeadLightBehavior](#), [WalkControllerBehavior](#), [WalkToTargetMachine](#), [WorldModel2Behavior](#), [WorldModel2Behavior::WalkNode](#), [WorldModel2Behavior::GawkNode](#), [WorldModel2Behavior::WaitNode](#), [StateNode](#), and [StartupBehavior](#).

Definition at line 21 of file BehaviorBase.h.

References [ReferenceCounter::RemoveReference\(\)](#), and [started](#).

7.26.3.3 std::string BehaviorBase::getClassDescription () [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented in [Controller](#), [Aibo3DControllerBehavior](#), [Aibo3DMonitorBehavior](#), [AutoGetupBehavior](#), [BanditMachine](#), [BatteryMonitorBehavior](#), [CameraBehavior](#), [ChaseBallBehavior](#), [DriveMeBehavior](#), [DumbWM2Behavior](#), [EStopControllerBehavior](#), [EvtRptBehavior](#), [FollowHeadBehavior](#), [HeadLevelBehavior](#), [HeadPointControllerBehavior](#), [SoundTestBehavior](#), [StareAtBallBehavior](#), [ToggleHeadLightBehavior](#), [WalkControllerBehavior](#), [WalkToTargetMachine](#), [WorldModel2Behavior](#), and [StartupBehavior](#).

Definition at line 30 of file BehaviorBase.h.

7.26.3.4 virtual std::string BehaviorBase::getDescription () const [inline, virtual]

Gives a short description of what this particular instantiation does (in case a more specific description is needed).

Reimplemented in [StateNode](#).

Definition at line 33 of file BehaviorBase.h.

References [getClassDescription\(\)](#).

7.26.3.5 **virtual std::string BehaviorBase::getName () const** [pure virtual]

Identifies the behavior in menus and such.

Implemented in [Controller](#), [FreeMemReportControl](#), [Aibo3DControllerBehavior](#), [Aibo3DMonitorBehavior](#), [AutoGetupBehavior](#), [BatteryMonitorBehavior](#), [CameraBehavior](#), [ChaseBallBehavior](#), [DriveMeBehavior](#), [DumbWM2Behavior](#), [EStopControllerBehavior](#), [EvtRptBehavior](#), [FollowHeadBehavior](#), [HeadLevelBehavior](#), [HeadPointControllerBehavior](#), [SimpleChaseBallBehavior](#), [SoundTestBehavior](#), [StareAtBallBehavior](#), [ToggleHeadLightBehavior](#), [WalkControllerBehavior](#), [StateNode](#), and [StartupBehavior](#).

7.26.3.6 **virtual bool BehaviorBase::isActive () const** [inline, virtual]

Returns true if the behavior is currently running.

Definition at line 36 of file BehaviorBase.h.

References started.

7.26.3.7 **virtual void BehaviorBase::processEvent (const [EventBase](#) &)** [inline, virtual]

Allows you to get away with not supplying a [processEvent\(\)](#) function for the [EventListener](#) interface. By default, does nothing.

Implements [EventListener](#).

Reimplemented in [Controller](#), [FreeMemReportControl](#), [AutoGetupBehavior](#), [BanditMachine::WaitNode](#), [BatteryMonitorBehavior](#), [CameraBehavior](#), [ChaseBallBehavior](#), [DriveMeBehavior](#), [DumbWM2Behavior](#), [EStopControllerBehavior](#), [EvtRptBehavior](#), [FollowHeadBehavior](#), [HeadLevelBehavior](#), [HeadPointControllerBehavior](#), [SimpleChaseBallBehavior](#), [SoundTestBehavior](#), [StareAtBallBehavior](#), [WalkControllerBehavior](#), [WalkToTargetMachine](#), [WorldModel2Behavior::WalkNode](#), [WorldModel2Behavior::GawkNode](#), [StateNode](#), and [StartupBehavior](#).

Definition at line 23 of file BehaviorBase.h.

7.26.4 Member Data Documentation

7.26.4.1 **bool [BehaviorBase::started](#)** [protected]

true when the behavior is active

Definition at line 39 of file BehaviorBase.h.

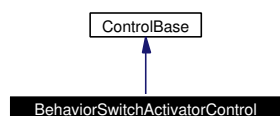
The documentation for this class was generated from the following file:

- [BehaviorBase.h](#)

7.27 BehaviorSwitchActivatorControl Class Reference

```
#include <BehaviorSwitchActivatorControl.h>
```

Inheritance diagram for BehaviorSwitchActivatorControl:



7.27.1 Detailed Description

Upon activation, will tell the specified [BehaviorSwitchControl](#) to start or stop the behavior.

Definition at line 9 of file BehaviorSwitchActivatorControl.h.

Public Types

- enum [Mode_t](#) { [start](#), [stop](#), [toggle](#) }
lets you tell it what action to perform

Public Member Functions

- [BehaviorSwitchActivatorControl](#) (const std::string &n, [BehaviorSwitchControlBase](#) *bscb, [Mode_t](#) m=toggle)
constructor
- virtual [~BehaviorSwitchActivatorControl](#) ()
destructor
- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#) disp_id, [Socket](#) *)
performs the action denoted by [mode](#)
- virtual std::string [getName](#) () const
returns the name of the control
- virtual std::string [getDescription](#) () const
returns a short description of what the control does

Protected Attributes

- [BehaviorSwitchControlBase * behswitch](#)
The behavior switch to activate/deactivate.
- [Mode_t mode](#)
the mode this control is in

Private Member Functions

- [BehaviorSwitchActivatorControl](#) (const [BehaviorSwitchActivatorControl](#) &)
don't copy this class
- [BehaviorSwitchActivatorControl](#) operator= (const [BehaviorSwitchActivatorControl](#) &)
don't assign this class

7.27.2 Member Enumeration Documentation

7.27.2.1 enum [BehaviorSwitchActivatorControl::Mode_t](#)

lets you tell it what action to perform

Enumeration values:

start
stop
toggle

Definition at line 12 of file BehaviorSwitchActivatorControl.h.

7.27.3 Constructor & Destructor Documentation

7.27.3.1 [BehaviorSwitchActivatorControl::BehaviorSwitchActivatorControl](#) (const std::string & *n*, [BehaviorSwitchControlBase](#) * *bscb*, [Mode_t](#) *m* = **toggle**) [[inline](#)]

constructor

Definition at line 15 of file BehaviorSwitchActivatorControl.h.

References [behswitch](#), and [mode](#).

7.27.3.2 virtual BehaviorSwitchActivatorControl::~~BehaviorSwitchActivatorControl() [inline, virtual]

destructor

Definition at line 18 of file BehaviorSwitchActivatorControl.h.

7.27.3.3 BehaviorSwitchActivatorControl::BehaviorSwitchActivatorControl(const BehaviorSwitchActivatorControl &) [private]

don't copy this class

7.27.4 Member Function Documentation

7.27.4.1 virtual ControlBase* BehaviorSwitchActivatorControl::activate(MotionManager::MC_ID disp_id, Socket*) [inline, virtual]

performs the action denoted by [mode](#)

Reimplemented from [ControlBase](#).

Definition at line 21 of file BehaviorSwitchActivatorControl.h.

References [behswitch](#), [ERS210Info::FaceLEDMask](#), [MotionManager::invalid_MC_ID](#), [MMAccessor< MC_t >::mc\(\)](#), [mode](#), [BehaviorSwitchControlBase::start\(\)](#), [start](#), [BehaviorSwitchControlBase::stop\(\)](#), [stop](#), [BehaviorSwitchControlBase::toggle\(\)](#), and [toggle](#).

7.27.4.2 virtual std::string BehaviorSwitchActivatorControl::getDescription() const [inline, virtual]

returns a short description of what the control does

Reimplemented from [ControlBase](#).

Definition at line 41 of file BehaviorSwitchActivatorControl.h.

References [behswitch](#), and [ControlBase::getDescription\(\)](#).

7.27.4.3 virtual std::string BehaviorSwitchActivatorControl::getName() const [inline, virtual]

returns the name of the control

Reimplemented from [ControlBase](#).

Definition at line 40 of file BehaviorSwitchActivatorControl.h.

References behswitch, and ControlBase::getName().

7.27.4.4 BehaviorSwitchActivatorControl BehaviorSwitchActivatorControl::operator= (const BehaviorSwitchActivatorControl &)
[private]

don't assign this class

7.27.5 Member Data Documentation

7.27.5.1 BehaviorSwitchControlBase* BehaviorSwitchActivatorControl::behswitch [protected]

The behavior switch to activate/deactivate.

Definition at line 44 of file BehaviorSwitchActivatorControl.h.

7.27.5.2 Mode_t BehaviorSwitchActivatorControl::mode [protected]

the mode this control is in

Definition at line 45 of file BehaviorSwitchActivatorControl.h.

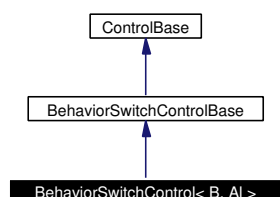
The documentation for this class was generated from the following file:

- [BehaviorSwitchActivatorControl.h](#)

7.28 BehaviorSwitchControl< B, AI > Class Template Reference

```
#include <BehaviorSwitchControl.h>
```

Inheritance diagram for BehaviorSwitchControl< B, AI >:



7.28.1 Detailed Description

```
template<class B, class AI = Factory< B >> class BehaviorSwitchControl< B, AI >
```

Allows proper switching between major behaviors, calling DoStart and DoStop.

Definition at line 75 of file BehaviorSwitchControl.h.

Public Member Functions

- [BehaviorSwitchControl](#) (const std::string &n, bool retain=false)
constructor, can use this to toggle a single behavior on and off
- [BehaviorSwitchControl](#) (const std::string &n, B *beh, BehaviorGroup *bg=NULL)
constructor, if you want to use an already constructed behavior, can pass NULL for behavior group
- [BehaviorSwitchControl](#) (const std::string &n, BehaviorGroup *bg, bool retain=false)
constructor, needs to know what group its in and whether to retain its behavior
- virtual [~BehaviorSwitchControl](#) ()
destructor
- virtual [BehaviorSwitchControl](#)< B, AI > * [start](#) ()

activates the behavior, handy for making start-up behaviors that you can turn off again with the [Controller](#)

- virtual [BehaviorSwitchControl](#)< B, AI > * [stop](#) ()
stops the behavior
- virtual [BehaviorSwitchControl](#)< B, AI > * [toggle](#) ()
toggles the behavior
- virtual std::string [getName](#) () const
adds a status to the name: - if in memory, # if running
- virtual std::string [getDescription](#) () const
returns a short description of what the control does

Protected Member Functions

- virtual void [stopother](#) ()
Stops the "other" guy's behavior - if ::behgrp is NULL, stops ourselves.
- virtual void [startmine](#) ()
Starts our behavior.
- virtual bool [isRunning](#) () const
Returns true if the associated behavior is running.
- virtual bool [isValid](#) () const
Returns true if mybeh is pointing to a valid object.

Protected Attributes

- B * [mybeh](#)
used to store the behavior. If retained and non-NULL, will be valid. However, if not retained, only valid if equals behgrp->curBehavior

Private Member Functions

- [BehaviorSwitchControl](#) (const [BehaviorSwitchControl](#) &)
shouldn't call this

- `BehaviorSwitchControl operator=` (const `BehaviorSwitchControl` &)

shouldn't call this

7.28.2 Constructor & Destructor Documentation

7.28.2.1 `template<class B, class AI = Factory< B >> BehaviorSwitchControl< B, AI >::BehaviorSwitchControl` (const std::string & *n*, bool *retain* = false) [inline]

constructor, can use this to toggle a single behavior on and off

Definition at line 78 of file BehaviorSwitchControl.h.

References BehaviorSwitchControl< B, AI >::mybeh.

7.28.2.2 `template<class B, class AI = Factory< B >> BehaviorSwitchControl< B, AI >::BehaviorSwitchControl` (const std::string & *n*, B * *beh*, BehaviorGroup * *bg* = NULL) [inline]

constructor, if you want to use an already constructed behavior, can pass NULL for behavior group

Definition at line 81 of file BehaviorSwitchControl.h.

References BehaviorSwitchControl< B, AI >::mybeh, and BehaviorSwitchControl-Base::retained.

7.28.2.3 `template<class B, class AI = Factory< B >> BehaviorSwitchControl< B, AI >::BehaviorSwitchControl` (const std::string & *n*, BehaviorGroup * *bg*, bool *retain* = false) [inline]

constructor, needs to know what group its in and whether to retain its behavior

Definition at line 88 of file BehaviorSwitchControl.h.

References BehaviorSwitchControl< B, AI >::mybeh.

7.28.2.4 `template<class B, class AI = Factory< B >> virtual BehaviorSwitchControl< B, AI >::~BehaviorSwitchControl` () [inline, virtual]

destructor

Definition at line 92 of file BehaviorSwitchControl.h.

References BehaviorSwitchControl< B, AI >::mybeh, BehaviorSwitchControlBase::retained, and BehaviorSwitchControl< B, AI >::stop().

7.28.2.5 `template<class B, class AI = Factory< B >> BehaviorSwitchControl< B, AI >::BehaviorSwitchControl (const BehaviorSwitchControl< B, AI > &) [private]`

shouldn't call this

7.28.3 Member Function Documentation

7.28.3.1 `template<class B, class AI = Factory< B >> virtual std::string BehaviorSwitchControl< B, AI >::getDescription () const [inline, virtual]`

returns a short description of what the control does

Reimplemented from [ControlBase](#).

Definition at line 111 of file BehaviorSwitchControl.h.

References BehaviorSwitchControl< B, AI >::isValid(), and BehaviorSwitchControl< B, AI >::mybeh.

7.28.3.2 `template<class B, class AI = Factory< B >> virtual std::string BehaviorSwitchControl< B, AI >::getName () const [inline, virtual]`

adds a status to the name: - if in memory, # if running

Reimplemented from [ControlBase](#).

Definition at line 105 of file BehaviorSwitchControl.h.

References ControlBase::getName(), BehaviorSwitchControl< B, AI >::isValid(), and BehaviorSwitchControl< B, AI >::mybeh.

7.28.3.3 `template<class B, class AI = Factory< B >> virtual bool BehaviorSwitchControl< B, AI >::isRunning () const [inline, protected, virtual]`

Returns true if the associated behavior is running.

Definition at line 156 of file BehaviorSwitchControl.h.

References BehaviorSwitchControlBase::behgrp, and BehaviorSwitchControl< B, AI >::mybeh.

7.28.3.4 `template<class B, class AI = Factory< B >> virtual bool
BehaviorSwitchControl< B, AI >::isValid () const [inline,
 protected, virtual]`

Returns true if mybeh is pointing to a valid object.

Definition at line 167 of file BehaviorSwitchControl.h.

References [BehaviorSwitchControl< B, AI >::isRunning\(\)](#), and [BehaviorSwitchControlBase::retained](#).

7.28.3.5 `template<class B, class AI = Factory< B >> BehaviorSwitchControl
BehaviorSwitchControl< B, AI >::operator= (const
BehaviorSwitchControl< B, AI > &) [private]`

shouldn't call this

7.28.3.6 `template<class B, class AI = Factory< B >> virtual
BehaviorSwitchControl<B,AI>* BehaviorSwitchControl< B, AI
 >::start () [inline, virtual]`

activates the behavior, handy for making start-up behaviors that you can turn off again with the [Controller](#)

If you start twice without stopping (ie it's already running), shouldn't do anything

Implements [BehaviorSwitchControlBase](#).

Definition at line 98 of file BehaviorSwitchControl.h.

References [BehaviorSwitchControl< B, AI >::isRunning\(\)](#), [BehaviorSwitchControl< B, AI >::startmine\(\)](#), and [BehaviorSwitchControl< B, AI >::stopother\(\)](#).

7.28.3.7 `template<class B, class AI = Factory< B >> virtual void
BehaviorSwitchControl< B, AI >::startmine () [inline,
 protected, virtual]`

Starts our behavior.

Definition at line 139 of file BehaviorSwitchControl.h.

References [BehaviorSwitchControlBase::behgrp](#), [BehaviorSwitchControl< B, AI >::mybeh](#), and [BehaviorSwitchControlBase::retained](#).

7.28.3.8 `template<class B, class AI = Factory< B >> virtual
 BehaviorSwitchControl<B,AI>* BehaviorSwitchControl< B, AI
 >::stop () [inline, virtual]`

stops the behavior

Implements [BehaviorSwitchControlBase](#).

Definition at line 100 of file BehaviorSwitchControl.h.

References [BehaviorSwitchControl< B, AI >::isRunning\(\)](#), and [BehaviorSwitchControl< B, AI >::stopother\(\)](#).

7.28.3.9 `template<class B, class AI = Factory< B >> virtual void
 BehaviorSwitchControl< B, AI >::stopother () [inline,
 protected, virtual]`

Stops the "other" guy's behavior - if ::begrp is NULL, stops ourselves.

Definition at line 122 of file BehaviorSwitchControl.h.

References [ASSERT](#), [BehaviorSwitchControlBase::begrp](#), [BehaviorSwitchControl< B, AI >::mybeh](#), and [BehaviorSwitchControlBase::retained](#).

7.28.3.10 `template<class B, class AI = Factory< B >> virtual
 BehaviorSwitchControl<B,AI>* BehaviorSwitchControl< B, AI
 >::toggle () [inline, virtual]`

toggles the behavior

Implements [BehaviorSwitchControlBase](#).

Definition at line 102 of file BehaviorSwitchControl.h.

References [BehaviorSwitchControl< B, AI >::isRunning\(\)](#), [BehaviorSwitchControl< B, AI >::startmine\(\)](#), and [BehaviorSwitchControl< B, AI >::stopother\(\)](#).

7.28.4 Member Data Documentation

7.28.4.1 `template<class B, class AI = Factory< B >> B*
 BehaviorSwitchControl< B, AI >::mybeh [protected]`

used to store the behavior. If retained and non-NULL, will be valid. However, if not retained, only valid if equals begrp->curBehavior

Definition at line 173 of file BehaviorSwitchControl.h.

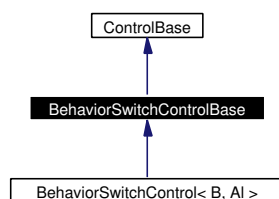
The documentation for this class was generated from the following file:

- [BehaviorSwitchControl.h](#)

7.29 BehaviorSwitchControlBase Class Reference

```
#include <BehaviorSwitchControl.h>
```

Inheritance diagram for BehaviorSwitchControlBase:



7.29.1 Detailed Description

Holds some utility classes and functions for [BehaviorSwitchControl](#) which shouldn't be stored in a templated class.

Definition at line 12 of file BehaviorSwitchControl.h.

Public Member Functions

- [BehaviorSwitchControlBase](#) (const std::string &n, [BehaviorGroup](#) *bg, bool retain)
constructor
- virtual [~BehaviorSwitchControlBase](#) ()
destructor
- virtual [BehaviorSwitchControlBase](#) * [start](#) ()=0
activates the behavior; handy for making start-up behaviors that you can turn off again with the [Controller](#)
- virtual [BehaviorSwitchControlBase](#) * [stop](#) ()=0
stops the behavior
- virtual [BehaviorSwitchControlBase](#) * [toggle](#) ()=0
toggles the behavior
- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#) display, [Socket](#) *gui)
tells the current behavior (if there is one) to stop then loads its own

Protected Attributes

- bool [retained](#)
true if the behavior should be generated once and retained after DoStop. Otherwise, a new one is generated each time it is started
- [BehaviorGroup](#) * [behgrp](#)
the behavior group this belongs to. Uses this to track the "current" behavior

Private Member Functions

- [BehaviorSwitchControlBase](#) (const [BehaviorSwitchControlBase](#) &)
shouldn't copy these
- [BehaviorSwitchControlBase](#) operator= (const [BehaviorSwitchControlBase](#) &)
shouldn't assign these

7.29.2 Constructor & Destructor Documentation

7.29.2.1 [BehaviorSwitchControlBase::BehaviorSwitchControlBase](#) (const std::string & n, [BehaviorGroup](#) * bg, bool retain) [inline]

constructor

Definition at line 32 of file BehaviorSwitchControl.h.

References [ReferenceCounter::AddReference\(\)](#), [behgrp](#), and [retained](#).

7.29.2.2 [virtual BehaviorSwitchControlBase::~~BehaviorSwitchControlBase](#) () [inline, virtual]

destructor

Definition at line 40 of file BehaviorSwitchControl.h.

References [behgrp](#), and [ReferenceCounter::RemoveReference\(\)](#).

7.29.2.3 [BehaviorSwitchControlBase::BehaviorSwitchControlBase](#) (const [BehaviorSwitchControlBase](#) &) [private]

shouldn't copy these

7.29.3 Member Function Documentation

7.29.3.1 `virtual ControlBase* BehaviorSwitchControlBase::activate(MotionManager::MC_ID display, Socket * gui)` [inline, virtual]

tells the current behavior (if there is one) to stop then loads its own

Returns:

NULL unless there are submenus

Reimplemented from [ControlBase](#).

Definition at line 53 of file BehaviorSwitchControl.h.

References [ControlBase::activate\(\)](#), [ControlBase::slotsSize\(\)](#), and [toggle\(\)](#).

7.29.3.2 `BehaviorSwitchControlBase BehaviorSwitchControlBase::operator=(const BehaviorSwitchControlBase &)` [private]

shouldn't assign these

7.29.3.3 `virtual BehaviorSwitchControlBase* BehaviorSwitchControlBase::start()` [pure virtual]

activates the behavior, handy for making start-up behaviors that you can turn off again with the [Controller](#)

If you start twice without stopping (ie it's already running), shouldn't do anything

Implemented in [BehaviorSwitchControl< B, AI >](#).

7.29.3.4 `virtual BehaviorSwitchControlBase* BehaviorSwitchControlBase::stop()` [pure virtual]

stops the behavior

Implemented in [BehaviorSwitchControl< B, AI >](#).

7.29.3.5 `virtual BehaviorSwitchControlBase* BehaviorSwitchControlBase::toggle()` [pure virtual]

toggles the behavior

Implemented in [BehaviorSwitchControl< B, AI >](#).

7.29.4 Member Data Documentation

7.29.4.1 [BehaviorGroup*](#) [BehaviorSwitchControlBase::behgrp](#) [protected]

the behavior group this belongs to. Uses this to track the "current" behavior

Definition at line 63 of file BehaviorSwitchControl.h.

7.29.4.2 **bool** [BehaviorSwitchControlBase::retained](#) [protected]

true if the behavior should be generated once and retained after DoStop. Otherwise, a new one is generated each time it is started

Definition at line 62 of file BehaviorSwitchControl.h.

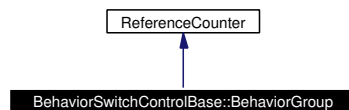
The documentation for this class was generated from the following file:

- [BehaviorSwitchControl.h](#)

7.30 BehaviorSwitchControlBase::BehaviorGroup Class Reference

```
#include <BehaviorSwitchControl.h>
```

Inheritance diagram for BehaviorSwitchControlBase::BehaviorGroup:



7.30.1 Detailed Description

A simple utility class to allow the BehaviorSwitchControl's to be able to deactivate the current behavior when a new one becomes active.

Most behaviors are either major actions which you'll only want one of active at a time, or else their background monitors of some sort, that can run in different combinations. Think radio buttons vs. checkboxes. This will help you implement the "radio button" style... just assign all the behaviors to the same group, they will automatically use it to turn the previous behavior off when a new one becomes active.

Pass NULL instead of one of these to get checkbox-style.

Definition at line 21 of file BehaviorSwitchControl.h.

Public Member Functions

- [BehaviorGroup](#) ()
constructor
- [~BehaviorGroup](#) ()
destructor, will stop the current behavior if it was a one-shot

Public Attributes

- [BehaviorBase](#) * [curBehavior](#)
pointer to current behavior

Private Member Functions

- [BehaviorGroup](#) (const [BehaviorGroup](#) &)
shouldn't be called
- [BehaviorGroup](#) operator= (const [BehaviorGroup](#) &)
shouldn't be called

7.30.2 Constructor & Destructor Documentation

7.30.2.1 BehaviorSwitchControlBase::BehaviorGroup::BehaviorGroup () [inline]

constructor

Definition at line 23 of file BehaviorSwitchControl.h.

References [curBehavior](#).

7.30.2.2 BehaviorSwitchControlBase::BehaviorGroup::~~BehaviorGroup () [inline]

destructor, will stop the current behavior if it was a one-shot

Definition at line 24 of file BehaviorSwitchControl.h.

References [curBehavior](#), and [BehaviorBase::DoStop\(\)](#).

7.30.2.3 BehaviorSwitchControlBase::BehaviorGroup::BehaviorGroup (const [BehaviorGroup](#) &) [private]

shouldn't be called

7.30.3 Member Function Documentation

7.30.3.1 [BehaviorGroup](#) BehaviorSwitchControlBase::BehaviorGroup::operator= (const [BehaviorGroup](#) &) [private]

shouldn't be called

7.30.4 Member Data Documentation

7.30.4.1 [BehaviorBase*](#) [BehaviorSwitchControlBase::BehaviorGroup::currentBehavior](#)

pointer to current behavior

Definition at line 25 of file BehaviorSwitchControl.h.

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

- [BehaviorSwitchControl.h](#)

7.31 BodyPosition Struct Reference

```
#include <Kinematics.h>
```

7.31.1 Detailed Description

holds the current location of the body, as a delta from when walking started

Todo

get rid of this

Definition at line 67 of file Kinematics.h.

Public Member Functions

- [BodyPosition \(\)](#)

constructor

Public Attributes

- [vector3d loc](#)

position of the center of the body

- [vector3d angle](#)

angle of the center of the body

7.31.2 Constructor & Destructor Documentation

7.31.2.1 BodyPosition::BodyPosition () [inline]

constructor

Definition at line 69 of file Kinematics.h.

References [angle](#), and [loc](#).

7.31.3 Member Data Documentation

7.31.3.1 [vector3d BodyPosition::angle](#)

angle of the center of the body

Definition at line 71 of file Kinematics.h.

7.31.3.2 [vector3d BodyPosition::loc](#)

position of the center of the body

Definition at line 70 of file Kinematics.h.

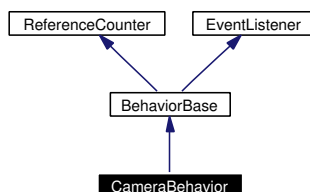
The documentation for this struct was generated from the following file:

- [Kinematics.h](#)

7.32 CameraBehavior Class Reference

```
#include <CameraBehavior.h>
```

Inheritance diagram for CameraBehavior:



7.32.1 Detailed Description

Will take images and write to log file.

Press the head button to take a picture, back button to write to memory stick. This isn't necessarily up to date, but is included as sample code. We should have a way to save pictures to memstick instead of relying solely on having wireless to transmit them over.

Definition at line 19 of file CameraBehavior.h.

Public Member Functions

- [CameraBehavior](#) ()
just sets up the variables
- virtual [~CameraBehavior](#) ()
calls [DoStop\(\)](#) if [isActive\(\)](#)
- virtual void [DoStart](#) ()
Register for events and creates and adds two motion commands - a walker and a tail wag.
- virtual void [DoStop](#) ()
Removes its two motion commands.
- virtual void [processEvent](#) (const [EventBase](#) &e)
Handles event processing.
- virtual std::string [getName](#) () const

returns name of behavior

Static Public Member Functions

- virtual std::string [getClassDescription](#) ()

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Member Functions

- double [RAD](#) (double x)

converts x degrees to radians

Protected Attributes

- const [EventBase](#) [camera_click](#)

event mask for taking a picture (head button)

- const [EventBase](#) [sensor_update](#)

event mask for sensor update

- [MotionManager::MC_ID](#) [headpointer_id](#)

MC_ID for head pointer.

- [MotionManager::MC_ID](#) [tailwag_id](#)

MC_ID for tail wag.

- [MotionManager::MC_ID](#) [led_id](#)

MC_ID for leds.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 [CameraBehavior::CameraBehavior](#) () [inline]

just sets up the variables

Definition at line 23 of file [CameraBehavior.h](#).

References `camera_click`, `ERS210Info::HeadFrButOffset`, `headpointer_id`, `led_id`, `sensor_update`, `tailwag_id`, and `SensorSourceID::UpdatedSID`.

7.32.2.2 **virtual CameraBehavior::~~CameraBehavior ()** [inline, virtual]

calls `DoStop()` if `isActive()`

Definition at line 35 of file `CameraBehavior.h`.

References `DoStop()`, and `BehaviorBase::isActive()`.

7.32.3 Member Function Documentation

7.32.3.1 **virtual void CameraBehavior::DoStart ()** [inline, virtual]

Register for events and creates and adds two motion commands - a walker and a tail wag.

Reimplemented from `BehaviorBase`.

Definition at line 38 of file `CameraBehavior.h`.

References `EventRouter::addListener()`, `MotionManager::addMotion()`, `camera_click`, `BehaviorBase::DoStart()`, `erouter`, `headpointer_id`, `MotionManager::invalid_MC_ID`, `led_id`, `motman`, `ERS210Info::PanOffset`, `processEvent()`, `sensor_update`, `ERS210Info::TailOffset`, `tailwag_id`, and `ERS210Info::TiltOffset`.

7.32.3.2 **virtual void CameraBehavior::DoStop ()** [inline, virtual]

Removes its two motion commands.

Reimplemented from `BehaviorBase`.

Definition at line 58 of file `CameraBehavior.h`.

References `BehaviorBase::DoStop()`, `erouter`, `EventRouter::forgetListener()`, `headpointer_id`, `MotionManager::invalid_MC_ID`, `led_id`, `motman`, `ERS210Info::PanOffset`, `MotionManager::removeMotion()`, `ERS210Info::TailOffset`, `tailwag_id`, and `ERS210Info::TiltOffset`.

7.32.3.3 **virtual std::string CameraBehavior::getClassDescription ()** [inline, static, virtual]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 111 of file CameraBehavior.h.

7.32.3.4 `virtual std::string CameraBehavior::getName () const` `[inline, virtual]`

returns name of behavior

Implements [BehaviorBase](#).

Definition at line 110 of file CameraBehavior.h.

7.32.3.5 `virtual void CameraBehavior::processEvent (const EventBase & e)` `[inline, virtual]`

Handles event processing.

After every sensor update, set head in direction of tail

Reimplemented from [BehaviorBase](#).

Definition at line 86 of file CameraBehavior.h.

References `camera_click`, `EventBase::equalOrLongerThan()`, `ERS210Info::FaceLEDMask`, `ERS210Info::HeadOffset`, `headpointer_id`, `led_id`, `MMAccessor< MC_t >::mc()`, `ERS210Info::outputRanges`, `WorldState::outputs`, `ERS210Info::PanOffset`, `RAD()`, `sensor_update`, `state`, `ERS210Info::TailOffset`, and `ERS210Info::TiltOffset`.

7.32.3.6 `double CameraBehavior::RAD (double x)` `[inline, protected]`

converts x degrees to radians

Definition at line 113 of file CameraBehavior.h.

7.32.4 Member Data Documentation

7.32.4.1 `const EventBase CameraBehavior::camera_click` `[protected]`

event mask for taking a picture (head button)

Definition at line 115 of file CameraBehavior.h.

7.32.4.2 `MotionManager::MC_ID CameraBehavior::headpointer_id` `[protected]`

MC_ID for head pointer.

Definition at line 117 of file CameraBehavior.h.

7.32.4.3 [MotionManager::MC_ID CameraBehavior::led_id](#) [protected]

MC_ID for leds.

Definition at line 119 of file CameraBehavior.h.

7.32.4.4 **const** [EventBase CameraBehavior::sensor_update](#) [protected]

event mask for sensor update

Definition at line 116 of file CameraBehavior.h.

7.32.4.5 [MotionManager::MC_ID CameraBehavior::tailwag_id](#) [protected]

MC_ID for tail wag.

Definition at line 118 of file CameraBehavior.h.

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

- [CameraBehavior.h](#)

7.33 char_traits< T > Class Template Reference

```
#include <ionetstream.h>
```

```
template<class T> class char_traits< T >
```

Public Types

- typedef T [char_type](#)
- typedef int [int_type](#)
- typedef T * [pos_type](#)
- typedef unsigned int [off_type](#)

Static Public Member Functions

- void [copy](#) ([pos_type](#) dst, [pos_type](#) src, [off_type](#) size)
- void [move](#) ([pos_type](#) dst, [pos_type](#) src, [off_type](#) size)
- int [to_int_type](#) (T c)
- int [eof](#) ()

7.33.1 Member Typedef Documentation

7.33.1.1 template<class T> typedef T [char_traits](#)< T >::[char_type](#)

Definition at line 211 of file ionetstream.h.

7.33.1.2 template<class T> typedef int [char_traits](#)< T >::[int_type](#)

Definition at line 212 of file ionetstream.h.

7.33.1.3 template<class T> typedef unsigned int [char_traits](#)< T >::[off_type](#)

Definition at line 214 of file ionetstream.h.

7.33.1.4 template<class T> typedef T* [char_traits](#)< T >::[pos_type](#)

Definition at line 213 of file ionetstream.h.

7.33.2 Member Function Documentation

7.33.2.1 `template<class T> void char_traits< T >::copy (pos_type dst, pos_type src, off_type size)` [`inline`, `static`]

Definition at line 215 of file `ionetstream.h`.

7.33.2.2 `template<class T> int char_traits< T >::eof ()` [`inline`, `static`]

Definition at line 222 of file `ionetstream.h`.

7.33.2.3 `template<class T> void char_traits< T >::move (pos_type dst, pos_type src, off_type size)` [`inline`, `static`]

Definition at line 218 of file `ionetstream.h`.

7.33.2.4 `template<class T> int char_traits< T >::to_int_type (T c)` [`inline`, `static`]

Definition at line 221 of file `ionetstream.h`.

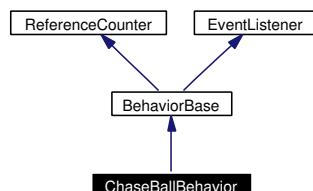
The documentation for this class was generated from the following file:

- [ionetstream.h](#)

7.34 ChaseBallBehavior Class Reference

```
#include <ChaseBallBehavior.h>
```

Inheritance diagram for ChaseBallBehavior:



7.34.1 Detailed Description

A simple behavior to chase after any objects seen by the vision system.

Definition at line 9 of file ChaseBallBehavior.h.

Public Member Functions

- [ChaseBallBehavior](#) ()
constructor
- virtual [~ChaseBallBehavior](#) ()
destructor
- virtual void [DoStart](#) ()
adds a headpointer and a walker, and a listens for vision events
- virtual void [DoStop](#) ()
removes motion commands and stops listening
- virtual void [processEvent](#) (const [EventBase](#) &event)
sets the head to point at the object and sets the body to move where the head points
- virtual std::string [getName](#) () const
returns name of behavior

Static Public Member Functions

- `std::string getClassDescription ()`
Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- `MotionManager::MC_ID headpointer_id`
a [HeadPointerMC](#) object
- `MotionManager::MC_ID walker_id`
a [WalkMC](#) object

7.34.2 Constructor & Destructor Documentation

7.34.2.1 `ChaseBallBehavior::ChaseBallBehavior ()` [inline]

constructor

Definition at line 12 of file `ChaseBallBehavior.h`.

References `headpointer_id`, and `walker_id`.

7.34.2.2 `virtual ChaseBallBehavior::~ChaseBallBehavior ()` [inline, virtual]

destructor

Definition at line 16 of file `ChaseBallBehavior.h`.

7.34.3 Member Function Documentation

7.34.3.1 `void ChaseBallBehavior::DoStart ()` [virtual]

adds a headpointer and a walker, and a listens for vision events

Reimplemented from [BehaviorBase](#).

Definition at line 12 of file `ChaseBallBehavior.cc`.

References `EventRouter::addListener()`, `MotionManager::addMotion()`, `BehaviorBase::DoStart()`, `Vision::enableEvents()`, `erouter`, `headpointer_id`, `motman`, `vision`, `EventBase::visionEGID`, and `walker_id`.

7.34.3.2 void ChaseBallBehavior::DoStop () [virtual]

removes motion commands and stops listening

Reimplemented from [BehaviorBase](#).

Definition at line 21 of file ChaseBallBehavior.cc.

References [Vision::disableEvents\(\)](#), [BehaviorBase::DoStop\(\)](#), [erouter](#), [EventRouter::forgetListener\(\)](#), [headpointer_id](#), [motman](#), [MotionManager::removeMotion\(\)](#), [vision](#), and [walker_id](#).

7.34.3.3 std::string ChaseBallBehavior::getClassDescription () [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 28 of file ChaseBallBehavior.h.

7.34.3.4 virtual std::string ChaseBallBehavior::getName () const [inline, virtual]

returns name of behavior

Implements [BehaviorBase](#).

Definition at line 27 of file ChaseBallBehavior.h.

7.34.3.5 void ChaseBallBehavior::processEvent (const [EventBase](#) & event) [virtual]

sets the head to point at the object and sets the body to move where the head points

Reimplemented from [BehaviorBase](#).

Definition at line 31 of file ChaseBallBehavior.cc.

References [MotionManager::checkinMotion\(\)](#), [MotionManager::checkoutMotion\(\)](#), [DtoR\(\)](#), [EventBase::getGeneratorID\(\)](#), [EventBase::getTypeID\(\)](#), [ERS210Info::HeadOffset](#), [headpointer_id](#), [motman](#), [WorldState::outputs](#), [ERS210Info::PanOffset](#), [HeadPointerMC::setJoints\(\)](#), [WalkMC::setTargetVelocity\(\)](#), [state](#), [EventBase::statusETID](#), [ERS210Info::TiltOffset](#), [EventBase::visionEGID](#), and [walker_id](#).

7.34.4 Member Data Documentation

7.34.4.1 [MotionManager::MC_ID](#) [ChaseBallBehavior::headpointer_id](#) [protected]

a [HeadPointerMC](#) object

Definition at line 31 of file ChaseBallBehavior.h.

7.34.4.2 [MotionManager::MC_ID](#) [ChaseBallBehavior::walker_id](#) [protected]

a [WalkMC](#) object

Definition at line 32 of file ChaseBallBehavior.h.

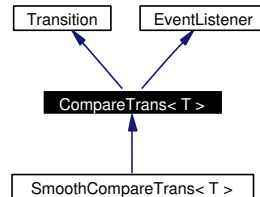
The documentation for this class was generated from the following files:

- [ChaseBallBehavior.h](#)
- [ChaseBallBehavior.cc](#)

7.35 CompareTrans< T > Class Template Reference

```
#include <CompareTrans.h>
```

Inheritance diagram for CompareTrans< T >:



7.35.1 Detailed Description

template<class T> class CompareTrans< T >

causes a transition if a value (through a pointer) goes above a given value

You will need to specify an event mask which will be listened for. This event will then be listened for - each time it is received, CompareTrans will check the values for possible activation.

For example, if you want to transition when the IR sensor goes below, say 200, pass &state->sensors[IRDistOffset], [CompareTrans::LT](#), 200, and EventBase(EventBase::sensorEGID, SensorSourceID::UpdatedSID, EventBase::statusETID) as the polling event. Or a timer event to just check at a certain interval.

If you pass a class as the templated type, only requires that < operator is defined for comparing inequality, == for equality, and a copy constructor (CompareTrans holds a protected copy of the value)

Passing NULL as the value to monitor will cause a transition on the first event received

Definition at line 25 of file CompareTrans.h.

Public Types

- enum [Test.t](#) {
[LT](#), [GT](#), [LTE](#), [GTE](#),
[EQ](#), [NE](#) }

use these values to sepecify what kind of comparison should be made to test for activation

Public Member Functions

- [CompareTrans](#) ([StateNode](#) *source, [StateNode](#) *destination, const T *monitor, [Test_t](#) test, const T &value, const [EventBase](#) &poll)
constructor, see class notes for information
- virtual void [enable](#) ()
starts listening
- virtual void [disable](#) ()
stops listening
- virtual void [processEvent](#) (const [EventBase](#) &)
don't care about the event, just a pulse to check the values

Protected Attributes

- const T * [mon](#)
address of value to monitor
- [Test_t](#) [tst](#)
test to make
- T [val](#)
value to compare against
- [EventBase](#) [poller](#)
event to listen to, when it comes, compare the values

Private Member Functions

- [CompareTrans](#) (const [CompareTrans](#) &node)
don't call this
- [CompareTrans](#) operator= (const [CompareTrans](#) &node)
don't call this

7.35.2 Member Enumeration Documentation

7.35.2.1 `template<class T> enum CompareTrans::Test_t`

use these values to sepecify what kind of comparison should be made to test for activation

Enumeration values:

- LT** less than
- GT** greater than
- LTE** less than or equal
- GTE** greater than or equal
- EQ** equal
- NE** not equal

Definition at line 28 of file CompareTrans.h.

7.35.3 Constructor & Destructor Documentation

7.35.3.1 `template<class T> CompareTrans< T >::CompareTrans (StateNode * source, StateNode * destination, const T * monitor, Test_t test, const T & value, const EventBase & poll) [inline]`

constructor, see class notes for information

Definition at line 38 of file CompareTrans.h.

References `CompareTrans< T >::mon`, `CompareTrans< T >::poller`, `CompareTrans< T >::tst`, and `CompareTrans< T >::val`.

7.35.3.2 `template<class T> CompareTrans< T >::CompareTrans (const CompareTrans< T > & node) [private]`

don't call this

7.35.4 Member Function Documentation

7.35.4.1 `template<class T> virtual void CompareTrans< T >::disable () [inline, virtual]`

stops listening

Implements [Transition](#).

Definition at line 46 of file CompareTrans.h.

References `erouter`, and `EventRouter::forgetListener()`.

7.35.4.2 `template<class T> virtual void CompareTrans< T >::enable ()`
`[inline, virtual]`

starts listening

Implements [Transition](#).

Definition at line 43 of file CompareTrans.h.

References `EventRouter::addListener()`, `erouter`, and `CompareTrans< T >::poller`.

7.35.4.3 `template<class T> CompareTrans CompareTrans< T >::operator=`
`(const CompareTrans< T > & node) [private]`

don't call this

7.35.4.4 `template<class T> virtual void CompareTrans< T >::processEvent`
`(const EventBase &) [inline, virtual]`

don't care about the event, just a pulse to check the values

Implements [EventListener](#).

Reimplemented in [SmoothCompareTrans< T >](#).

Definition at line 49 of file CompareTrans.h.

References `Transition::activate()`, `CompareTrans< T >::EQ`, `CompareTrans< T >::GT`, `CompareTrans< T >::GTE`, `CompareTrans< T >::LT`, `CompareTrans< T >::LTE`, `CompareTrans< T >::mon`, `CompareTrans< T >::NE`, `CompareTrans< T >::tst`, and `CompareTrans< T >::val`.

7.35.5 Member Data Documentation

7.35.5.1 `template<class T> const T* CompareTrans< T >::mon`
`[protected]`

address of value to monitor

Definition at line 73 of file CompareTrans.h.

7.35.5.2 `template<class T> EventBase CompareTrans< T >::poller`
[protected]

event to listen to, when it comes, compare the values

Definition at line 76 of file CompareTrans.h.

7.35.5.3 `template<class T> Test.t CompareTrans< T >::tst` [protected]

test to make

Definition at line 74 of file CompareTrans.h.

7.35.5.4 `template<class T> T CompareTrans< T >::val` [protected]

value to compare against

Definition at line 75 of file CompareTrans.h.

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

- [CompareTrans.h](#)

7.36 Config Class Reference

```
#include <Config.h>
```

7.36.1 Detailed Description

provides global access to system configuration information

Definition at line 9 of file Config.h.

Public Types

- enum [section_t](#) {
 [sec_wireless](#) = 0, [sec_vision](#), [sec_main](#), [sec_behaviors](#),
 [sec_controller](#), [sec_motion](#), [sec_worldmodel2](#), [sec_sound](#),
 [sec_invalid](#) }
 section IDs

Public Member Functions

- [Config](#) (const char *filename)
 constructor
- [~Config](#) ()
 destructor
- void [readConfig](#) (const char *filename)
 call this function when it's time to read the configuration file

Public Attributes

- [Config::wireless_config](#) [wireless](#)
 wirless information
- [Config::vision_config](#) [vision](#)
 vision information
- [Config::main_config](#) [main](#)

core functionality information

- [Config::behaviors_config](#) behaviors
placeholder
- [Config::controller_config](#) controller
controller information
- [Config::motion_config](#) motion
motion information
- [Config::worldmodel2_config](#) worldmodel2
world model information
- [Config::sound_config](#) sound
sound information

Static Protected Member Functions

- bool [extractBool](#) (const char *value)
returns bool value corresponding to a value of "t", "f", "true", "false", "y", "n", "yes", "no", or zero/nonzero number

7.36.2 Member Enumeration Documentation

7.36.2.1 enum [Config::section_t](#)

section IDs

Enumeration values:

- sec_wireless** denotes wireless section of config file
- sec_vision** denotes vision section of config file
- sec_main** denotes main section of config file, for misc. settings
- sec_behaviors** denotes behaviors section of config file
- sec_controller** denotes controller section of config file
- sec_motion** denotes motion section of config file
- sec_worldmodel2** denotes worldmodel section of config file
- sec_sound** denotes sound section of config file
- sec_invalid** denotes an invalid section of config file

Definition at line 20 of file Config.h.

7.36.3 Constructor & Destructor Documentation

7.36.3.1 Config::Config (const char * *filename*) [inline]

constructor

Definition at line 12 of file Config.h.

References behaviors, controller, main, motion, readConfig(), sound, vision, wireless, and worldmodel2.

7.36.3.2 Config::~Config () [inline]

destructor

Definition at line 17 of file Config.h.

7.36.4 Member Function Documentation

7.36.4.1 bool Config::extractBool (const char * *value*) [static, protected]

returns bool value corresponding to a *value* of "t", "f", "true", "false", "y", "n", "yes", "no", or zero/nonzero number

Definition at line 166 of file Config.cc.

7.36.4.2 void Config::readConfig (const char * *filename*)

call this function when it's time to read the configuration file

Definition at line 8 of file Config.cc.

References Config::main_config::aibo3d_port, Config::controller_config::cancel_snd, Config::vision_config::colors, Config::main_config::console_port, controller, Config::main_config::debug_level, Config::worldmodel2_config::dm_port, Config::main_config::error_level, Config::motion_config::estop_off_snd, Config::motion_config::estop_on_snd, Config::main_config::estopControl_port, extractBool(), Config::worldmodel2_config::fs_port, Config::vision_config::gain, Config::worldmodel2_config::gm_port, Config::controller_config::gui_port, Config::main_config::headControl_port, Config::worldmodel2_config::hm_port, Config::wireless_config::id, main, motion, Config::controller_config::next_snd, Config::vision_config::obj_port, Config::sound_config::preload, Config::controller_config::prev_snd, Config::vision_config::raw_port, Config::controller_config::read_snd, Config::vision_config::resolution, Config::vision_config::rle_port, Config::sound_config::root, Config::motion_config::root, Config::sound_config::sample_bits, Config::sound-

config::sample_rate, sec_behaviors, sec_controller, sec_invalid, sec_main, sec_motion, sec_sound, sec_vision, sec_wireless, sec_worldmodel2, section_t, Config::controller_config::select_snd, Config::vision_config::shutter_speed, sound, Config::main_config::stderr_port, Config::vision_config::thresh, Config::main_config::use_VT100, Config::main_config::verbose_level, vision, Config::main_config::walkControl_port, Config::vision_config::white_balance, wireless, worldmodel2, Config::main_config::wsjoints_port, and Config::main_config::wspids_port.

7.36.5 Member Data Documentation

7.36.5.1 struct [Config::behaviors_config](#) [Config::behaviors](#)

placeholder

7.36.5.2 struct [Config::controller_config](#) [Config::controller](#)

controller information

7.36.5.3 struct [Config::main_config](#) [Config::main](#)

core functionality information

7.36.5.4 struct [Config::motion_config](#) [Config::motion](#)

motion information

7.36.5.5 struct [Config::sound_config](#) [Config::sound](#)

sound information

7.36.5.6 struct [Config::vision_config](#) [Config::vision](#)

vision information

7.36.5.7 struct [Config::wireless_config](#) [Config::wireless](#)

wirless information

7.36.5.8 struct [Config::worldmodel2_config](#) [Config::worldmodel2](#)

world model information

The documentation for this class was generated from the following files:

- [Config.h](#)
- [Config.cc](#)

7.37 Config::behaviors_config Struct Reference

```
#include <Config.h>
```

7.37.1 Detailed Description

placeholder

Definition at line 79 of file Config.h.

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

- [Config.h](#)

7.38 Config::controller_config Struct Reference

```
#include <Config.h>
```

7.38.1 Detailed Description

controller information

Definition at line 83 of file Config.h.

Public Member Functions

- [controller_config \(\)](#)
constructor

Public Attributes

- int [gui_port](#)
port to listen for the GUI to connect to aibo on
- char [select_snd](#) [50]
sound file to use for "select" action
- char [next_snd](#) [50]
sound file to use for "next" action
- char [prev_snd](#) [50]
sound file to use for "prev" action
- char [read_snd](#) [50]
sound file to use for "read from std-in" action
- char [cancel_snd](#) [50]
sound file to use for "cancel" action

7.38.2 Constructor & Destructor Documentation

7.38.2.1 Config::controller_config::controller_config () [inline]

constructor

Definition at line 92 of file Config.h.

References `cancel_snd`, `gui_port`, `next_snd`, `prev_snd`, `read_snd`, and `select_snd`.

7.38.3 Member Data Documentation

7.38.3.1 char [Config::controller_config::cancel_snd\[50\]](#)

sound file to use for "cancel" action

Definition at line 89 of file Config.h.

7.38.3.2 int [Config::controller_config::gui_port](#)

port to listen for the GUI to connect to aibo on

Definition at line 84 of file Config.h.

7.38.3.3 char [Config::controller_config::next_snd\[50\]](#)

sound file to use for "next" action

Definition at line 86 of file Config.h.

7.38.3.4 char [Config::controller_config::prev_snd\[50\]](#)

sound file to use for "prev" action

Definition at line 87 of file Config.h.

7.38.3.5 char [Config::controller_config::read_snd\[50\]](#)

sound file to use for "read from std-in" action

Definition at line 88 of file Config.h.

7.38.3.6 char [Config::controller_config::select_snd\[50\]](#)

sound file to use for "select" action

Definition at line 85 of file Config.h.

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

- [Config.h](#)

7.39 Config::main_config Struct Reference

```
#include <Config.h>
```

7.39.1 Detailed Description

core functionality information

Definition at line 56 of file Config.h.

Public Member Functions

- [main_config \(\)](#)
constructor

Public Attributes

- int [console_port](#)
port to send/receive "console" information on (separate from system console)
- int [stderr_port](#)
port to send error information to
- int [error_level](#)
controls amount of info to error port
- int [debug_level](#)
controls amount of debug info
- int [verbose_level](#)
controls verbosity of info
- int [wsjoints_port](#)
port to send joint positions on
- int [wspids_port](#)
port to send pid info on
- int [headControl_port](#)
port for receiving head commands

- int [walkControl_port](#)
port for receiving walk commands
- int [estopControl_port](#)
port for receiving walk commands
- int [aibo3d_port](#)
port for send/receive of joint positions from Aibo 3D GUI
- bool [use_VT100](#)
if true, enables VT100 console codes (currently only in [Controller](#) menus - 1.3)

7.39.2 Constructor & Destructor Documentation

7.39.2.1 Config::main_config::main_config () [inline]

constructor

Definition at line 71 of file Config.h.

References [aibo3d_port](#), [console_port](#), [debug_level](#), [error_level](#), [estopControl_port](#), [headControl_port](#), [stderr_port](#), [use_VT100](#), [verbose_level](#), [walkControl_port](#), [wsjoints_port](#), and [wspids_port](#).

7.39.3 Member Data Documentation

7.39.3.1 int [Config::main_config::aibo3d_port](#)

port for send/receive of joint positions from Aibo 3D GUI

Definition at line 67 of file Config.h.

7.39.3.2 int [Config::main_config::console_port](#)

port to send/receive "console" information on (separate from system console)

Definition at line 57 of file Config.h.

7.39.3.3 int [Config::main_config::debug_level](#)

controls amount of debug info

Definition at line 60 of file Config.h.

7.39.3.4 int [Config::main_config::error_level](#)

controls amount of info to error port

Definition at line 59 of file Config.h.

7.39.3.5 int [Config::main_config::estopControl_port](#)

port for receiving walk commands

Definition at line 66 of file Config.h.

7.39.3.6 int [Config::main_config::headControl_port](#)

port for receiving head commands

Definition at line 64 of file Config.h.

7.39.3.7 int [Config::main_config::stderr_port](#)

port to send error information to

Definition at line 58 of file Config.h.

7.39.3.8 bool [Config::main_config::use_VT100](#)

if true, enables VT100 console codes (currently only in [Controller](#) menus - 1.3)

Definition at line 68 of file Config.h.

7.39.3.9 int [Config::main_config::verbose_level](#)

controls verbosity of info

Definition at line 61 of file Config.h.

7.39.3.10 int [Config::main_config::walkControl_port](#)

port for receiving walk commands

Definition at line 65 of file Config.h.

7.39.3.11 int [Config::main_config::wsjoints_port](#)

port to send joint positions on

Definition at line 62 of file Config.h.

7.39.3.12 int [Config::main_config::wspids_port](#)

port to send pid info on

Definition at line 63 of file Config.h.

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

- [Config.h](#)

7.40 Config::motion_config Struct Reference

```
#include <Config.h>
```

7.40.1 Detailed Description

motion information

Definition at line 98 of file Config.h.

Public Member Functions

- std::string [makePath](#) (std::string name)
returns an absolute path if is relative (to root), otherwise just name
- [motion_config](#) ()
constructor

Public Attributes

- std::string [root](#)
path on memory stick to "motion" files - for instance, position (.pos) and motion sequence (.mot)
- char [estop_on_snd](#) [50]
sound file to use when e-stop turned on
- char [estop_off_snd](#) [50]
sound file to use when e-stop turned off

7.40.2 Constructor & Destructor Documentation

7.40.2.1 Config::motion_config::motion_config () [inline]

constructor

Definition at line 109 of file Config.h.

References [estop_off_snd](#), [estop_on_snd](#), and [root](#).

7.40.3 Member Function Documentation

7.40.3.1 `std::string Config::motion_config::makePath (std::string name)` [inline]

returns an absolute path if *is* relative (to root), otherwise just *name*

Definition at line 104 of file Config.h.

References root.

7.40.4 Member Data Documentation

7.40.4.1 `char Config::motion_config::estop_off_snd[50]`

sound file to use when e-stop turned off

Definition at line 101 of file Config.h.

7.40.4.2 `char Config::motion_config::estop_on_snd[50]`

sound file to use when e-stop turned on

Definition at line 100 of file Config.h.

7.40.4.3 `std::string Config::motion_config::root`

path on memory stick to "motion" files - for instance, position (.pos) and motion sequence (.mot)

Definition at line 99 of file Config.h.

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

- [Config.h](#)

7.41 Config::sound_config Struct Reference

```
#include <Config.h>
```

7.41.1 Detailed Description

sound information

Definition at line 125 of file Config.h.

Public Member Functions

- std::string [makePath](#) (std::string name)
returns an absolute path if is relative (to root), otherwise just name
- [sound_config](#) ()
constructor

Public Attributes

- std::string [root](#)
path to sound clips
- unsigned int [sample_rate](#)
sample rate to send to system, currently only 8000 or 16000 supported
- unsigned int [sample_bits](#)
sample bit depth, either 8 or 16
- std::vector< std::string > [preload](#)
list of sounds to preload at boot

7.41.2 Constructor & Destructor Documentation

7.41.2.1 Config::sound_config::sound_config () [inline]

constructor

Definition at line 137 of file Config.h.

References [preload](#), [root](#), [sample_bits](#), and [sample_rate](#).

7.41.3 Member Function Documentation

7.41.3.1 `std::string Config::sound_config::makePath (std::string name)` [inline]

returns an absolute path if *is* relative (to root), otherwise just *name*

Definition at line 132 of file Config.h.

References root.

7.41.4 Member Data Documentation

7.41.4.1 `std::vector<std::string> Config::sound_config::preload`

list of sounds to preload at boot

Definition at line 129 of file Config.h.

7.41.4.2 `std::string Config::sound_config::root`

path to sound clips

Definition at line 126 of file Config.h.

7.41.4.3 `unsigned int Config::sound_config::sample_bits`

sample bit depth, either 8 or 16

Definition at line 128 of file Config.h.

7.41.4.4 `unsigned int Config::sound_config::sample_rate`

sample rate to send to system, currently only 8000 or 16000 supported

Definition at line 127 of file Config.h.

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

- [Config.h](#)

7.42 Config::vision_config Struct Reference

```
#include <Config.h>
```

7.42.1 Detailed Description

vision information

Definition at line 40 of file Config.h.

Public Member Functions

- [vision_config](#) ()
constructor

Public Attributes

- int [white_balance](#)
white balance
- int [gain](#)
gain
- int [shutter_speed](#)
shutter speed
- int [resolution](#)
resolution
- char [thresh](#) [30]
thresholds
- char [colors](#) [30]
colors
- int [raw_port](#)
port to send raw frames on
- int [rle_port](#)
port to send RLE frames on

- int [obj_port](#)

port to send object info on

7.42.2 Constructor & Destructor Documentation

7.42.2.1 Config::vision_config::vision_config () `[inline]`

constructor

Definition at line 52 of file Config.h.

References [colors](#), [gain](#), [obj_port](#), [raw_port](#), [resolution](#), [rle_port](#), [shutter_speed](#), [thresh](#), and [white_balance](#).

7.42.3 Member Data Documentation

7.42.3.1 char [Config::vision_config::colors](#)[30]

[colors](#)

Definition at line 46 of file Config.h.

7.42.3.2 int [Config::vision_config::gain](#)

[gain](#)

Definition at line 42 of file Config.h.

7.42.3.3 int [Config::vision_config::obj_port](#)

port to send object info on

Definition at line 49 of file Config.h.

7.42.3.4 int [Config::vision_config::raw_port](#)

port to send raw frames on

Definition at line 47 of file Config.h.

7.42.3.5 int [Config::vision_config::resolution](#)

resolution

Definition at line 44 of file Config.h.

7.42.3.6 int [Config::vision_config::rle_port](#)

port to send RLE frames on

Definition at line 48 of file Config.h.

7.42.3.7 int [Config::vision_config::shutter_speed](#)

shutter speed

Definition at line 43 of file Config.h.

7.42.3.8 char [Config::vision_config::thresh\[30\]](#)

thresholds

Definition at line 45 of file Config.h.

7.42.3.9 int [Config::vision_config::white_balance](#)

white balance

Definition at line 41 of file Config.h.

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

- [Config.h](#)

7.43 Config::wireless_config Struct Reference

```
#include <Config.h>
```

7.43.1 Detailed Description

wirless information

Definition at line 33 of file Config.h.

Public Member Functions

- [wireless_config \(\)](#)
constructor

Public Attributes

- [int id](#)
id number (in case you have more than one AIBO)

7.43.2 Constructor & Destructor Documentation

7.43.2.1 Config::wireless_config::wireless_config () [inline]

constructor

Definition at line 36 of file Config.h.

References [id](#).

7.43.3 Member Data Documentation

7.43.3.1 int Config::wireless_config::id

id number (in case you have more than one AIBO)

Definition at line 34 of file Config.h.

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

- [Config.h](#)

7.44 Config::worldmodel2_config Struct Reference

```
#include <Config.h>
```

7.44.1 Detailed Description

world model information

Definition at line 115 of file Config.h.

Public Member Functions

- [worldmodel2_config\(\)](#)

constructor

Public Attributes

- int [dm_port](#)

ports to use for sending world model information

- int [hm_port](#)

ports to use for sending world model information

- int [gm_port](#)

ports to use for sending world model information

- int [fs_port](#)

ports to use for sending world model information

7.44.2 Constructor & Destructor Documentation

7.44.2.1 Config::worldmodel2_config::worldmodel2_config() [inline]

constructor

Definition at line 121 of file Config.h.

References [dm_port](#), [fs_port](#), [gm_port](#), and [hm_port](#).

7.44.3 Member Data Documentation

7.44.3.1 int [Config::worldmodel2_config::dm_port](#)

ports to use for sending world model information

Definition at line 118 of file Config.h.

7.44.3.2 int [Config::worldmodel2_config::fs_port](#)

ports to use for sending world model information

Definition at line 118 of file Config.h.

7.44.3.3 int [Config::worldmodel2_config::gm_port](#)

ports to use for sending world model information

Definition at line 118 of file Config.h.

7.44.3.4 int [Config::worldmodel2_config::hm_port](#)

ports to use for sending world model information

Definition at line 118 of file Config.h.

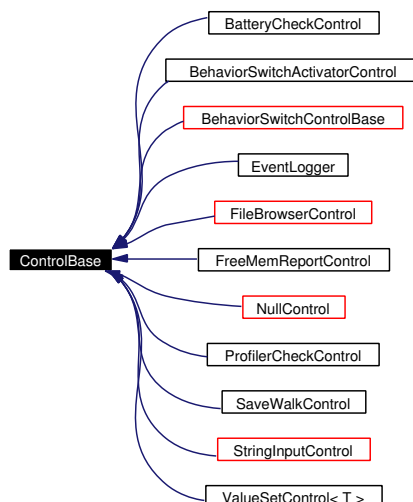
The documentation for this struct was generated from the following file:

- [Config.h](#)

7.45 ControlBase Class Reference

```
#include <ControlBase.h>
```

Inheritance diagram for ControlBase:



7.45.1 Detailed Description

Base class for all items in the [Controller](#) hierarchy.

These are similar to behaviors in that they can do processing and are told when to start and stop.

However, the important difference is that these have to follow a much tighter set of guidelines for a more refined purpose - user interface. Controls do not directly receive events - the [Controller](#) will process events for them and call the appropriate functions at the appropriate times. Controls are expected to fit into a heirarchical scheme, where each control (except the root) has a parent which created it, and may return its own children where appropriate.

This base class will do most of the work of maintain submenus for you, and will call appropriate virtual functions which you can override. Controls generally live in Behaviors/Controls/ as the [Controller](#) itself is a behavior, and these controls merely its tools.

The ControlBase pointers which are returned at various points are the responsibility of the creator. [Controller](#) will not delete them upon deactivation.

GUI Theory:

There are 3 levels to the user interface.

1. Robot/Local: Uses the LEDs and sounds for immediate feedback. No external resources needed
2. Text: Uses a console to display/request information.
3. GUI: Uses a graphical interface on an external screen

Obviously, higher levels require more technological resources, which also means there's more to go wrong and debug. However, another important distinction between the first level and the others is that the first level does not require the user to lose direct contact with the robot. Requiring the user to move back and forth from robot to computer can be much more frustrating than decoding LED signals or press head buttons. There are safety issues when triggering behaviors remotely if the robot is out of immediate reach. But of course, having a GUI and text output is extremely valuable in terms of ease of use and efficiency.

So, the lesson is to try to support all 3 levels so that your interfaces will be robust and efficient in a variety of environments. You'll thank yourself when you're demoing on the road and you can't get wavelan up, or the guest machine you're supposed to use doesn't have Java, or whatever.

Todo

ControlBase's should use [ReferenceCounter](#) so memory management is not an issue

See also:

[Controller](#), [NullControl](#)

Definition at line 57 of file ControlBase.h.

Public Member Functions

- [ControlBase](#) ()
Constructor.
- [ControlBase](#) (const std::string &n)
Constructor; initializes with a name.
- [ControlBase](#) (const std::string &n, const std::string &d)
Constructor; initializes with a name.
- virtual [~ControlBase](#) ()
Destructor.

- virtual [ControlBase](#) & [setName](#) (const std::string &n)
sets the name of the control
- virtual std::string [getName](#) () const
returns the name of the control
- virtual [ControlBase](#) & [setDescription](#) (const std::string d)
sets the description of the control
- virtual std::string [getDescription](#) () const
returns a short description of what the control does
- const std::vector< [ControlBase](#) * > & [getSlots](#) () const
returns the vector of controls
- std::string [getSlotName](#) (unsigned int i) const
returns the string that will appear in slot i
- unsigned int [slotsSize](#) () const
returns the number of options available
- void [setSlot](#) (unsigned int i, [ControlBase](#) *o)
sets i'th element of ::options to o
- void [pushSlot](#) ([ControlBase](#) *o)
sets next unused element of ::options to o
- void [clearSlots](#) ()
deletes each slot item and clears the slots
- virtual const std::vector< unsigned int > & [getHighlights](#) () const
returns a vector of the indicies of highlighted slots
- virtual void [setHighlights](#) (const std::vector< unsigned int > &hi)
sets the highlighted slots
- virtual void [highlightFirst](#) ()
sets the hilight to the first non-null slot
- virtual [MotionManager::MC_ID](#) [getDisplay](#) ()
returns display being used

- virtual [ControlBase](#) & [setDisplay](#) ([MotionManager::MC_ID](#) d)
sets display to use

Controller Functions

You probably want to override some of these, call the [ControlBase](#) functions from your code if you want default sound effects (or look in [Config::controller_config](#)).

- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#) disp_id, [Socket](#) *gui)
Called when the control is activated (or the control system is reactivating).
- virtual void [pause](#) ()
called when a control is being overridden by a child, or the control system is deactivating (e-stop being turned off)
- virtual void [refresh](#) ()
called when the child has died and this control should refresh its display
- virtual void [deactivate](#) ()
called when this control is being popped from the control stack
- virtual [ControlBase](#) * [doSelect](#) ()
when the user has trigger an "open selection" - default is to return the highlighted control
- virtual [ControlBase](#) * [doNextItem](#) ()
when the user wants to increment the control - default is to return the first non-null slot after the last highlight
- virtual [ControlBase](#) * [doPrevItem](#) ()
when the user wants to decrement the control - default is to return the last non-null slot before the first highlight
- virtual [ControlBase](#) * [doCancel](#) ()
when the user wants to cancel - you should almost always return NULL now unless you need to confirm something (e.g. "Save changes?")
- virtual [ControlBase](#) * [doReadStdIn](#) (const [std::string](#) &prompt=[std::string](#)())
prompt the user for text input on the current input device (cin, tekkotsu console (sout), or GUI)
- virtual [ControlBase](#) * [takeInput](#) (const [std::string](#) &msg)
called when the user has supplied a text string (may not have been prompted by [doReadStdIn](#)()!)

- virtual bool [validInput](#) (const std::string &msg)
may be called before takeInput to verify this Control can make sense of msg

Protected Member Functions

- virtual void [clearMenu](#) ()
clears the display (if use_VT100 is on)
- float [hilightsAvg](#) () const
returns the average of the hilighted indicies - used to know to play the "next" sound, or the "prev" sound when the hilight changes

Protected Attributes

- std::string [name](#)
the name of this control
- std::string [description](#)
the description of this control
- std::vector< unsigned int > [hilights](#)
keep sorted - index(es) of current selection - can have multiple if using GUI
- std::vector< [ControlBase](#) * > [options](#)
vector of controls to select from
- bool [doRewrite](#)
toggles using VT100 codes to reposition the cursor at the beginning of the menu
- [MotionManager::MC_ID](#) [display_id](#)
LedMC to use for displaying selection.
- [Socket](#) * [gui_comm](#)
socket to communicate with the GUI, if it is connected

Private Member Functions

- [ControlBase](#) (const [ControlBase](#) &)
you can override, but don't call this...
- [ControlBase](#) & [operator=](#) (const [ControlBase](#) &)
you can override, but don't call this...

7.45.2 Constructor & Destructor Documentation

7.45.2.1 [ControlBase::ControlBase \(\)](#) [inline]

Constructor.

Definition at line 60 of file [ControlBase.h](#).

References [description](#), [display_id](#), [doRewrite](#), [gui_comm](#), [hilights](#), [name](#), and [options](#).

7.45.2.2 [ControlBase::ControlBase \(const std::string & n\)](#) [inline]

Constructor, initializes with a name.

Definition at line 61 of file [ControlBase.h](#).

References [description](#), [display_id](#), [doRewrite](#), [gui_comm](#), [hilights](#), [name](#), and [options](#).

7.45.2.3 [ControlBase::ControlBase \(const std::string & n, const std::string & d\)](#) [inline]

Constructor, initializes with a name.

Definition at line 62 of file [ControlBase.h](#).

References [description](#), [display_id](#), [doRewrite](#), [gui_comm](#), [hilights](#), [name](#), and [options](#).

7.45.2.4 [virtual ControlBase::~~ControlBase \(\)](#) [inline, virtual]

Destructor.

Definition at line 65 of file [ControlBase.h](#).

References [clearSlots\(\)](#).

7.45.2.5 **ControlBase::ControlBase (const [ControlBase](#) &) [private]**

you can override, but don't call this...

7.45.3 Member Function Documentation

7.45.3.1 **[ControlBase](#) * [ControlBase](#)::activate ([MotionManager::MC_ID](#) *disp_id*, [Socket](#) * *gui*) [virtual]**

Called when the control is activated (or the control system is reactivating).

Takes the id number of a [LedMC](#) which the control should use, maintained by [Controller](#). Controls share the display which is passed, and may use the socket *gui* to communicate with the GUI controller, if it is connected.

Returns:

a [ControlBase](#) pointer. Return:

- *this* if the control should stay active (if it's not a one-shot command)
- `NULL` to return to parent
- other address to spawn a child control

Reimplemented in [BatteryCheckControl](#), [BehaviorActivatorControl](#), [BehaviorSwitchActivatorControl](#), [BehaviorSwitchControlBase](#), [FileBrowserControl](#), [HelpControl](#), [NullControl](#), [ProfilerCheckControl](#), [RebootControl](#), [SaveWalkControl](#), [ShutdownControl](#), [StringInputControl](#), and [ValueEditControl< T >](#).

Definition at line 8 of file [ControlBase.cc](#).

References [display_id](#), [gui_comm](#), [highlightFirst\(\)](#), and [refresh\(\)](#).

7.45.3.2 **void [ControlBase](#)::clearMenu () [protected, virtual]**

clears the display (if `use_VT100` is on)

Definition at line 310 of file [ControlBase.cc](#).

References [config](#), [Config::main](#), [options](#), [Socket::printf\(\)](#), [sout](#), and [Config::main_config::use_VT100](#).

7.45.3.3 **void [ControlBase](#)::clearSlots ()**

deletes each slot item and clears the slots

Definition at line 277 of file [ControlBase.cc](#).

References [hilights](#), and [options](#).

7.45.3.4 void ControlBase::deactivate () [virtual]

called when this control is being popped from the control stack

Reimplemented in [BatteryCheckControl](#), [LoadPostureControl](#), and [SaveWalkControl](#).

Definition at line 87 of file ControlBase.cc.

References [clearMenu\(\)](#), [config](#), [display_id](#), [doRewrite](#), [hilights](#), [MotionManager::invalid_MC_ID](#), [Config::main](#), [MMAccessor< MC_t >::mc\(\)](#), and [Config::main_config::use_VT100](#).

7.45.3.5 ControlBase * ControlBase::doCancel () [virtual]

when the user wants to cancel - you should almost always return NULL now unless you need to confirm something (e.g. "Save changes?")

Definition at line 178 of file ControlBase.cc.

References [Config::controller_config::cancel_snd](#), [config](#), [Config::controller](#), [SoundManager::PlayFile\(\)](#), and [sndman](#).

7.45.3.6 ControlBase * ControlBase::doNextItem () [virtual]

when the user wants to increment the control - default is to return the first non-null slot after the last hilight

Reimplemented in [NullControl](#), and [ValueEditControl< T >](#).

Definition at line 142 of file ControlBase.cc.

References [config](#), [Config::controller](#), [hilights](#), [Config::controller_config::next_snd](#), [options](#), [SoundManager::PlayFile\(\)](#), [refresh\(\)](#), and [sndman](#).

7.45.3.7 ControlBase * ControlBase::doPrevItem () [virtual]

when the user wants to decrement the control - default is to return the last non-null slot before the first hilight

Reimplemented in [NullControl](#), and [ValueEditControl< T >](#).

Definition at line 160 of file ControlBase.cc.

References [config](#), [Config::controller](#), [hilights](#), [options](#), [SoundManager::PlayFile\(\)](#), [Config::controller_config::prev_snd](#), [refresh\(\)](#), and [sndman](#).

7.45.3.8 **ControlBase** * **ControlBase::doReadStdIn** (const std::string & *prompt* = std::string()) [virtual]

prompt the user for text input on the current input device (cin, tekkotsu console (sout), or GUI)

Reimplemented in [NullControl](#), and [StringInputControl](#).

Definition at line 184 of file ControlBase.cc.

References [config](#), [Config::controller](#), [display_id](#), [ERS210Info::FaceLEDMask](#), [gui_comm](#), [MotionManager::invalid_MC_ID](#), [Wireless::isConnected\(\)](#), [MMAccessor<MC_t >::mc\(\)](#), [SoundManager::PlayFile\(\)](#), [Socket::printf\(\)](#), [Config::controller_config::read_snd](#), [sndman](#), [Socket::sock](#), [sout](#), [takeInput\(\)](#), and [wireless](#).

7.45.3.9 **ControlBase** * **ControlBase::doSelect** () [virtual]

when the user has trigger an "open selection" - default is to return the hilighted control

Reimplemented in [BatteryCheckControl](#), [EventLogger](#), [FileBrowserControl](#), [MCValueEditControl< T >](#), [NullControl](#), [RebootControl](#), [ShutdownControl](#), and [ValueEditControl< T >](#).

Definition at line 101 of file ControlBase.cc.

References [clearMenu\(\)](#), [config](#), [Config::controller](#), [display_id](#), [doRewrite](#), [ERS210Info::FaceLEDMask](#), [gui_comm](#), [hilights](#), [MotionManager::invalid_MC_ID](#), [Config::main](#), [MMAccessor<MC_t >::mc\(\)](#), [options](#), [SoundManager::PlayFile\(\)](#), [Socket::printf\(\)](#), [Config::controller_config::select_snd](#), [sndman](#), [sout](#), and [Config::main_config::use_VT100](#).

7.45.3.10 **virtual** std::string **ControlBase::getDescription** () const [inline, virtual]

returns a short description of what the control does

Reimplemented in [BehaviorSwitchActivatorControl](#), and [BehaviorSwitchControl< B, AI >](#).

Definition at line 95 of file ControlBase.h.

References [description](#).

7.45.3.11 **virtual** **MotionManager::MC_ID** **ControlBase::getDisplay** () [inline, virtual]

returns display being used

Definition at line 108 of file ControlBase.h.

References `display_id`, and `MotionManager::MC_ID`.

7.45.3.12 `virtual const std::vector<unsigned int>& ControlBase::getHighlights () const [inline, virtual]`

returns a vector of the indices of highlighted slots

Definition at line 104 of file `ControlBase.h`.

References `highlights`.

7.45.3.13 `virtual std::string ControlBase::getName () const [inline, virtual]`

returns the name of the control

Reimplemented in [BehaviorSwitchActivatorControl](#), [BehaviorSwitchControl< B, AI >](#), [FreeMemReportControl](#), and [ValueEditControl< T >](#).

Definition at line 92 of file `ControlBase.h`.

References `name`.

7.45.3.14 `std::string ControlBase::getSlotName (unsigned int i) const`

returns the string that will appear in slot *i*

Definition at line 259 of file `ControlBase.cc`.

References `options`.

7.45.3.15 `const std::vector<ControlBase*>& ControlBase::getSlots () const [inline]`

returns the vector of controls

Definition at line 97 of file `ControlBase.h`.

References `options`.

7.45.3.16 `void ControlBase::highlightFirst () [virtual]`

sets the highlight to the first non-null slot

Definition at line 300 of file `ControlBase.cc`.

References `highlights`, and `options`.

7.45.3.17 float ControlBase::hilightsAvg () const [protected]

returns the average of the hilighted indicies - used to know to play the "next" sound, or the "prev" sound when the hilight changes

Definition at line 317 of file ControlBase.cc.

References hilights.

7.45.3.18 ControlBase& ControlBase::operator= (const ControlBase &)
[private]

you can override, but don't call this...

7.45.3.19 void ControlBase::pause () [virtual]

called when a control is being overridden by a child, or the control system is deactivating (e-stop being turned off)

Reimplemented in [BatteryCheckControl](#), and [ValueEditControl< T >](#).

Definition at line 16 of file ControlBase.cc.

References doRewrite.

7.45.3.20 void ControlBase::pushSlot (ControlBase * o)

sets next unused element of ::options to o

Definition at line 273 of file ControlBase.cc.

References options.

7.45.3.21 void ControlBase::refresh () [virtual]

called when the child has died and this control should refresh its display

Reimplemented in [BatteryCheckControl](#), [EventLogger](#), [FreeMemReportControl](#), and [StringInputControl](#).

Definition at line 22 of file ControlBase.cc.

References clearMenu(), config, display_id, doRewrite, getName(), getSlotName(), gui_comm, hilights, MotionManager::invalid_MC_ID, Wireless::isConnected(), Config::main, MMAccessor< MC_t >::mc(), LedEngine::onedigit, options, Socket::printf(), Socket::sock, sout, LedEngine::twodigit, Config::main_config::use_VT100, wireless, and Socket::write().

7.45.3.22 virtual [ControlBase](#)& [ControlBase::setDescription](#) (const std::string *d*) [inline, virtual]

sets the description of the control

Definition at line 94 of file ControlBase.h.

References description.

7.45.3.23 virtual [ControlBase](#)& [ControlBase::setDisplay](#) ([MotionManager::MC.ID](#) *d*) [inline, virtual]

sets display to use

Definition at line 109 of file ControlBase.h.

References display_id.

7.45.3.24 void [ControlBase::setHighlights](#) (const std::vector< unsigned int > & *hi*) [virtual]

sets the highlighted slots

Definition at line 284 of file ControlBase.cc.

References config, Config::controller, hilights, hilightsAvg(), Config::controller.-config::next_snd, options, SoundManager::PlayFile(), Config::controller.-config::prev_snd, refresh(), and sndman.

7.45.3.25 virtual [ControlBase](#)& [ControlBase::setName](#) (const std::string & *n*) [inline, virtual]

sets the name of the control

Definition at line 91 of file ControlBase.h.

References name.

7.45.3.26 void [ControlBase::setSlot](#) (unsigned int *i*, [ControlBase](#) * *o*)

sets *i*'th element of ::options to *o*

Definition at line 267 of file ControlBase.cc.

References options.

7.45.3.27 unsigned int ControlBase::slotsSize () const [inline]

returns the number of options available

Definition at line 99 of file ControlBase.h.

References options.

7.45.3.28 ControlBase * ControlBase::takeInput (const std::string & msg)
[virtual]

called when the user has supplied a text string (may not have been prompted by [doReadStdIn\(\)](#)!)

Reimplemented in [NullControl](#), [SavePostureControl](#), [StringInputControl](#), and [ValueEditControl< T >](#).

Definition at line 213 of file ControlBase.cc.

References clearMenu(), config, doRewrite, doSelect(), hilights, Config::main, options, Socket::printf(), refresh(), serr, sout, and Config::main_config::use_VT100.

7.45.3.29 bool ControlBase::validInput (const std::string & msg) [virtual]

may be called before takeInput to verify this Control can make sense of msg

Definition at line 254 of file ControlBase.cc.

References options.

7.45.4 Member Data Documentation**7.45.4.1 std::string ControlBase::description** [protected]

the description of this control

Definition at line 119 of file ControlBase.h.

7.45.4.2 MotionManager::MC_ID ControlBase::display_id [protected]

[LedMC](#) to use for displaying selection.

Definition at line 126 of file ControlBase.h.

7.45.4.3 bool ControlBase::doRewrite [protected]

toggles using VT100 codes to reposition the cursor at the beginning of the menu

we don't always want to do this, any time someone else might have written to the display we set this to false so we don't overwrite it.

Definition at line 122 of file ControlBase.h.

7.45.4.4 [Socket*](#) [ControlBase::gui_comm](#) [protected]

socket to communicate with the GUI, if it is connected

Definition at line 127 of file ControlBase.h.

7.45.4.5 [std::vector<unsigned int>](#) [ControlBase::hilights](#) [protected]

keep sorted - index(es) of current selection - can have multiple if using GUI

Definition at line 120 of file ControlBase.h.

7.45.4.6 [std::string](#) [ControlBase::name](#) [protected]

the name of this control

Definition at line 118 of file ControlBase.h.

7.45.4.7 [std::vector<ControlBase*>](#) [ControlBase::options](#) [protected]

vector of controls to select from

Definition at line 121 of file ControlBase.h.

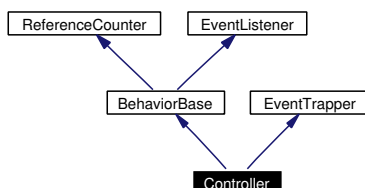
The documentation for this class was generated from the following files:

- [ControlBase.h](#)
- [ControlBase.cc](#)

7.46 Controller Class Reference

```
#include <Controller.h>
```

Inheritance diagram for Controller:



7.46.1 Detailed Description

Handles the menu/command system... when it detects the [EmergencyStopMC](#) is activated, it'll kick into high priority.

Keeps track of a command stack. A Control can designate another sub-control, which will receive events until it finishes

Events will then be sent to the parent again.

The GUI uses the same commands as the user (makes it much easier to have only one parser). The commands are:

- `'!refresh'` - redisplay the current control (handy on first connecting, or when other output has scrolled it off the screen)
- `'!reset'` - return to the root control
- `'!next'` - calls [ControlBase::doNextItem\(\)](#) of the current control
- `'!prev'` - calls [ControlBase::doPrevItem\(\)](#) of the current control
- `'!select'` - calls [ControlBase::doSelect\(\)](#) of the current control
- `'!cancel'` - calls [ControlBase::doCancel\(\)](#) of the current control
- `'!msg text'` - broadcasts *text* as a [TextMsgEvent](#)
- `'!highlight [n1 [n2 [...]]]` - highlights zero, one, or more items in the menu
- `'!input text'` - calls [ControlBase::takeInput\(text\)](#) on the currently highlighted control(s)
- any text not beginning with `'!'` - sent to [ControlBase::takeInput\(\)](#) of the current control

In return, to send the menus to the GUI, the following messages are sent: (newlines are required where shown)

- `'refresh`
text:title
int:numitems
*bool:hasSubmenus*₁
*bool:highlighted*₁
*text:item-title*₁
*text:item-description*₁
...
*bool:hasSubmenus*_{numitems}
*bool:highlighted*_{numitems}
*text:item-title*_{numitems}
*text:item-description*_{numitems} ' - refreshes the current menu
- `'status text'` - sets the status bar to *text* (until the next refresh)
- `'load`
text:classname
text:instancename int:port
[*arg1* [*arg2* [...]]]' - tells the GUI to load the java class named *classname*, and have it connect to *port*, passing it the argument list. *classname* should contain a constructor of the form `Classname(String host, int port, String args[])` the argument list is parsed as if it were on the console - unescaped or unquoted spaces will separate args into elements in the array
- `'close`
text:instancename' - calls `close()` on an object previously created by a `load` message. The Java object is expected to contain a function `void close()`.

`bool` types are expected to be numerical values, 0 for false, non-zero for true.

`load` and `close` are intended to allow pop-up windows for custom displays.

The upstream is the responsibility of the individual Controls, but the protocol is listed here to keep it together. When a control's state changes, it's that control's responsibility to refresh the UI (LEDs, console, and GUI as appropriate). Thus, future extensions to the upstream protocol are between the control which will use it and the GUI. Future extensions to the downstream protocol would involve changing Controller and the GUI.

The Controller may connect to `serr` in the future to pop-up an alert anytime output to `serr` occurs.

Note that all state is maintained on the robot - even if the GUI is connected, you can still use the buttons to interact with the controller, and the GUI will update to reflect the changes. In HCI (Human Computer Interaction) parlance, this is the MVC, Model-View-Controller architecture, almost by necessity. (HCI happens to be my double major when I was an undergrad ;)

Definition at line 81 of file Controller.h.

Public Member Functions

- [Controller](#) ()
Constructor.
- [Controller](#) ([ControlBase](#) *r)
Constructor, sets a default root control.
- virtual [~Controller](#) ()
Destructor.
- virtual void [DoStart](#) ()
register for events and resets the cmdstack
- virtual void [DoStop](#) ()
stop listening for events and resets the cmdstack
- virtual bool [trapEvent](#) (const [EventBase](#) &e)
passes an event to the top control
- virtual void [processEvent](#) (const [EventBase](#) &e)
just for e-stop activation/deactivation
- void [reset](#) ()
will take the command stack back down to the root
- void [refresh](#) ()
refreshes the display, for times like sub-control dying, the previous control needs to reset it's display
- void [push](#) ([ControlBase](#) *c)
puts a new control on top
- void [pop](#) ()
kills the top control, goes to previous

- `ControlBase * top ()`
returns the current control
- `Controller & setRoot (ControlBase *r)`
sets the root level control
- `Controller & setEStopID (MotionManager::MC_ID estopid)`
Sets the emergency stop MC to monitor for pausing.
- `virtual std::string getName () const`
Identifies the behavior in menus and such.

Static Public Member Functions

- `std::string getClassDescription ()`
Gives a short description of what this class of behaviors does... you should override this (but don't have to).
- `void loadGUI (const std::string &type, const std::string &name, unsigned int port)`
attempts to open a Java object on the desktop
- `void loadGUI (const std::string &type, const std::string &name, unsigned int port, const std::vector< std::string > &args)`
attempts to open a Java object on the desktop
- `void closeGUI (const std::string &name)`
calls close() on a Java object loaded with loadGUI() (on the desktop)
- `int gui_comm_callback (char *buf, int bytes)`
called by wireless when there's new data from the GUI
- `int console_callback (char *buf, int bytes)`
called by wireless when someone has entered new data on the tekkotsu console (NOT cin)

Static Public Attributes

- [EventBase nextItem](#)
event masks used by [processEvent\(\)](#)
- [EventBase prevItem](#)
event masks used by [processEvent\(\)](#)
- [EventBase nextItemFast](#)
event masks used by [processEvent\(\)](#)
- [EventBase prevItemFast](#)
event masks used by [processEvent\(\)](#)
- [EventBase selectItem](#)
event masks used by [processEvent\(\)](#)
- [EventBase cancel](#)
event masks used by [processEvent\(\)](#)

Protected Member Functions

- void [init](#) ()
assigns appropriate values to the static event bases
- void [takeLine](#) (const std::string &s)
called with each line that's entered on the tekkotsu console or from the GUI
- bool [setNext](#) ([ControlBase](#) *next)
maintains top Control
- void [activate](#) ()
called when the estop switches on
- void [deactivate](#) ()
called when the estop switches off
- bool [chkCmdStack](#) ()
returns true if a valid control is available on the stack

Static Protected Member Functions

- bool `calcPulse` (unsigned int t, unsigned int last, unsigned int period)
returns true when the current time and last time are in different periods
- std::string `makeLower` (const std::string &s)
returns lower case version of s
- std::string `removePrefix` (const std::string &str, const std::string &pre)
returns str with pre removed - if pre is not fully matched, str is returned unchanged

Protected Attributes

- `MotionManager::MC_ID` display
invalid_MC_ID if not active, otherwise id of high priority LEDs
- `MotionManager::MC_ID` estop_id
the `EmergencyStopMC` MC_ID that this Controller is monitoring
- `ControlBase * root`
the base control, if cmdstack underflows, it will be reset to this
- std::stack< `ControlBase *` > `cmdstack`
*the stack of the current control hierarchy
should never contain NULL entries*
- unsigned int `last_time`
the time of the last event
- unsigned int `cur_time`
the time of the current event (do() can check this instead of calling `get_time()`)*
- float `nextEv_val`
the magnitude of the last next event (::nextItem)
- unsigned int `nextEv_dur`
the duration of the last next event (::nextItem)
- float `prevEv_val`
the magnitude of the last prev event (::prevItem)

- unsigned int `prevEv_dur`
the duration of the last prev event (::prevItem)
- bool `alreadyGotBoth`
if `doReadStdIn()` was already called, but the buttons are both still down
- `Socket * gui_comm`
the socket to listen on for the gui

Static Protected Attributes

- `Controller * theOneController` = NULL
currently can't pull connection socket off of server socket, so only one Controller

Private Member Functions

- `Controller` (const `Controller` &)
shouldn't be called...
- `Controller` & `operator=` (const `Controller` &)
shouldn't be called...

7.46.2 Constructor & Destructor Documentation

7.46.2.1 `Controller::Controller ()` [inline]

Constructor.

Definition at line 83 of file `Controller.h`.

References `alreadyGotBoth`, `cmdstack`, `cur_time`, `display`, `estop_id`, `gui_comm`, `init()`, `last_time`, `nextEv_dur`, `nextEv_val`, `prevEv_dur`, `prevEv_val`, `root`, `SocketNS::SOCK_STREAM`, and `wireless`.

7.46.2.2 `Controller::Controller (ControlBase * r)` [inline]

Constructor, sets a default root control.

Definition at line 84 of file `Controller.h`.

References alreadyGotBoth, cmdstack, cur_time, display, estop_id, gui_comm, init(), last_time, nextEv_dur, nextEv_val, prevEv_dur, prevEv_val, root, SocketNS::SOCK_STREAM, and wireless.

7.46.2.3 virtual Controller::~~Controller() [inline, virtual]

Destructor.

Definition at line 85 of file Controller.h.

References root.

7.46.2.4 Controller::Controller(const Controller &) [private]

shouldn't be called...

7.46.3 Member Function Documentation

7.46.3.1 void Controller::activate() [protected]

called when the estop switches on

causes the top control to activate, registers for button events

Definition at line 314 of file Controller.cc.

References MotionManager::addMotion(), EventRouter::addTrapper(), EventBase::buttonEGID, chkCmdStack(), cmdstack, display, erouter, ERS210Info::FaceLEDMask, gui_comm, MotionManager::kEmergencyPriority, and motman.

7.46.3.2 bool Controller::calcPulse(unsigned int *t*, unsigned int *last*, unsigned int *period*) [inline, static, protected]

returns true when the current time and last time are in different periods

Definition at line 166 of file Controller.h.

7.46.3.3 bool Controller::chkCmdStack() [protected]

returns true if a valid control is available on the stack

if the stack is empty, will push root if it's non-null

Definition at line 336 of file Controller.cc.

References cmdstack, display, gui_comm, push(), and root.

7.46.3.4 void Controller::closeGUI (const std::string & name) [static]

calls close() on a Java object loaded with [loadGUI\(\)](#) (on the desktop)

Definition at line 175 of file Controller.cc.

References [gui_comm](#), [Socket::printf\(\)](#), and [theOneController](#).

7.46.3.5 int Controller::console_callback (char * buf, int bytes) [static]

called by wireless when someone has entered new data on the tekkotsu console (NOT cin)

Definition at line 203 of file Controller.cc.

References [erouter](#), [gui_comm](#), [Wireless::isConnected\(\)](#), [EventRouter::postEvent\(\)](#), [Socket::sock](#), [takeLine\(\)](#), [theOneController](#), and [wireless](#).

7.46.3.6 void Controller::deactivate () [protected]

called when the estop switches off

causes the top control to deactivate, stops listening for buttons

Definition at line 326 of file Controller.cc.

References [cmdstack](#), [display](#), [erouter](#), [MotionManager::invalid_MC_ID](#), [ERS210Info::LEDOffset](#), [motman](#), [ERS210Info::NumLEDs](#), [MotionManager::removeMotion\(\)](#), [EventRouter::removeTrapper\(\)](#), and [MotionManager::setOutput\(\)](#).

7.46.3.7 void Controller::DoStart () [virtual]

register for events and resets the cmdstack

Reimplemented from [BehaviorBase](#).

Definition at line 25 of file Controller.cc.

References [EventRouter::addListener\(\)](#), [Config::controller_config::cancel_snd](#), [config](#), [Config::controller](#), [BehaviorBase::DoStart\(\)](#), [erouter](#), [EventBase::estopEGID](#), [gui_comm](#), [gui_comm_callback\(\)](#), [Config::controller_config::gui_port](#), [Wireless::listen\(\)](#), [SoundManager::LoadFile\(\)](#), [Config::controller_config::next_snd](#), [Config::controller_config::prev_snd](#), [Config::controller_config::read_snd](#), [reset\(\)](#), [Config::controller_config::select_snd](#), [Wireless::setReceiver\(\)](#), [sndman](#), [Socket::sock](#), [theOneController](#), and [wireless](#).

7.46.3.8 void Controller::DoStop () [virtual]

stop listening for events and resets the cmdstack

Reimplemented from [BehaviorBase](#).

Definition at line 40 of file Controller.cc.

References [Config::controller_config::cancel_snd](#), [Wireless::close\(\)](#), [config](#), [Config::controller](#), [BehaviorBase::DoStop\(\)](#), [erouter](#), [EventRouter::forgetListener\(\)](#), [gui_comm](#), [Config::controller_config::next_snd](#), [Config::controller_config::prev_snd](#), [Config::controller_config::read_snd](#), [SoundManager::ReleaseFile\(\)](#), [reset\(\)](#), [Config::controller_config::select_snd](#), [sndman](#), [theOneController](#), and [wireless](#).

7.46.3.9 std::string Controller::getClassDescription () [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 117 of file Controller.h.

7.46.3.10 virtual std::string Controller::getName () const [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 116 of file Controller.h.

7.46.3.11 int Controller::gui_comm_callback (char * buf, int bytes) [static]

called by wireless when there's new data from the GUI

Definition at line 179 of file Controller.cc.

References [takeLine\(\)](#), and [theOneController](#).

7.46.3.12 void Controller::init () [protected]

assigns appropriate values to the static event bases

Definition at line 231 of file Controller.cc.

References [EventBase::buttonEGID](#), [cancel](#), [EventBase::deactivateETID](#), [WorldState::ERS210Mask](#), [WorldState::ERS220Mask](#), [nextItem](#), [nextItemFast](#), [prevItem](#), [prevItemFast](#), [WorldState::robotDesign](#), [selectItem](#), [state](#), and [EventBase::statusETID](#).

7.46.3.13 `void Controller::loadGUI (const std::string & type, const std::string & name, unsigned int port, const std::vector< std::string > & args)`
[static]

attempts to open a Java object on the desktop

Definition at line 159 of file Controller.cc.

References gui_comm, theOneController, and Socket::write().

7.46.3.14 `void Controller::loadGUI (const std::string & type, const std::string & name, unsigned int port)` [inline, static]

attempts to open a Java object on the desktop

Definition at line 121 of file Controller.h.

7.46.3.15 `std::string Controller::makeLower (const std::string & s)`
[static, protected]

returns lower case version of *s*

Definition at line 350 of file Controller.cc.

7.46.3.16 `Controller& Controller::operator= (const Controller &)`
[private]

shouldn't be called...

7.46.3.17 `void Controller::pop ()`

kills the top control, goes to previous

Definition at line 134 of file Controller.cc.

References cmdstack, and refresh().

7.46.3.18 `void Controller::processEvent (const EventBase & e)` [virtual]

just for e-stop activation/deactivation

Reimplemented from BehaviorBase.

Definition at line 101 of file Controller.cc.

References activate(), EventBase::activateETID, deactivate(), display, EventBase::getTypeID(), and MotionManager::invalid_MC_ID.

7.46.3.19 void Controller::push (ControlBase * *c*)

puts a new control on top

Definition at line 126 of file Controller.cc.

References chkCmdStack(), cmdstack, display, gui_comm, and setNext().

7.46.3.20 void Controller::refresh ()

refreshes the display, for times like sub-control dying, the previous control needs to reset it's display

Definition at line 120 of file Controller.cc.

References chkCmdStack(), and cmdstack.

7.46.3.21 std::string Controller::removePrefix (const std::string & *str*, const std::string & *pre*) [static, protected]

returns *str* with *pre* removed - if *pre* is not fully matched, *str* is returned unchanged

Definition at line 359 of file Controller.cc.

7.46.3.22 void Controller::reset ()

will take the command stack back down to the root

Definition at line 111 of file Controller.cc.

References cmdstack, and pop().

7.46.3.23 Controller & Controller::setEStopID (MotionManager::MC_ID *estopid*)

Sets the emergency stop MC to monitor for pausing.

Definition at line 147 of file Controller.cc.

References activate(), deactivate(), display, estop_id, MotionManager::invalid_MC_ID, motman, and MotionManager::peekMotion().

7.46.3.24 bool Controller::setNext (ControlBase * *next*) [protected]

maintains top Control

Parameters:

next one of:

- NULL: [pop\(\)](#) ::cmdstack
- ::cmdstack.top(): nothing
- other address: ::push(*next*)

Returns:

true, all the time, for convenience from [trapEvent\(\)](#)

Definition at line 306 of file Controller.cc.

References cmdstack, pop(), and push().

7.46.3.25 [Controller](#) & Controller::setRoot ([ControlBase](#) * *r*)

sets the root level control

Definition at line 140 of file Controller.cc.

References refresh(), reset(), and root.

7.46.3.26 void Controller::takeLine (const std::string & *s*) [protected]

called with each line that's entered on the tekkotsu console or from the GUI

Definition at line 254 of file Controller.cc.

References cmdstack, erouter, EventRouter::postEvent(), refresh(), reset(), and set-Next().

7.46.3.27 [ControlBase](#)* Controller::top () [inline]

returns the current control

Definition at line 110 of file Controller.h.

References cmdstack.

7.46.3.28 bool Controller::trapEvent (const [EventBase](#) & *e*) [virtual]

passes an event to the top control

Implements [EventTrapper](#).

Definition at line 53 of file Controller.cc.

References `alreadyGotBoth`, `calcPulse()`, `cancel`, `chkCmdStack()`, `cmdstack`, `cur_time`, `get_time()`, `EventBase::getDuration()`, `EventBase::getMagnitude()`, `EventBase::getTypeID()`, `last_time`, `nextEv_dur`, `nextEv_val`, `nextItem`, `nextItemFast`, `prevEv_dur`, `prevEv_val`, `prevItem`, `prevItemFast`, `EventBase::sameGenSource()`, `selectItem`, and `setNext()`.

7.46.4 Member Data Documentation

7.46.4.1 `bool Controller::alreadyGotBoth` [protected]

if `doReadStdIn()` was already called, but the buttons are both still down

Definition at line 185 of file `Controller.h`.

7.46.4.2 `EventBase Controller::cancel` [static]

event masks used by `processEvent()`

Definition at line 22 of file `Controller.cc`.

7.46.4.3 `std::stack< ControlBase* > Controller::cmdstack` [protected]

the stack of the current control hierarchy

should never contain NULL entries

Definition at line 163 of file `Controller.h`.

7.46.4.4 `unsigned int Controller::cur_time` [protected]

the time of the current event (`do*()` can check this instead of calling `get_time()`)

Definition at line 180 of file `Controller.h`.

7.46.4.5 `MotionManager::MC_ID Controller::display` [protected]

`invalid_MC_ID` if not active, otherwise id of high priority LEDs

Definition at line 153 of file `Controller.h`.

7.46.4.6 `MotionManager::MC_ID Controller::estop_id` [protected]

the `EmergencyStopMC` MC_ID that this Controller is monitoring

Definition at line 156 of file `Controller.h`.

7.46.4.7 `Socket* Controller::gui_comm` [protected]

the socket to listen on for the gui

Definition at line 187 of file Controller.h.

7.46.4.8 `unsigned int Controller::last_time` [protected]

the time of the last event

Definition at line 179 of file Controller.h.

7.46.4.9 `unsigned int Controller::nextEv_dur` [protected]

the duration of the last next event (::nextItem)

Definition at line 182 of file Controller.h.

7.46.4.10 `float Controller::nextEv_val` [protected]

the magnitude of the last next event (::nextItem)

Definition at line 181 of file Controller.h.

7.46.4.11 `EventBase Controller::nextItem` [static]

event masks used by `processEvent()`

Definition at line 17 of file Controller.cc.

7.46.4.12 `EventBase Controller::nextItemFast` [static]

event masks used by `processEvent()`

Definition at line 19 of file Controller.cc.

7.46.4.13 `unsigned int Controller::prevEv_dur` [protected]

the duration of the last prev event (::prevItem)

Definition at line 184 of file Controller.h.

7.46.4.14 float [Controller::prevEv_val](#) [protected]

the magnitude of the last prev event (::prevItem)

Definition at line 183 of file Controller.h.

7.46.4.15 [EventBase Controller::prevItem](#) [static]

event masks used by [processEvent\(\)](#)

Definition at line 18 of file Controller.cc.

7.46.4.16 [EventBase Controller::prevItemFast](#) [static]

event masks used by [processEvent\(\)](#)

Definition at line 20 of file Controller.cc.

7.46.4.17 [ControlBase*](#) [Controller::root](#) [protected]

the base control, if cmdstack underflows, it will be reset to this

Definition at line 159 of file Controller.h.

7.46.4.18 [EventBase Controller::selectItem](#) [static]

event masks used by [processEvent\(\)](#)

Definition at line 21 of file Controller.cc.

7.46.4.19 [Controller *](#) [Controller::theOneController](#) = NULL [static, protected]

currently can't pull connection socket off of server socket, so only one Controller

Definition at line 14 of file Controller.cc.

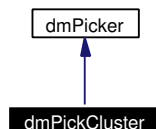
The documentation for this class was generated from the following files:

- [Controller.h](#)
- [Controller.cc](#)

7.47 dmPickCluster Struct Reference

```
#include <almStructures.h>
```

Inheritance diagram for dmPickCluster:



7.47.1 Detailed Description

Picks cluster membership out of the SDM. For completeness—not use!

Definition at line 109 of file almStructures.h.

Public Member Functions

- virtual float [operator\(\)](#) ([dm_cell](#) &c) const
functor - TSS_TODO

7.47.2 Member Function Documentation

7.47.2.1 virtual float dmPickCluster::operator() ([dm_cell](#) & c) const [inline, virtual]

functor - TSS_TODO

Implements [dmPicker](#).

Definition at line 111 of file almStructures.h.

References [_dm_cell::cluster](#).

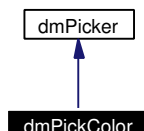
The documentation for this struct was generated from the following file:

- [almStructures.h](#)

7.48 dmPickColor Struct Reference

```
#include <almStructures.h>
```

Inheritance diagram for dmPickColor:



7.48.1 Detailed Description

Picks color measurements out of the SDM.

Definition at line 104 of file almStructures.h.

Public Member Functions

- virtual float [operator\(\)](#) ([dm_cell](#) &c) const
functor - TSS_TODO

7.48.2 Member Function Documentation

7.48.2.1 virtual float dmPickColor::operator() ([dm_cell](#) & c) const [inline, virtual]

functor - TSS_TODO

Implements [dmPicker](#).

Definition at line 106 of file almStructures.h.

References [_dm_cell::color](#).

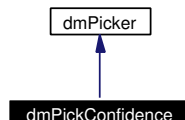
The documentation for this struct was generated from the following file:

- [almStructures.h](#)

7.49 dmPickConfidence Struct Reference

```
#include <almStructures.h>
```

Inheritance diagram for dmPickConfidence:



7.49.1 Detailed Description

Picks confidence values out of the SDM.

Definition at line 99 of file almStructures.h.

Public Member Functions

- virtual float [operator\(\)](#) ([dm_cell](#) &c) const
functor - TSS.TODO

7.49.2 Member Function Documentation

7.49.2.1 virtual float dmPickConfidence::operator() ([dm_cell](#) & c) const [inline, virtual]

functor - TSS.TODO

Implements [dmPicker](#).

Definition at line 101 of file almStructures.h.

References [_dm_cell::confidence](#).

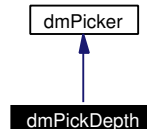
The documentation for this struct was generated from the following file:

- [almStructures.h](#)

7.50 dmPickDepth Struct Reference

```
#include <almStructures.h>
```

Inheritance diagram for dmPickDepth:



7.50.1 Detailed Description

Picks depth measurements out of the SDM.

Definition at line 94 of file almStructures.h.

Public Member Functions

- virtual float [operator\(\)](#) ([dm_cell](#) &c) const
functor - TSS_TODO

7.50.2 Member Function Documentation

7.50.2.1 virtual float dmPickDepth::operator() ([dm_cell](#) &c) const [inline, virtual]

functor - TSS_TODO

Implements [dmPicker](#).

Definition at line 96 of file almStructures.h.

References [_dm_cell::depth](#).

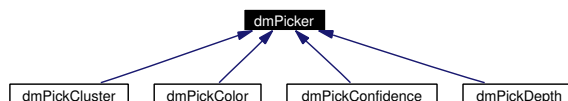
The documentation for this struct was generated from the following file:

- [almStructures.h](#)

7.51 dmPicker Struct Reference

```
#include <almStructures.h>
```

Inheritance diagram for dmPicker:



7.51.1 Detailed Description

Abstract base class for depth map data pickers. Implementations available.

Definition at line 62 of file almStructures.h.

Public Member Functions

- virtual float [operator\(\)](#) ([dm_cell](#) &c) const=0
functor - TSS.TODO

7.51.2 Member Function Documentation

7.51.2.1 virtual float [dmPicker::operator\(\)](#) ([dm_cell](#) &c) const [pure virtual]

functor - TSS.TODO

Implemented in [dmPickDepth](#), [dmPickConfidence](#), [dmPickColor](#), and [dmPickCluster](#).

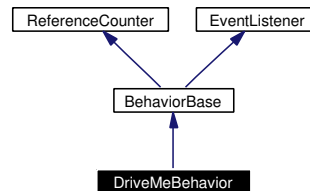
The documentation for this struct was generated from the following file:

- [almStructures.h](#)

7.52 DriveMeBehavior Class Reference

```
#include <DriveMeBehavior.h>
```

Inheritance diagram for DriveMeBehavior:



7.52.1 Detailed Description

A very simple behavior that asks the user for [WalkMC](#) walking parameters and a walk duration.

The AIBO walks accordingly and then stands up, then asks again. And so on and so on.

Input is from cin, not the tekkotsu console (sout)

Definition at line 13 of file DriveMeBehavior.h.

Public Member Functions

- [DriveMeBehavior](#) ()
constructor
- virtual [~DriveMeBehavior](#) ()
destructor
- virtual void [DoStart](#) ()
By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.
- virtual void [DoStop](#) ()
By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).
- virtual void [processEvent](#) (const [EventBase](#) &event)

Allows you to get away with not supplying a [processEvent\(\)](#) function for the [Event-Listener](#) interface. By default, does nothing.

- virtual std::string [getName](#) () const

Identifies the behavior in menus and such.

Static Public Member Functions

- std::string [getClassDescription](#) ()

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- [MotionManager::MC_ID](#) walker_id

walks

- [MotionManager::MC_ID](#) stand_id

stands up first

- [SharedObject](#)< [MotionSequenceMC](#)< [MotionSequence::SizeSmall](#) > >
[stand](#)

for standing

- double [last_dx](#)

the last dx received

- double [last_dy](#)

the last dy received

- double [last_da](#)

the last da received

- unsigned int [last_time](#)

timestamp of last parameter set

7.52.2 Constructor & Destructor Documentation

7.52.2.1 DriveMeBehavior::DriveMeBehavior ()

constructor

Definition at line 21 of file DriveMeBehavior.cc.

References [stand](#).

7.52.2.2 virtual DriveMeBehavior::~DriveMeBehavior () [inline, virtual]

destructor

Definition at line 16 of file DriveMeBehavior.h.

7.52.3 Member Function Documentation

7.52.3.1 void DriveMeBehavior::DoStart () [virtual]

By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.

Reimplemented from [BehaviorBase](#).

Definition at line 33 of file DriveMeBehavior.cc.

References [EventManager::addListener\(\)](#), [MotionManager::addMotion\(\)](#), [EventManager::addTimer\(\)](#), [BehaviorBase::DoStart\(\)](#), [erouter](#), [MotionManager::kStdPriority](#), [motman](#), [stand](#), [stand_id](#), [EventBase::timerEGID](#), and [walker_id](#).

7.52.3.2 void DriveMeBehavior::DoStop () [virtual]

By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).

Reimplemented from [BehaviorBase](#).

Definition at line 46 of file DriveMeBehavior.cc.

References [BehaviorBase::DoStop\(\)](#), [erouter](#), [EventManager::forgetListener\(\)](#), [motman](#), [MotionManager::removeMotion\(\)](#), [EventManager::removeTimer\(\)](#), [stand_id](#), and [walker_id](#).

7.52.3.3 `std::string DriveMeBehavior::getClassDescription ()` [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 24 of file DriveMeBehavior.h.

7.52.3.4 `virtual std::string DriveMeBehavior::getName () const` [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 23 of file DriveMeBehavior.h.

7.52.3.5 `void DriveMeBehavior::processEvent (const EventBase & event)` [virtual]

Allows you to get away with not supplying a `processEvent()` function for the [Event-Listener](#) interface. By default, does nothing.

Reimplemented from [BehaviorBase](#).

Definition at line 59 of file DriveMeBehavior.cc.

References `EventRouter::addTimer()`, `MotionManager::checkinMotion()`, `MotionManager::checkoutMotion()`, `erouter`, `EventBase::getGeneratorID()`, `last_da`, `last_dx`, `last_dy`, `last_time`, `motman`, `MotionSequence::play()`, `WalkMC::setTargetVelocity()`, `stand_id`, `EventBase::timerEGID`, and `walker_id`.

7.52.4 Member Data Documentation

7.52.4.1 `double DriveMeBehavior::last_da` [protected]

the last da received

Definition at line 33 of file DriveMeBehavior.h.

7.52.4.2 `double DriveMeBehavior::last_dx` [protected]

the last dx received

Definition at line 31 of file DriveMeBehavior.h.

7.52.4.3 double [DriveMeBehavior::last_dy](#) [protected]

the last dy received

Definition at line 32 of file DriveMeBehavior.h.

7.52.4.4 unsigned int [DriveMeBehavior::last_time](#) [protected]

timestamp of last parameter set

Definition at line 34 of file DriveMeBehavior.h.

7.52.4.5 [SharedObject< MotionSequenceMC<MotionSequence::SizeSmall> >](#)
[DriveMeBehavior::stand](#) [protected]

for standing

Definition at line 29 of file DriveMeBehavior.h.

7.52.4.6 [MotionManager::MC_ID DriveMeBehavior::stand_id](#)
[protected]

stands up first

Definition at line 28 of file DriveMeBehavior.h.

7.52.4.7 [MotionManager::MC_ID DriveMeBehavior::walker_id](#)
[protected]

walks

Definition at line 27 of file DriveMeBehavior.h.

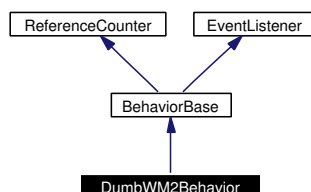
The documentation for this class was generated from the following files:

- [DriveMeBehavior.h](#)
- [DriveMeBehavior.cc](#)

7.53 DumbWM2Behavior Class Reference

```
#include <DumbWM2Behavior.h>
```

Inheritance diagram for DumbWM2Behavior:



7.53.1 Detailed Description

Simply turns on a WM2 object. Useful for running concurrently with other behaviors and seeing what shows up in the world model.

Definition at line 9 of file DumbWM2Behavior.h.

Public Member Functions

- [DumbWM2Behavior](#) ()
constructor
- virtual [~DumbWM2Behavior](#) ()
destructor
- virtual void [DoStart](#) ()
By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.
- virtual void [DoStop](#) ()
By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).
- virtual void [processEvent](#) (const [EventBase](#) &)
doesn't do anything
- virtual std::string [getName](#) () const
Identifies the behavior in menus and such.

Static Public Member Functions

- `std::string getClassDescription ()`

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- `WorldModel2 WM2`

the world model this is associated with

7.53.2 Constructor & Destructor Documentation

7.53.2.1 `DumbWM2Behavior::DumbWM2Behavior ()` [inline]

constructor

Definition at line 12 of file DumbWM2Behavior.h.

References WM2.

7.53.2.2 `virtual DumbWM2Behavior::~~DumbWM2Behavior ()` [inline, virtual]

destructor

Definition at line 15 of file DumbWM2Behavior.h.

7.53.3 Member Function Documentation

7.53.3.1 `virtual void DumbWM2Behavior::DoStart ()` [inline, virtual]

By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.

Reimplemented from [BehaviorBase](#).

Definition at line 18 of file DumbWM2Behavior.h.

References [BehaviorBase::DoStart\(\)](#), [WorldModel2::enableIR\(\)](#), and WM2.

7.53.3.2 virtual void DumbWM2Behavior::DoStop () [inline, virtual]

By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).

Reimplemented from [BehaviorBase](#).

Definition at line 23 of file DumbWM2Behavior.h.

References [WorldModel2::disableIR\(\)](#), [BehaviorBase::DoStop\(\)](#), and [WM2](#).

7.53.3.3 std::string DumbWM2Behavior::getClassDescription () [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 32 of file DumbWM2Behavior.h.

7.53.3.4 virtual std::string DumbWM2Behavior::getName () const [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 31 of file DumbWM2Behavior.h.

7.53.3.5 virtual void DumbWM2Behavior::processEvent (const [EventBase](#) &) [inline, virtual]

doesn't do anything

Reimplemented from [BehaviorBase](#).

Definition at line 29 of file DumbWM2Behavior.h.

7.53.4 Member Data Documentation**7.53.4.1 [WorldModel2](#) DumbWM2Behavior::WM2 [protected]**

the world model this is associated with

Definition at line 35 of file DumbWM2Behavior.h.

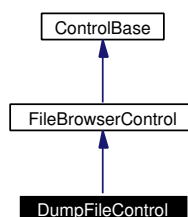
The documentation for this class was generated from the following file:

- [DumbWM2Behavior.h](#)

7.54 DumpFileControl Class Reference

```
#include <DumpFileControl.h>
```

Inheritance diagram for DumpFileControl:



7.54.1 Detailed Description

Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...).

Definition at line 9 of file DumpFileControl.h.

Public Member Functions

- [DumpFileControl](#) (const std::string &n, const std::string &r)
Constructor.
- virtual [~DumpFileControl](#) ()
Destructor.

Protected Member Functions

- virtual [ControlBase](#) * [selectedFile](#) (const std::string &f)
does the actual loading of the [MotionSequence](#)

7.54.2 Constructor & Destructor Documentation

7.54.2.1 DumpFileControl::DumpFileControl (const std::string & n, const std::string & r) [inline]

Constructor.

Definition at line 12 of file DumpFileControl.h.

7.54.2.2 `virtual DumpFileControl::~~DumpFileControl () [inline, virtual]`

Destructor.

Definition at line 16 of file DumpFileControl.h.

7.54.3 Member Function Documentation

7.54.3.1 `virtual ControlBase* DumpFileControl::selectedFile (const std::string &f) [inline, protected, virtual]`

does the actual loading of the [MotionSequence](#)

Reimplemented from [FileBrowserControl](#).

Definition at line 20 of file DumpFileControl.h.

References [ControlBase::doRewrite](#), [Socket::getWriteBuffer\(\)](#), [Socket::printf\(\)](#), [serr](#), [sout](#), and [Socket::write\(\)](#).

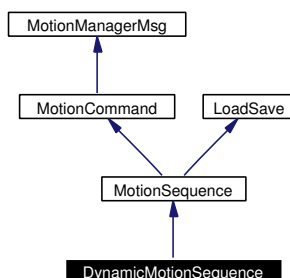
The documentation for this class was generated from the following file:

- [DumpFileControl.h](#)

7.55 DynamicMotionSequence Class Reference

```
#include <DynamicMotionSequence.h>
```

Inheritance diagram for DynamicMotionSequence:



7.55.1 Detailed Description

Uses STL's vector for dynamic memory allocation - don't use this as a motion command, pointers in shared memory regions can be invalid in other processes.

See [MotionSequenceMC](#) for documentation on its members

Definition at line 10 of file DynamicMotionSequence.h.

Public Member Functions

- [DynamicMotionSequence](#) ()
constructor
- [DynamicMotionSequence](#) (const char *filename)
constructor, loads from a file and then resets the playtime to beginning and begins to play
- virtual [~DynamicMotionSequence](#) ()
destructor
- virtual int [updateOutputs](#) ()
is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)
- virtual void [clear](#) ()
empties out the sequence (constant time operation - faster than a series of pops)

- virtual unsigned int [getMaxFrames](#) () const
returns the maximum number of key frames (Move's) which can be stored, determined by the instantiating MotionSequenceMC's template parameter
- virtual unsigned int [getUsedFrames](#) () const
returns the number of used key frames (Move's) which have been stored by the instantiation [MotionSequence](#) subclass

Protected Types

- typedef std::vector< Move > [list_t](#)
shorthand for the [ListMemBuf](#) that stores all of the movement frames

Protected Member Functions

- virtual Move & [getKeyFrame](#) (Move_idx_t x)
returns the Move struct corresponding to x in the subclass's actual data structure
- virtual const Move & [getKeyFrame](#) (Move_idx_t x) const
returns the Move struct corresponding to x in the subclass's actual data structure
- virtual [Move_idx_t](#) [newKeyFrame](#) ()
causes subclass to create a new Move structure, returns its index
- virtual void [eraseKeyFrame](#) (Move_idx_t x)
causes subclass to mark the corresponding Move structure as free
- void [setRange](#) (unsigned int t, [Move_idx_t](#) &prev, [Move_idx_t](#) &next) const
Sets prev and next to the appropriate values for the given time and output index.

Protected Attributes

- [list_t](#) [moves](#)
stores all of the movement keyframes
- std::vector< [Move_idx_t](#) > [erased](#)
recycles erased keyframes, can't just shift elements in [moves](#), it would throw of index numbers in Move structures

7.55.2 Member Typedef Documentation

7.55.2.1 `typedef std::vector<Move> DynamicMotionSequence::list_t [protected]`

shorthand for the [ListMemBuf](#) that stores all of the movement frames

Definition at line 76 of file `DynamicMotionSequence.h`.

7.55.3 Constructor & Destructor Documentation

7.55.3.1 `DynamicMotionSequence::DynamicMotionSequence () [inline]`

constructor

Definition at line 13 of file `DynamicMotionSequence.h`.

References `clear()`, `erased`, and `moves`.

7.55.3.2 `DynamicMotionSequence::DynamicMotionSequence (const char * filename) [inline, explicit]`

constructor, loads from a file and then resets the playtime to beginning and begins to play

Definition at line 15 of file `DynamicMotionSequence.h`.

References `clear()`, `erased`, `LoadSave::LoadFile()`, `moves`, and `MotionSequence::set-PlayTime()`.

7.55.3.3 `virtual DynamicMotionSequence::~~DynamicMotionSequence () [inline, virtual]`

destructor

Definition at line 17 of file `DynamicMotionSequence.h`.

7.55.4 Member Function Documentation

7.55.4.1 `virtual void DynamicMotionSequence::clear () [inline, virtual]`

empties out the sequence (constant time operation - faster than a series of pops)

Implements [MotionSequence](#).

Definition at line 58 of file DynamicMotionSequence.h.

References `erased`, `MotionSequence::invalid_move`, `moves`, `MotionSequence::nexts`, `ERS210Info::NumOutputs`, `MotionSequence::prevs`, `MotionSequence::setPlayTime()`, and `MotionSequence::starts`.

7.55.4.2 **virtual void DynamicMotionSequence::eraseKeyFrame (Move_idx_t x)** [inline, protected, virtual]

causes subclass to mark the corresponding Move structure as free

Implements [MotionSequence](#).

Definition at line 94 of file DynamicMotionSequence.h.

References `erased`.

7.55.4.3 **virtual const Move& DynamicMotionSequence::getKeyFrame (Move_idx_t x) const** [inline, protected, virtual]

returns the Move struct corresponding to *x* in the subclass's actual data structure

Implements [MotionSequence](#).

Definition at line 83 of file DynamicMotionSequence.h.

References `moves`.

7.55.4.4 **virtual Move& DynamicMotionSequence::getKeyFrame (Move_idx_t x)** [inline, protected, virtual]

returns the Move struct corresponding to *x* in the subclass's actual data structure

Implements [MotionSequence](#).

Definition at line 82 of file DynamicMotionSequence.h.

References `moves`.

7.55.4.5 **virtual unsigned int DynamicMotionSequence::getMaxFrames () const** [inline, virtual]

returns the maximum number of key frames (Move's) which can be stored, determined by the instantiating MotionSequenceMC's template parameter

Implements [MotionSequence](#).

Definition at line 71 of file DynamicMotionSequence.h.

7.55.4.6 virtual unsigned int DynamicMotionSequence::getUsedFrames () const
[inline, virtual]

returns the number of used key frames (Move's) which have been stored by the instantiation [MotionSequence](#) subclass

Implements [MotionSequence](#).

Definition at line 72 of file DynamicMotionSequence.h.

References [erased](#), and [moves](#).

7.55.4.7 virtual Move_idx_t DynamicMotionSequence::newKeyFrame ()
[inline, protected, virtual]

causes subclass to create a new Move structure, returns its index

Implements [MotionSequence](#).

Definition at line 84 of file DynamicMotionSequence.h.

References [erased](#), [MotionSequence::Move_idx_t](#), and [moves](#).

7.55.4.8 void DynamicMotionSequence::setRange (unsigned int t, Move_idx_t &prev, Move_idx_t &next) const [inline, protected, virtual]

Sets prev and next to the appropriate values for the given time and output index.

Implements [MotionSequence](#).

Definition at line 95 of file DynamicMotionSequence.h.

References [MotionSequence::invalid_move](#), and [moves](#).

7.55.4.9 virtual int DynamicMotionSequence::updateOutputs () [inline, virtual]

is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)

Returns:

zero if no changes were made, non-zero otherwise

See also:

[RobotInfo::NumFrames](#)

[RobotInfo::FrameTime](#)

Reimplemented from [MotionSequence](#).

Definition at line 33 of file DynamicMotionSequence.h.

References [MotionSequence::calcOutput\(\)](#), [ERS210Info::FrameTime](#), [MotionSequence::getOutputCmd\(\)](#), [MotionSequence::invalid_move](#), [MotionSequence::isPlaying\(\)](#), [motman](#), [MotionSequence::Move_idx_t](#), [moves](#), [MotionSequence::nexts](#), [ERS210Info::NumFrames](#), [ERS210Info::NumOutputs](#), [MotionSequence::playtime](#), [MotionSequence::prevs](#), [MotionManager::setOutput\(\)](#), [setRange\(\)](#), [OutputCmd::unset\(\)](#), and [MotionSequence::updateOutputs\(\)](#).

7.55.5 Member Data Documentation

7.55.5.1 `std::vector<Move_idx_t> DynamicMotionSequence::erased` [protected]

recycles erased keyframes, can't just shift elements in [moves](#), it would throw off index numbers in Move structures

Definition at line 80 of file DynamicMotionSequence.h.

7.55.5.2 `list_t DynamicMotionSequence::moves` [protected]

stores all of the movement keyframes

Definition at line 79 of file DynamicMotionSequence.h.

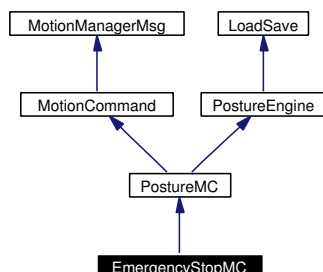
The documentation for this class was generated from the following file:

- [DynamicMotionSequence.h](#)

7.56 EmergencyStopMC Class Reference

```
#include <EmergencyStopMC.h>
```

Inheritance diagram for EmergencyStopMC:



7.56.1 Detailed Description

overrides all joints with high priority freeze, blinks tail pink/red/blue cycle

Uses MotionCommand::kEmergencyPriority. Monitors the feedback on joints and adjusts joints to react to pressures above a certain threshold. This allows you to mold the body while it's in this mode, while retaining enough stiffness to hold against gravity.

This [MotionCommand](#) is intended to always be left running. It polls [WorldState::state](#) for a double-tap on the back button, which causes it to set its joint values. to whatever their current state is. LEDs are left blank, except the tail, which is used to indicate that the emergency stop is on.

A second double-tap will cause it to set all joints to MotionCommand::unusedJoint

The tail LEDs only ever go up to .5, so that if you really care whether the tail light was set by an underlying behavior/motion, you should be able to tell by looking closely (if blue is going from .5 to 1, that's because it's already set)

Definition at line 24 of file EmergencyStopMC.h.

Public Member Functions

- [EmergencyStopMC \(\)](#)
constructor
- virtual [~EmergencyStopMC \(\)](#)
destructor

- virtual int [updateOutputs](#) ()
checks for feedback or double tap
- virtual void [takeSnapshot](#) ()
records current positions of joints
- void [setActive](#) (bool a)
allows you to modify [active](#)
- bool [getActive](#) ()
returns [active](#)
- void [setStopped](#) (bool p, bool sound=true)
allows you to modify [paused](#)
- bool [getStopped](#) () const
returns [paused](#)
- void [setDbtTapDuration](#) (unsigned int d)
sets [duration](#)
- unsigned int [getDbtTapDuration](#) () const
returns [duration](#)
- void [setResetSensitivity](#) (float r)
takes a value [0,1] to set [pidcutoff](#)
- float [getResetSensitivity](#) ()
returns a value [0-1], corresponding to [pidcutoff](#)

Protected Member Functions

- void [freezeJoints](#) ()
code to execute when locking joints
- void [releaseJoints](#) ()
code to execute when releasing joints

Protected Attributes

- bool [paused](#)
true if the joints are current locked up
- bool [stilldown](#)
true if the back button was down on last updateJointCmds
- bool [active](#)
true if the EmergencyStopMC is monitoring the back button (if false, won't pause on a double-tap)
- unsigned int [period](#)
period of cycles on tail LEDs
- unsigned int [timeoflastbtn](#)
time of the last button press
- unsigned int [timeofthisbtn](#)
time of the current button press
- unsigned int [timeoflastfreeze](#)
the time estop was last turned on
- unsigned int [duration](#)
the maximum time (in milliseconds) of consecutive button-down's to count as a double tap
- unsigned char [piddutyavgs](#) [NumPIDJoints]
a running average of PID feedback ("duty"), so one bad reading doesn't cause a movement, need a consistent pressure
- unsigned char [pidcutoff](#)
abs pid duty cycle above which we just reset joint to current
- [LedEngine](#) [ledengine](#)
used to do LED effects on the tail

7.56.2 Constructor & Destructor Documentation

7.56.2.1 EmergencyStopMC::EmergencyStopMC ()

constructor

Definition at line 11 of file EmergencyStopMC.cc.

References `LedEngine::cycle()`, `WorldState::ERS210Mask`, `WorldState::ERS220Mask`, `ledengine`, `ERS210Info::NumPIDJoints`, `period`, `piddutyavgs`, `WorldState::robotDesign`, `MotionCommand::setAutoPrune()`, and `state`.

7.56.2.2 `virtual EmergencyStopMC::~EmergencyStopMC () [inline, virtual]`

destructor

Definition at line 27 of file EmergencyStopMC.h.

7.56.3 Member Function Documentation

7.56.3.1 `void EmergencyStopMC::freezeJoints () [protected]`

code to execute when locking joints

Definition at line 101 of file EmergencyStopMC.cc.

References `EventBase::activateETID`, `PostureEngine::cmds`, `WorldState::ERS210Mask`, `WorldState::ERS220Mask`, `EventBase::estopEGID`, `MotionManagerMsg::getID()`, `ERS210Info::LEDOffset`, `ERS210Info::NumLEDs`, `ERS210Info::NumOutputs`, `MotionCommand::postEvent()`, `WorldState::robotDesign`, `OutputCmd::set()`, `state`, `takeSnapshot()`, `OutputCmd::unset()`, and `OutputCmd::weight`.

7.56.3.2 `bool EmergencyStopMC::getActive () [inline]`

returns [active](#)

Definition at line 33 of file EmergencyStopMC.h.

References [active](#).

7.56.3.3 `unsigned int EmergencyStopMC::getDbtTapDuration () const [inline]`

returns [duration](#)

Definition at line 37 of file EmergencyStopMC.h.

References [duration](#).

7.56.3.4 float EmergencyStopMC::getResetSensitivity () [inline]

returns a value [0-1], corresponding to [pidcutoff](#)

Definition at line 39 of file EmergencyStopMC.h.

References [pidcutoff](#).

7.56.3.5 bool EmergencyStopMC::getStopped () const [inline]

returns [paused](#)

Definition at line 35 of file EmergencyStopMC.h.

References [paused](#).

7.56.3.6 void EmergencyStopMC::releaseJoints () [protected]

code to execute when releasing joints

Definition at line 120 of file EmergencyStopMC.cc.

References [PostureEngine::cmds](#), [EventBase::deactivateETID](#), [WorldState::ERS210Mask](#), [WorldState::ERS220Mask](#), [EventBase::estopEGID](#), [get_time\(\)](#), [MotionManagerMsg::getID\(\)](#), [ERS210Info::LEDOffset](#), [motman](#), [ERS210Info::NumLEDs](#), [ERS210Info::NumOutputs](#), [MotionCommand::postEvent\(\)](#), [WorldState::robotDesign](#), [MotionManager::setOutput\(\)](#), [state](#), [timeoflastfreeze](#), and [OutputCmd::unset\(\)](#).

7.56.3.7 void EmergencyStopMC::setActive (bool *a*)

allows you to modify [active](#)

Definition at line 67 of file EmergencyStopMC.cc.

References [active](#), [freezeJoints\(\)](#), [paused](#), and [releaseJoints\(\)](#).

7.56.3.8 void EmergencyStopMC::setDbtTapDuration (unsigned int *d*) [inline]

sets [duration](#)

Definition at line 36 of file EmergencyStopMC.h.

References [duration](#).

7.56.3.9 void EmergencyStopMC::setResetSensitivity (float *r*) [inline]

takes a value [0,1] to set [pidcutoff](#)

Definition at line 38 of file EmergencyStopMC.h.

References [pidcutoff](#).

7.56.3.10 void EmergencyStopMC::setStopped (bool *p*, bool *sound* = true)

allows you to modify [paused](#)

Definition at line 78 of file EmergencyStopMC.cc.

References [active](#), [config](#), [PostureMC::dirty](#), [Config::motion_config::estop_off_snd](#), [Config::motion_config::estop_on_snd](#), [freezeJoints\(\)](#), [get_time\(\)](#), [Config::motion](#), [paused](#), [SoundManager::PlayFile\(\)](#), [releaseJoints\(\)](#), [sndman](#), and [timeoflastfreeze](#).

7.56.3.11 void EmergencyStopMC::takeSnapshot () [virtual]

records current positions of joints

Reimplemented from [PostureMC](#).

Definition at line 61 of file EmergencyStopMC.cc.

References [PostureEngine::cmds](#), [PostureMC::dirty](#), [ERS210Info::NumOutputs](#), [WorldState::outputs](#), [state](#), and [OutputCmd::value](#).

7.56.3.12 int EmergencyStopMC::updateOutputs () [virtual]

checks for feedback or double tap

Reimplemented from [PostureMC](#).

Definition at line 32 of file EmergencyStopMC.cc.

References [ERS210Info::BackButOffset](#), [WorldState::button_times](#), [PostureEngine::cmds](#), [duration](#), [get_time\(\)](#), [ledengine](#), [ERS210Info::LEDOffset](#), [ERS210Info::NumPIDJoints](#), [paused](#), [pidcutoff](#), [WorldState::pidduties](#), [piddutyavgs](#), [setStopped\(\)](#), [state](#), [stilldown](#), [timeoflastbtn](#), [timeofthisbtn](#), [LedEngine::updateLEDs\(\)](#), [PostureMC::updateOutputs\(\)](#), and [OutputCmd::value](#).

7.56.4 Member Data Documentation

7.56.4.1 `bool EmergencyStopMC::active` [protected]

true if the EmergencyStopMC is monitoring the back button (if false, won't pause on a double-tap)

Definition at line 47 of file EmergencyStopMC.h.

7.56.4.2 `unsigned int EmergencyStopMC::duration` [protected]

the maximum time (in milliseconds) of consecutive button-down's to count as a double tap

Definition at line 52 of file EmergencyStopMC.h.

7.56.4.3 `LedEngine EmergencyStopMC::ledengine` [protected]

used to do LED effects on the tail

Definition at line 55 of file EmergencyStopMC.h.

7.56.4.4 `bool EmergencyStopMC::paused` [protected]

true if the joints are current locked up

Definition at line 45 of file EmergencyStopMC.h.

7.56.4.5 `unsigned int EmergencyStopMC::period` [protected]

period of cycles on tail LEDs

Definition at line 48 of file EmergencyStopMC.h.

7.56.4.6 `unsigned char EmergencyStopMC::pidcutoff` [protected]

abs pid duty cycle above which we just reset joint to current

Definition at line 54 of file EmergencyStopMC.h.

7.56.4.7 `unsigned char EmergencyStopMC::piddutyavgs[NumPIDJoints]` [protected]

a running average of PID feedback ("duty"), so one bad reading doesn't cause a movement, need a consistent pressure

Definition at line 53 of file EmergencyStopMC.h.

7.56.4.8 bool [EmergencyStopMC::stilldown](#) [protected]

true if the back button was down on last updateJointCmds

Definition at line 46 of file EmergencyStopMC.h.

7.56.4.9 unsigned int [EmergencyStopMC::timeoflastbtn](#) [protected]

time of the last button press

Definition at line 49 of file EmergencyStopMC.h.

7.56.4.10 unsigned int [EmergencyStopMC::timeoflastfreeze](#) [protected]

the time estop was last turned on

Definition at line 51 of file EmergencyStopMC.h.

7.56.4.11 unsigned int [EmergencyStopMC::timeofthisbtn](#) [protected]

time of the current button press

Definition at line 50 of file EmergencyStopMC.h.

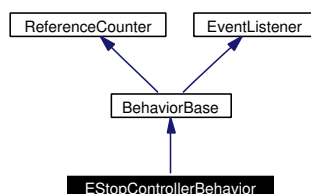
The documentation for this class was generated from the following files:

- [EmergencyStopMC.h](#)
- [EmergencyStopMC.cc](#)

7.57 EStopControllerBehavior Class Reference

```
#include <EStopControllerBehavior.h>
```

Inheritance diagram for EStopControllerBehavior:



7.57.1 Detailed Description

Listens to control commands coming in from the command port for remotely controlling the head.

Definition at line 14 of file EStopControllerBehavior.h.

Public Member Functions

- [EStopControllerBehavior](#) ([MotionManager::MC_ID](#) estop)
constructor
- virtual [~EStopControllerBehavior](#) ()
destructor
- virtual void [DoStart](#) ()
By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.
- virtual void [DoStop](#) ()
By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).
- virtual void [processEvent](#) (const [EventBase](#) &)
Allows you to get away with not supplying a [processEvent\(\)](#) function for the [EventListener](#) interface. By default, does nothing.
- virtual std::string [getName](#) () const

Identifies the behavior in menus and such.

- virtual void [runCommand](#) (std::string s)

Static Public Member Functions

- int [callback](#) (char *buf, int bytes)
called by wireless when there's new data
- std::string [getClassDescription](#) ()
Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Static Public Attributes

- [EStopControllerBehavior](#) * [theOne](#) = NULL

Protected Attributes

- [Socket](#) * [cmdsock](#)
The input command stream socket.
- [MotionManager::MC_ID](#) [estop_id](#)
The estop to control.

Private Member Functions

- [EStopControllerBehavior](#) (const [EStopControllerBehavior](#) &)
don't call
- [EStopControllerBehavior](#) operator= (const [EStopControllerBehavior](#) &)
don't call

7.57.2 Constructor & Destructor Documentation

7.57.2.1 [EStopControllerBehavior::EStopControllerBehavior](#) ([MotionManager::MC_ID](#) *estop*) [inline]

constructor

Definition at line 25 of file EStopControllerBehavior.h.

References cmdsock, estop_id, SocketNS::SOCK_STREAM, theOne, and wireless.

7.57.2.2 `virtual EStopControllerBehavior::~~EStopControllerBehavior ()`
[inline, virtual]

destructor

Definition at line 33 of file EStopControllerBehavior.h.

7.57.2.3 `EStopControllerBehavior::EStopControllerBehavior (const`
`EStopControllerBehavior &) [private]`

don't call

7.57.3 Member Function Documentation

7.57.3.1 `int EStopControllerBehavior::callback (char * buf, int bytes)`
[static]

called by wireless when there's new data

Definition at line 53 of file EStopControllerBehavior.cc.

References runCommand(), and theOne.

7.57.3.2 `void EStopControllerBehavior::DoStart () [virtual]`

By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.

Reimplemented from [BehaviorBase](#).

Definition at line 7 of file EStopControllerBehavior.cc.

References EventRouter::addListener(), callback(), cmdsock, config, BehaviorBase::DoStart(), erouter, Config::main_config::estopControl_port, EventBase::estop-EGID, Wireless::listen(), Config::main, Wireless::setReceiver(), Socket::sock, and wireless.

7.57.3.3 `void EStopControllerBehavior::DoStop () [virtual]`

By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).

Reimplemented from [BehaviorBase](#).

Definition at line 17 of file EStopControllerBehavior.cc.

References [Wireless::close\(\)](#), [cmdsock](#), [BehaviorBase::DoStop\(\)](#), [erouter](#), [EventRouter::forgetListener\(\)](#), and [wireless](#).

7.57.3.4 `std::string EStopControllerBehavior::getClassDescription ()` [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 42 of file EStopControllerBehavior.h.

References [config](#), [Config::main_config::estopControl_port](#), and [Config::main](#).

7.57.3.5 `virtual std::string EStopControllerBehavior::getName () const` [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 41 of file EStopControllerBehavior.h.

7.57.3.6 `EStopControllerBehavior EStopControllerBehavior::operator= (const EStopControllerBehavior &) [private]`

don't call

7.57.3.7 `void EStopControllerBehavior::processEvent (const EventBase & e)` [virtual]

Allows you to get away with not supplying a [processEvent\(\)](#) function for the [Event-Listener](#) interface. By default, does nothing.

Reimplemented from [BehaviorBase](#).

Definition at line 44 of file EStopControllerBehavior.cc.

References [EventBase::activateETID](#), [cmdsock](#), [EventBase::deactivateETID](#), [EventBase::getTypeID\(\)](#), and [Socket::printf\(\)](#).

7.57.3.8 `void EStopControllerBehavior::runCommand (std::string s)` [virtual]

Definition at line 26 of file `EStopControllerBehavior.cc`.

References `MMAccessor< MC_t >::checkin()`, `cmdsock`, `estop_id`, `Socket::printf()`, and `serr`.

7.57.4 Member Data Documentation

7.57.4.1 `Socket* EStopControllerBehavior::cmdsock` [protected]

The input command stream socket.

Definition at line 52 of file `EStopControllerBehavior.h`.

7.57.4.2 `MotionManager::MC_ID EStopControllerBehavior::estop_id` [protected]

The estop to control.

Definition at line 55 of file `EStopControllerBehavior.h`.

7.57.4.3 `EStopControllerBehavior * EStopControllerBehavior::theOne = NULL` [static]

Points to the one `EStopControllerBehavior` object that the input command stream is talking to. A kludge. Dunno how you're gonna make sure you're not using this uninitialized.

Definition at line 5 of file `EStopControllerBehavior.cc`.

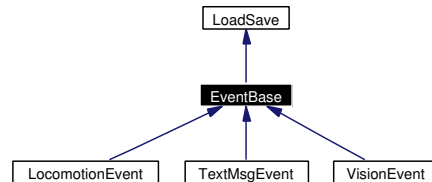
The documentation for this class was generated from the following files:

- [EStopControllerBehavior.h](#)
- [EStopControllerBehavior.cc](#)

7.58 EventBase Class Reference

```
#include <EventBase.h>
```

Inheritance diagram for EventBase:



7.58.1 Detailed Description

The basis of events passed around the high level system.

Contains the list of 'known' event generators in EventGeneratorID.t and EventGeneratorNames[]. If you want to make a new generator, all you have to do is add a new entry to the ID list (EventGeneratorID.t) and then put it's name in the EventGeneratorNames[] array.

Alternatively, there is an 'unlicensed spectrum' available under the unknownEGID. You can send out events from that generator just like any other, but it should only be used for quick tests and hacking around...

The SourceID is generator specific. A SID of *i* from the button generator will refer to a particular button, whereas a SID from vision refers to seeing a particular object. These SIDs are usually defined in the generators themselves. See the EventGeneratorID.t list for links to the generators.

The duration field is also generator specific - some may refer to the time since the last activation event (e.g. button events) where as others refer to time since last status (e.g. sensors updates)

Definition at line 31 of file EventBase.h.

Public Types

- enum [EventGeneratorID.t](#) {
[unknownEGID](#) = 0, [visionEGID](#), [buttonEGID](#), [worldModelEGID](#),
[aiEGID](#), [audioEGID](#), [sensorEGID](#), [powerEGID](#),
[timerEGID](#), [stateMachineEGID](#), [locomotionEGID](#), [textmsgEGID](#),
[estopEGID](#), [motmanEGID](#), [numEGIDs](#) }

Lists all possible event generator ids.

- enum `EventTypeID_t` { `activateETID`, `statusETID`, `deactivateETID`, `numETIDs` }

an event type id is used to denote whether it's the first in a sequence (button down), in a sequence (button still down), or last (button up)

Public Member Functions

Constructors/Destructors

- `EventBase ()`
constructor
- `EventBase (EventGeneratorID_t gid, unsigned int sid, EventTypeID_t tid, unsigned int dur=0)`
constructor
- `EventBase (EventGeneratorID_t gid, unsigned int sid, EventTypeID_t tid, unsigned int dur, const std::string &n, float mag)`
constructor
- virtual `~EventBase ()`
destructor

Methods

- virtual const std::string & `getName ()` const
gets the name of the event - useful for debugging, outputs
- virtual `EventBase & setName (const std::string &n)`
sets name to a given string, prevents overwriting by generated names
- virtual float `getMagnitude ()` const
gets "strength" of event - you might have useful values... used by AI
- virtual `EventBase & setMagnitude (float m)`
sets "strength" of event - but you might have useful values... used by AI
- virtual unsigned int `getTimeStamp ()` const
time event was created
- virtual `EventGeneratorID_t getGeneratorID ()` const
gets the generator ID for this event

- virtual [EventBase](#) & [setGeneratorID](#) ([EventGeneratorID_t](#) gid)
sets the generator ID for this event
- virtual unsigned int [getSourceID](#) () const
gets the source ID for this event
- virtual [EventBase](#) & [setSourceID](#) (unsigned int gid)
sets the source ID for this event
- virtual [EventTypeID_t](#) [getTypeID](#) () const
gets the type ID
- virtual [EventBase](#) & [setTypeID](#) ([EventTypeID_t](#) tid)
sets the type ID
- virtual unsigned int [getDuration](#) () const
OPTIONAL gets the time since the beginning of this sequence (the timestamp of the activate event).
- virtual [EventBase](#) & [setDuration](#) (unsigned int d)
OPTIONAL gets the time since the beginning of this sequence (the timestamp of the activate event).
- virtual const std::string & [resetName](#) ()
resets name to generated form, overwriting any previous name
- virtual bool [isCustomName](#) () const
returns true if not using the generated name
- bool [operator<](#) (const [EventBase](#) &e) const
gets the name of the event - useful for debugging, outputs
- virtual bool [operator==](#) (const [EventBase](#) &eb) const
is true if the genID, typeID, and sourceID's all match
- bool [sameGenSource](#) (const [EventBase](#) eb) const
tests to see if events have the same generator and source IDs
- bool [longerThan](#) (const [EventBase](#) eb) const
compares event duration and ensures same event generator, source, and type - useful for event masks
- bool [shorterThan](#) (const [EventBase](#) eb) const
compares event duration and ensures same event generator, source, and type - useful for event masks
- bool [equalOrLongerThan](#) (const [EventBase](#) eb) const

compares event duration and ensures same event generator, source, and type - useful for event masks

- bool [equalOrShorterThan](#) (const [EventBase](#) eb) const
compares event duration and ensures same event generator, source, and type - useful for event masks

LoadSave interface

Useful for serializing events to send between processes

- virtual unsigned int [getBinSize](#) () const
calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.
- virtual unsigned int [LoadBuffer](#) (const char buf[], unsigned int len)
Load from a saved buffer.
- virtual unsigned int [SaveBuffer](#) (char buf[], unsigned int len) const
Save to a given buffer.

Static Public Attributes

- const char *const [EventGeneratorNames](#) [numEGIDs]
Holds string versions of each of the generator's names, handy for debugging so you can output the events as readable strings.

Protected Member Functions

- virtual void [genName](#) ()
This does the actual generation of names based on genID, sourceID, and typeID.

Protected Attributes

- std::string [stim_id](#)
the name of the event, use the same name consistently or else will be seen as different stimuli
- float [magnitude](#)
the current "strength" of the event/stimuli... MAKE SURE this gets set to ZERO IF event is DEACTIVATE

- unsigned int [timestamp](#)
the time the event was created - set automatically by constructor
- bool [nameisgen](#)
tracks whether the current name (stim_id) is generated or set
- [EventGeneratorID_t](#) [genID](#)
generator ID, EventGeneratorID_t
- [EventTypeID_t](#) [typeID](#)
type ID, EventTypeID_t
- unsigned int [sourceID](#)
the source ID for this event Source IDs are defined by the generator that made it
- unsigned int [duration](#)
the time since this sequence started (like, how long the button has been pressed) ideally, this would be 0 for activate, (activate.timestamp-get_time()) for status and deactivate

7.58.2 Member Enumeration Documentation

7.58.2.1 enum [EventBase::EventGeneratorID_t](#)

Lists all possible event generator ids.

An event generator is a abstract source of events, used for listening to and parsing certain classes of events

IF YOU ADD AN EVENT GENERATOR, DON'T FORGET TO NAME IT ([EventBase::EventGeneratorNames](#), below)

Enumeration values:

- unknownEGID** default EGID, used if you forget to set it, probably not a good idea to use this for anything except errors or testing quick hacks
- visionEGID** vision processing from camera
- buttonEGID** button up, down, and still down, [ButtonSourceID::ButtonSourceID_t](#)
- worldModelEGID** not being used, yet (for when objects are detected/lost?)
- aiEGID** not being used, yet (might use this when AI makes decisions?)
- audioEGID** Sends an event when a sound starts/ends playback, status events as chained sounds end.

sensorEGID currently only used to alert of new sensor readings, may also want to generate IR proximity warnings? [SensorSourceID::SensorSourceID_t](#)

powerEGID used to generate low power warnings, temperature, etc. [PowerSourceID::PowerSourceID_t](#)

timerEGID EGID from which timers are sent, you set timers, but you don't have to also listen `EventRouter::setTimer()`.

stateMachineEGID Sends an event upon entering and leaving a [StateNode](#).

locomotionEGID Sends events regarding transportation in the world; you can/should assume these will all be [LocomotionEvent](#) classes.

textmsgEGID Sends events when a text msg is received on console.

estopEGID Sends an event when the estop is turned on or off.

motmanEGID Sends events when a [MotionCommand](#) is added or removed.

numEGIDs the number of generators available

Definition at line 37 of file `EventBase.h`.

7.58.2.2 enum [EventBase::EventTypeID_t](#)

an event type id is used to denote whether it's the first in a sequence (button down), in a sequence (button still down), or last (button up)

Enumeration values:

activateETID e.g. button down

statusETID e.g. button still down

deactivateETID e.g. button up

numETIDs the number of different event types

Definition at line 59 of file `EventBase.h`.

7.58.3 Constructor & Destructor Documentation

7.58.3.1 [EventBase::EventBase \(\)](#)

constructor

See also:

[EventRouter::postEvent\(\)](#)

Definition at line 22 of file `EventBase.cc`.

References `genName()`, and `get_time()`.

7.58.3.2 EventBase::EventBase ([EventGeneratorID_t](#) *gid*, unsigned int *sid*, [EventTypeID_t](#) *tid*, unsigned int *dur* = 0)

constructor

See also:

[EventRouter::postEvent\(\)](#)

Definition at line 28 of file EventBase.cc.

References deactivateETID, genName(), get_time(), and setMagnitude().

7.58.3.3 EventBase::EventBase ([EventGeneratorID_t](#) *gid*, unsigned int *sid*, [EventTypeID_t](#) *tid*, unsigned int *dur*, const std::string & *n*, float *mag*)

constructor

See also:

[EventRouter::postEvent\(\)](#)

Definition at line 38 of file EventBase.cc.

References get_time().

7.58.3.4 virtual EventBase::~EventBase () [inline, virtual]

destructor

Definition at line 72 of file EventBase.h.

7.58.4 Member Function Documentation

7.58.4.1 bool EventBase::equalOrLongerThan (const [EventBase](#) *eb*) const [inline]

compares event duration and ensures same event generator, source, and type - useful for event masks

Definition at line 110 of file EventBase.h.

References duration, and operator==().

7.58.4.2 **bool EventBase::equalOrShorterThan (const [EventBase](#) eb) const** [inline]

compares event duration and ensures same event generator, source, and type - useful for event masks

Definition at line 111 of file EventBase.h.

References duration, and operator==().

7.58.4.3 **void EventBase::genName ()** [protected, virtual]

This does the actual generation of names based on genID, sourceID, and typeID.

Definition at line 114 of file EventBase.cc.

References EventGeneratorNames, genID, nameisgen, numEGIDs, sourceID, and stim_id.

7.58.4.4 **unsigned int EventBase::getBinSize () const** [virtual]

calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.

Returns:

number of bytes read/written, 0 if error (or empty)

Implements [LoadSave](#).

Reimplemented in [LocomotionEvent](#), [TextMsgEvent](#), and [VisionEvent](#).

Definition at line 46 of file EventBase.cc.

References LoadSave::creatorSize(), duration, magnitude, nameisgen, sourceID, stim_id, LoadSave::stringpad, and timestamp.

7.58.4.5 **virtual unsigned int EventBase::getDuration () const** [inline, virtual]

OPTIONAL gets the time since the beginning of this sequence (the timestamp of the activate event).

See also:

[duration](#)

Definition at line 93 of file EventBase.h.

References duration.

7.58.4.6 virtual [EventGeneratorID_t](#) EventBase::getGeneratorID () const
[inline, virtual]

gets the generator ID for this event

See also:

[EventGeneratorID_t](#)

Definition at line 84 of file EventBase.h.

References EventGeneratorID_t, and genID.

7.58.4.7 virtual float EventBase::getMagnitude () const [inline, virtual]

gets "strength" of event - you might have useful values... used by AI

Definition at line 79 of file EventBase.h.

References magnitude.

7.58.4.8 virtual const std::string& EventBase::getName () const [inline, virtual]

gets the name of the event - useful for debugging, outputs

Definition at line 76 of file EventBase.h.

References stim_id.

7.58.4.9 virtual unsigned int EventBase::getSourceID () const [inline, virtual]

gets the source ID for this event

See also:

[sourceID](#)

Definition at line 87 of file EventBase.h.

References sourceID.

7.58.4.10 virtual unsigned int EventBase::getTimeStamp () const [inline, virtual]

time event was created

Definition at line 82 of file EventBase.h.

References timestamp.

7.58.4.11 `virtual EventTypeID_t EventBase::getTypeID () const` [inline, virtual]

gets the type ID

See also:

[EventTypeID_t](#)

Definition at line 90 of file EventBase.h.

References EventTypeID_t, and typeID.

7.58.4.12 `virtual bool EventBase::isCustomName () const` [inline, virtual]

returns true if not using the generated name

Definition at line 97 of file EventBase.h.

References nameisgen.

7.58.4.13 `unsigned int EventBase::LoadBuffer (const char buf [], unsigned int len)` [virtual]

Load from a saved buffer.

Parameters:

buf pointer to the memory where you should begin loading

len length of *buf* available (this isn't all yours, might be more stuff saved after yours)

Returns:

the number of bytes actually used

Implements [LoadSave](#).

Reimplemented in [LocomotionEvent](#), [TextMsgEvent](#), and [VisionEvent](#).

Definition at line 61 of file EventBase.cc.

References [LoadSave::checkCreator\(\)](#), [LoadSave::decode\(\)](#), [duration](#), [Event-GeneratorID_t](#), [EventTypeID_t](#), [genID](#), [magnitude](#), [nameisgen](#), [sourceID](#), [stim.id](#), [timestamp](#), and [typeID](#).

7.58.4.14 `bool EventBase::longerThan (const EventBase eb) const` [inline]

compares event duration and ensures same event generator, source, and type - useful for event masks

Definition at line 108 of file EventBase.h.

References duration, and operator==().

7.58.4.15 `bool EventBase::operator< (const EventBase & e) const` [inline]

gets the name of the event - useful for debugging, outputs

Definition at line 99 of file EventBase.h.

References timestamp.

7.58.4.16 `virtual bool EventBase::operator== (const EventBase & eb) const`
[inline, virtual]

is true if the genID, typeID, and sourceID's all match

Definition at line 102 of file EventBase.h.

References genID, sourceID, and typeID.

7.58.4.17 `virtual const std::string& EventBase::resetName ()` [inline, virtual]

resets name to generated form, overwriting any previous name

Definition at line 96 of file EventBase.h.

References genName(), nameisgen, and stim_id.

7.58.4.18 `bool EventBase::sameGenSource (const EventBase eb) const`
[inline]

tests to see if events have the same generator and source IDs

Definition at line 106 of file EventBase.h.

References genID, and sourceID.

7.58.4.19 `unsigned int EventBase::SaveBuffer (char buf[], unsigned int len)`
`const` [virtual]

Save to a given buffer.

Parameters:

buf pointer to the memory where you should begin writing

len length of *buf* available. (this isn't all yours, constrain yourself to what you returned in [getBinSize\(\)](#))

Returns:

the number of bytes actually used

Implements [LoadSave](#).

Reimplemented in [LocomotionEvent](#), [TextMsgEvent](#), and [VisionEvent](#).

Definition at line 89 of file `EventBase.cc`.

References [duration](#), [LoadSave::encode\(\)](#), [genID](#), [magnitude](#), [nameisgen](#), [LoadSave::saveCreator\(\)](#), [sourceID](#), [stim_id](#), [timestamp](#), and [typeID](#).

7.58.4.20 `virtual EventBase& EventBase::setDuration (unsigned int d)`
[inline, virtual]

OPTIONAL gets the time since the beginning of this sequence (the timestamp of the activate event).

See also:

[duration](#)

Definition at line 94 of file `EventBase.h`.

References [duration](#).

7.58.4.21 `virtual EventBase& EventBase::setGeneratorID`
`(EventGeneratorID_t gid)` [inline, virtual]

sets the generator ID for this event

See also:

[EventGeneratorID_t](#)

Definition at line 85 of file `EventBase.h`.

References [genID](#), and [genName\(\)](#).

7.58.4.22 `virtual EventBase& EventBase::setMagnitude (float m)` [`inline`, `virtual`]

sets "strength" of event - but you might have useful values... used by AI

Definition at line 80 of file EventBase.h.

References `magnitude`.

7.58.4.23 `virtual EventBase& EventBase::setName (const std::string & n)` [`inline`, `virtual`]

sets name to a given string, prevents overwriting by generated names

Definition at line 77 of file EventBase.h.

References `nameisgen`, and `stim_id`.

7.58.4.24 `virtual EventBase& EventBase::setSourceID (unsigned int gid)` [`inline`, `virtual`]

sets the source ID for this event

See also:

[sourceID](#)

Definition at line 88 of file EventBase.h.

References `genName()`, and `sourceID`.

7.58.4.25 `virtual EventBase& EventBase::setTypeID (EventTypeID_t tid)` [`inline`, `virtual`]

sets the type ID

See also:

[EventTypeID_t](#)

Definition at line 91 of file EventBase.h.

References `genName()`, and `typeID`.

7.58.4.26 `bool EventBase::shorterThan (const EventBase eb) const` [`inline`]

compares event duration and ensures same event generator, source, and type - useful for event masks

Definition at line 109 of file EventBase.h.

References `duration`, and `operator==()`.

7.58.5 Member Data Documentation

7.58.5.1 unsigned int `EventBase::duration` [protected]

the time since this sequence started (like, how long the button has been pressed) ideally, this would be 0 for activate, (`activate.timestamp-get_time()`) for status and deactivate

Definition at line 134 of file EventBase.h.

7.58.5.2 const char *const `EventBase::EventGeneratorNames` [static]

Initial value:

```
{
    "UnknownGen",
    "Vision",
    "Button",
    "WorldModel",
    "AI",
    "Audio",
    "Sensor",
    "Power",
    "Timer",
    "StateMachine",
    "Locomotion",
    "TextMsg",
    "EStop",
    "MotionManager"
}
```

Holds string versions of each of the generator's names, handy for debugging so you can output the events as readable strings.

Definition at line 4 of file EventBase.cc.

7.58.5.3 `EventGeneratorID_t EventBase::genID` [protected]

generator ID, `EventGeneratorID_t`

Definition at line 128 of file EventBase.h.

7.58.5.4 float [EventBase::magnitude](#) [protected]

the current "strength" of the event/stimuli... MAKE SURE this gets set to ZERO IF event is DEACTIVATE

Definition at line 122 of file EventBase.h.

7.58.5.5 bool [EventBase::nameisgen](#) [protected]

tracks whether the current name (stim_id) is generated or set

Definition at line 125 of file EventBase.h.

7.58.5.6 unsigned int [EventBase::sourceID](#) [protected]

the source ID for this event Source IDs are defined by the generator that made it

This should give authors flexibility to design their modules without having to worry about ID space collision

Definition at line 130 of file EventBase.h.

7.58.5.7 std::string [EventBase::stim_id](#) [protected]

the name of the event, use the same name consistently or else will be seen as different stimuli

Definition at line 121 of file EventBase.h.

7.58.5.8 unsigned int [EventBase::timestamp](#) [protected]

the time the event was created - set automatically by constructor

Definition at line 123 of file EventBase.h.

7.58.5.9 [EventTypeID_t](#) [EventBase::typeID](#) [protected]

type ID, EventTypeID_t

Definition at line 129 of file EventBase.h.

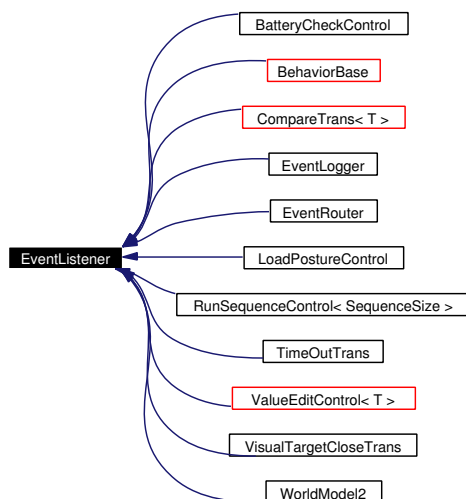
The documentation for this class was generated from the following files:

- [EventBase.h](#)
- [EventBase.cc](#)

7.59 EventListener Class Reference

```
#include <EventListener.h>
```

Inheritance diagram for EventListener:



7.59.1 Detailed Description

An interface to allow a standard method of passing events.

Definition at line 7 of file EventListener.h.

Public Member Functions

- virtual [~EventListener](#) ()
destructor
- virtual void [processEvent](#) (const [EventBase](#) &event)=0
for receiving events - you must override this to inherit

7.59.2 Constructor & Destructor Documentation

7.59.2.1 virtual [EventListener::~EventListener](#) () [inline, virtual]

destructor

Definition at line 10 of file EventListener.h.

7.59.3 Member Function Documentation

7.59.3.1 virtual void EventListener::processEvent (const [EventBase](#) & event) [pure virtual]

for receiving events - you must override this to inherit

See also:

[EventRouter](#)

Parameters:

event the event being received

Implemented in [BehaviorBase](#), [Controller](#), [BatteryCheckControl](#), [EventLogger](#), [FreeMemReportControl](#), [LoadPostureControl](#), [RunSequenceControl](#)< [SequenceSize](#) >, [ValueEditControl](#)< [T](#) >, [AutoGetupBehavior](#), [BanditMachine::WaitNode](#), [BatteryMonitorBehavior](#), [CameraBehavior](#), [ChaseBallBehavior](#), [DriveMeBehavior](#), [DumbWM2Behavior](#), [EStopControllerBehavior](#), [EvtRptBehavior](#), [FollowHeadBehavior](#), [HeadLevelBehavior](#), [HeadPointControllerBehavior](#), [SimpleChaseBallBehavior](#), [SoundTestBehavior](#), [StareAtBallBehavior](#), [WalkControllerBehavior](#), [WalkToTargetMachine](#), [WorldModel2Behavior::WalkNode](#), [WorldModel2Behavior::GawkNode](#), [StateNode](#), [CompareTrans](#)< [T](#) >, [SmoothCompareTrans](#)< [T](#) >, [TimeOutTrans](#), [VisualTargetCloseTrans](#), [EventRouter](#), [StartupBehavior](#), and [WorldModel2](#).

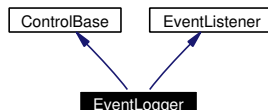
The documentation for this class was generated from the following file:

- [EventListener.h](#)

7.60 EventLogger Class Reference

```
#include <EventLogger.h>
```

Inheritance diagram for EventLogger:



7.60.1 Detailed Description

allows logging of events to the console or a file

Definition at line 10 of file EventLogger.h.

Public Member Functions

- [EventLogger](#) ()
constructor
- virtual [ControlBase](#) * [doSelect](#) ()
opens a custom (embedded) menu to toggle individual EGIDs
- virtual void [refresh](#) ()
called when the child has died and this control should refresh its display
- virtual void [processEvent](#) (const [EventBase](#) &event)
sends all events received to stdout and/or logfile

Protected Member Functions

- void [setStatus](#) (unsigned int i, char c)
sets the status char of slot i to c
- void [checkLogFile](#) ()
checks to see if logfilePath differs from the StringInputControl's value and switches it if it is

Protected Attributes

- `std::string` [logfilePath](#)
address of the logfile, if any (empty string is no logfile)
- `std::ofstream` [logfile](#)
if a filename is given, events are logged to here
- `unsigned int` [verbosity](#)
controls the level of verbosity - currently 0 through 2

7.60.2 Constructor & Destructor Documentation

7.60.2.1 EventLogger::EventLogger ()

constructor

Definition at line 10 of file EventLogger.cc.

References `ControlBase::ControlBase()`, `EventBase::EventGeneratorNames`, `EventBase::numEGIDs`, `ControlBase::pushSlot()`, and `verbosity`.

7.60.3 Member Function Documentation

7.60.3.1 void EventLogger::checkLogFile () [protected]

checks to see if `logfilePath` differs from the `StringInputControl`'s value and switches it if it is

Definition at line 95 of file EventLogger.cc.

References `ASSERTRET`, `StringInputControl::getLastInput()`, `ControlBase::getName()`, `logfile`, `logfilePath`, `EventBase::numEGIDs`, `ControlBase::options`, `ControlBase::setName()`, and `setStatus()`.

7.60.3.2 [ControlBase](#) * EventLogger::doSelect () [virtual]

opens a custom (embedded) menu to toggle individual EGIDs

Reimplemented from [ControlBase](#).

Definition at line 21 of file EventLogger.cc.

References `EventRouter::addListener()`, `config`, `Config::controller`, `erouter`, `EventBase::EventGeneratorID_t`, `ControlBase::hilights`, `logfile`, `EventBase::numEGIDs`,

ControlBase::options, SoundManager::PlayFile(), refresh(), EventRouter::remove-Listener(), Config::controller_config::select_snd, setStatus(), and sndman.

7.60.3.3 void EventLogger::processEvent (const [EventBase](#) & event) [virtual]

sends all events received to stdout and/or logfile

Implements [EventListener](#).

Definition at line 62 of file EventLogger.cc.

References [EventBase::activateETID](#), [checkLogFile\(\)](#), [EventBase::deactivateETID](#), [EventBase::getDuration\(\)](#), [EventBase::getMagnitude\(\)](#), [EventBase::getName\(\)](#), [EventBase::getTimeStamp\(\)](#), [EventBase::getTypeID\(\)](#), [logfile](#), [EventBase::numEGIDs](#), [EventBase::numETIDs](#), [ControlBase::options](#), [EventBase::statusETID](#), and [verbosity](#).

7.60.3.4 void EventLogger::refresh () [virtual]

called when the child has died and this control should refresh its display

Reimplemented from [ControlBase](#).

Definition at line 56 of file EventLogger.cc.

References [checkLogFile\(\)](#), and [ControlBase::refresh\(\)](#).

7.60.3.5 void EventLogger::setStatus (unsigned int *i*, char *c*) [protected]

sets the status char of slot *i* to *c*

Definition at line 89 of file EventLogger.cc.

References [ControlBase::options](#).

7.60.4 Member Data Documentation

7.60.4.1 std::ofstream [EventLogger::logfile](#) [protected]

if a filename is given, events are logged to here

Definition at line 34 of file EventLogger.h.

7.60.4.2 std::string [EventLogger::logfilePath](#) [protected]

address of the logfile, if any (empty string is no logfile)

Definition at line 31 of file EventLogger.h.

7.60.4.3 unsigned int [EventLogger::verbosity](#) [protected]

controls the level of verbosity - currently 0 through 2

Definition at line 37 of file EventLogger.h.

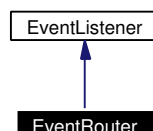
The documentation for this class was generated from the following files:

- [EventLogger.h](#)
- [EventLogger.cc](#)

7.61 EventRouter Class Reference

```
#include <EventRouter.h>
```

Inheritance diagram for EventRouter:



7.61.1 Detailed Description

This class will handle distribution of events as well as management of timers.

Classes must inherit from [EventListener](#) and/or [EventTrapper](#) in order to receive events.

Use the global [erouter](#) EventRouter to post and subscribe to events

When multiple listeners are subscribed, the order in which an event is distributed among them looks like this:

1. "Specific" listeners: any listener which specifies a particular source id. (It doesn't matter if they specify a type id or not.)
 - older listeners get events before younger listeners
2. "General" listeners: those that subscribe to an entire generator
 - older listeners get events before younger listeners

...but if you're relying on that ordering, there should be a cleaner way to do whatever you're doing.

If one behavior unsubscribes another one during a [processEvent\(\)](#), that behavior will still get the "current" event before the unsubscription takes place.

Buffering events has not been tested thoroughly...

See also:

[EventBase::EventGeneratorID.t](#) for a complete listing of all generators, as well as instructions on how to add new generators.

Definition at line 40 of file EventRouter.h.

Public Member Functions

- [EventRouter](#) ()
Constructs the router, [buffertime](#) defaults to 1.
- virtual [~EventRouter](#) ()
just calls [reset](#) and [removeAllTimers\(\)](#)
- void [reset](#) ()
erases all listeners, trappers and timers, resets EventRouter
- void [setBufferTime](#) (unsigned int t)
sets the time to wait between buffer clears, see [EventRouter::buffertime](#)
- unsigned int [getBufferTime](#) ()
returns the time to wait between buffer clears, see [EventRouter::buffertime](#)

Posting/Processing Events

- void [postEvent](#) ([EventBase::EventGeneratorID_t](#) egid, unsigned int sid, [EventBase::EventTypeID_t](#) etid, unsigned int dur)
recommended to create and post an event using current buffer setting
- void [postEvent](#) ([EventBase::EventGeneratorID_t](#) egid, unsigned int sid, [EventBase::EventTypeID_t](#) etid, unsigned int dur, const std::string &n, float m)
recommended to create and post an event using current buffer setting
- void [postEvent](#) ([EventBase](#) *e)
recommended to create and post an event using current buffer setting
- void [processTimers](#) ()
determines if timers need to be posted, and posts them if so.
- void [processEventBuffer](#) ()
clears the event buffer, deletes events as it does so.
- void [processEvent](#) (const [EventBase](#) &e)
*forces unbuffered - sends event *now*. Will clear the buffer first if needed to ensure proper event ordering*

Listener Detection

- bool [hasListeners](#) ([EventBase::EventGeneratorID_t](#) egid)

counts both listeners and trappers, so stuff can tell if it even needs to bother generating an event...

- bool `hasListeners (EventBase::EventGeneratorID_t egid, unsigned int sid)`
counts both listeners and trappers, so stuff can tell if it even needs to bother generating an event...
- bool `hasListeners (EventBase::EventGeneratorID_t egid, unsigned int sid, EventBase::EventTypeID_t etid)`
counts both listeners and trappers, so stuff can tell if it even needs to bother generating an event...

Timer Management

- void `addTimer (EventListener *el, unsigned int sid, unsigned int delay, bool repeat=true)`
adds a timer or sets a timer if it doesn't already exist.
- void `addTimer (EventListener *el, const EventBase &e, bool repeat=true)`
calls the other `addTimer()` with the event's source id and duration, doesn't check to see if the generator is timerEGID
- void `removeTimer (EventListener *el)`
clears all pending timers for listener el
- void `removeTimer (EventListener *el, unsigned int sid)`
clears any pending timers with source id sid for listener el
- void `removeAllTimers ()`
clears all timers for all listeners

Listener Management

- void `addListener (EventListener *el, const EventBase &e)`
Adds a listener for a specific source id and type from a given event generator, adding a Timer event will invoke `addTimer(el, e.getSourceID(), e.getDuration(), true)`.
- void `addListener (EventListener *el, EventBase::EventGeneratorID_t egid)`
Adds a listener for all events from a given event generator.
- void `addListener (EventListener *el, EventBase::EventGeneratorID_t egid, unsigned int sid)`
Adds a listener for all types from a specific source and generator.

- void `addListener` (`EventListener` *el, `EventBase::EventGeneratorID_t` egid, unsigned int sid, `EventBase::EventTypeID_t` etid)
Adds a listener for a specific source id and type from a given event generator.
- void `removeListener` (`EventListener` *el, const `EventBase` &e)
stops sending specified events from the generator to the listener. If a timer is passed it will invoke `removeTimer`(el, e.getSourceID())
- void `removeListener` (`EventListener` *el, `EventBase::EventGeneratorID_t` egid)
stops sending specified events from the generator to the listener.
- void `removeListener` (`EventListener` *el, `EventBase::EventGeneratorID_t` egid, unsigned int sid)
stops sending specified events from the generator to the listener.
- void `removeListener` (`EventListener` *el, `EventBase::EventGeneratorID_t` egid, unsigned int sid, `EventBase::EventTypeID_t` etid)
stops sending specified events from the generator to the listener.
- void `removeListener` (`EventListener` *el)
stops sending ALL events to the listener
- void `forgetListener` (`EventListener` *el)
clears timers and removes listener from all events

Trapper Management

- void `addTrapper` (`EventTrapper` *el, const `EventBase` &e)
Adds a trapper for a specific source id and type from a given event generator.
- void `addTrapper` (`EventTrapper` *el, `EventBase::EventGeneratorID_t` egid)
Adds a trapper for all events from a given event generator.
- void `addTrapper` (`EventTrapper` *el, `EventBase::EventGeneratorID_t` egid, unsigned int sid)
Adds a trapper for all types from a specific source and generator.
- void `addTrapper` (`EventTrapper` *el, `EventBase::EventGeneratorID_t` egid, unsigned int sid, `EventBase::EventTypeID_t` etid)
Adds a trapper for a specific source id and type from a given event generator.
- void `addTrapper` (`EventTrapper` *el)
adds a trapper for ALL events
- void `removeTrapper` (`EventTrapper` *el, const `EventBase` &e)

stops sending specified events from the generator to the trapper.

- void `removeTrapper` (`EventTrapper *el`, `EventBase::EventGeneratorID_t egid`)

stops sending specified events from the generator to the trapper.

- void `removeTrapper` (`EventTrapper *el`, `EventBase::EventGeneratorID_t egid`, `unsigned int sid`)

stops sending specified events from the generator to the trapper.

- void `removeTrapper` (`EventTrapper *el`, `EventBase::EventGeneratorID_t egid`, `unsigned int sid`, `EventBase::EventTypeID_t etid`)

stops sending specified events from the generator to the trapper.

- void `removeTrapper` (`EventTrapper *el`)

stops sending ALL events to the trapper

Protected Types

- typedef `std::vector< TimerEntry * >::iterator timer_it_t`

makes code more readable

Protected Member Functions

- void `doSendBuffer` ()
does the work of clearing the buffer
- void `doSendEvent` (`const EventBase &e`)
does the work of sending an event

- void `chkTimers` ()
just for debugging

- void `dispTimers` ()
just for debugging

Protected Attributes

- `std::vector< TimerEntry * > timers`

the list of timer entries being maintained, kept sorted by time they go active

- `std::vector< EventBase * > events`
used to store buffered events
- `bool doSendBufferLock`
in case of recursive calls to [processEventBuffer\(\)](#)/[doSendBuffer\(\)](#)
- `unsigned int lastBufClear`
time of last event buffer clear
- `unsigned int buffertime`
The time between clearings of the buffer.
- [EventMapper](#) trappers
A mapping of which [EventTrapper](#)'s should get a chance to trap the event.
- [EventMapper](#) listeners
A mapping of which [EventListener](#)'s should receive events.

7.61.2 Member Typedef Documentation

7.61.2.1 `typedef std::vector<TimerEntry*>::iterator EventRouter::timer_it_t [protected]`

makes code more readable

Definition at line 146 of file `EventRouter.h`.

7.61.3 Constructor & Destructor Documentation

7.61.3.1 `EventRouter::EventRouter ()`

Constructs the router, [buffertime](#) defaults to 1.

Definition at line 7 of file `EventRouter.cc`.

7.61.3.2 `virtual EventRouter::~EventRouter () [inline, virtual]`

just calls reset and [removeAllTimers\(\)](#)

Definition at line 43 of file `EventRouter.h`.

References [removeAllTimers\(\)](#), and [reset\(\)](#).

7.61.4 Member Function Documentation

7.61.4.1 `void EventRouter::addListener (EventListener * el,
EventBase::EventGeneratorID_t egid, unsigned int sid,
EventBase::EventTypeID_t etid)`

Adds a listener for a specific source id and type from a given event generator.

Definition at line 82 of file EventRouter.cc.

References EventRouter::EventManager::addMapping(), and listeners.

7.61.4.2 `void EventRouter::addListener (EventListener * el,
EventBase::EventGeneratorID_t egid, unsigned int sid)`

Adds a listener for all types from a specific source and generator.

Definition at line 78 of file EventRouter.cc.

References EventRouter::EventManager::addMapping(), EventBase::EventTypeID_t, listeners, and EventBase::numETIDs.

7.61.4.3 `void EventRouter::addListener (EventListener * el,
EventBase::EventGeneratorID_t egid)`

Adds a listener for all events from a given event generator.

Definition at line 75 of file EventRouter.cc.

References EventRouter::EventManager::addMapping(), and listeners.

7.61.4.4 `void EventRouter::addListener (EventListener * el, const EventBase &
e)`

Adds a listener for a specific source id and type from a given event generator, adding a Timer event will invoke addTimer(*el*, *e.getSourceID()*, *e.getDuration()*, true).

Definition at line 69 of file EventRouter.cc.

References EventRouter::EventManager::addMapping(), addTimer(), EventBase::getDuration(), EventBase::getGeneratorID(), EventBase::getSourceID(), EventBase::getTypeID(), listeners, and EventBase::timerEGID.

7.61.4.5 void EventRouter::addTimer (EventListener * *el*, const EventBase & *e*, bool *repeat* = true) [inline]

calls the other [addTimer\(\)](#) with the event's source id and duration, doesn't check to see if the generator is timerEGID

Definition at line 74 of file EventRouter.h.

References [addTimer\(\)](#), [EventBase::getDuration\(\)](#), and [EventBase::getSourceID\(\)](#).

7.61.4.6 void EventRouter::addTimer (EventListener * *el*, unsigned int *sid*, unsigned int *delay*, bool *repeat* = true)

adds a timer or sets a timer if it doesn't already exist.

timers are unique by [EventListener](#) and source ID - can't have two timers for the same *el* and *sid* a delay of 0 with repeating will cause an event to be sent at every opportunity, use very sparingly a delay of -1U will call [removeTimer\(\)](#) if it already exists, otherwise is ignored

Parameters:

- el* the [EventListener](#) to send the timer event to
- sid* the source ID to use on that event (if you need to send more info, send a pointer to a struct of your devising, typecasted as int)
- delay* the delay between the first (and future) calls
- repeat* set to true if you want to keep receiving this event, otherwise it will only send once

Definition at line 48 of file EventRouter.cc.

References [removeTimer\(\)](#), [timer_it_t](#), and [timers](#).

7.61.4.7 void EventRouter::addTrapper (EventTrapper * *el*) [inline]

adds a trapper for ALL events

Definition at line 102 of file EventRouter.h.

References [EventRouter::EventManager::addMapping\(\)](#), [EventBase::EventGeneratorID_t](#), [EventBase::numEGIDs](#), and [trappers](#).

7.61.4.8 void EventRouter::addTrapper (EventTrapper * *el*, EventBase::EventGeneratorID_t *egid*, unsigned int *sid*, EventBase::EventTypeID_t *etid*) [inline]

Adds a trapper for a specific source id and type from a given event generator.

Definition at line 100 of file EventRouter.h.

References EventRouter::EventManager::addMapping(), and trappers.

7.61.4.9 void EventRouter::addTrapper (EventTrapper * el, EventBase::EventGeneratorID_t egid, unsigned int sid) [inline]

Adds a trapper for all types from a specific source and generator.

Definition at line 99 of file EventRouter.h.

References EventRouter::EventManager::addMapping(), EventBase::EventTypeID_t, EventBase::numETIDs, and trappers.

7.61.4.10 void EventRouter::addTrapper (EventTrapper * el, EventBase::EventGeneratorID_t egid) [inline]

Adds a trapper for all events from a given event generator.

Definition at line 98 of file EventRouter.h.

References EventRouter::EventManager::addMapping(), and trappers.

7.61.4.11 void EventRouter::addTrapper (EventTrapper * el, const EventBase & e) [inline]

Adds a trapper for a specific source id and type from a given event generator.

Definition at line 97 of file EventRouter.h.

References EventRouter::EventManager::addMapping(), EventBase::getGeneratorID(), EventBase::getSourceID(), EventBase::getTypeID(), and trappers.

7.61.4.12 void EventRouter::chkTimers () [inline, protected]

just for debugging

Definition at line 150 of file EventRouter.h.

References dispTimers(), timer_it_t, and timers.

7.61.4.13 void EventRouter::dispTimers () [inline, protected]

just for debugging

Definition at line 161 of file EventRouter.h.

References get_time(), timer_it_t, and timers.

7.61.4.14 void EventRouter::doSendBuffer () [protected]

does the work of clearing the buffer

Definition at line 123 of file EventRouter.cc.

References doSendBufferLock, doSendEvent(), events, get_time(), and lastBufClear.

7.61.4.15 void EventRouter::doSendEvent (const EventBase & e) [protected]

does the work of sending an event

Definition at line 148 of file EventRouter.cc.

References EventRouter::EventManager::getMapping(), listeners, and trappers.

7.61.4.16 void EventRouter::forgetListener (EventListener * el) [inline]

clears timers and removes listener from all events

Definition at line 93 of file EventRouter.h.

References removeListener(), and removeTimer().

7.61.4.17 unsigned int EventRouter::getBufferTime () [inline]

returns the time to wait between buffer clears, see [EventRouter::buffertime](#)

Returns:

time to wait between buffer clears

See also:

[buffertime](#)

Definition at line 47 of file EventRouter.h.

References buffertime.

7.61.4.18 bool EventRouter::hasListeners (EventBase::EventGeneratorID_t egid, unsigned int sid, EventBase::EventTypeID_t etid) [inline]

counts both listeners and trappers, so stuff can tell if it even needs to bother generating an event...

Definition at line 69 of file EventRouter.h.

References EventRouter::EventManager::hasMapping(), listeners, and trappers.

7.61.4.19 `bool EventRouter::hasListeners (EventBase::EventGeneratorID_t egid, unsigned int sid)` [inline]

counts both listeners and trappers, so stuff can tell if it even needs to bother generating an event...

Definition at line 68 of file EventRouter.h.

References EventRouter::EventManager::hasMapping(), listeners, and trappers.

7.61.4.20 `bool EventRouter::hasListeners (EventBase::EventGeneratorID_t egid)` [inline]

counts both listeners and trappers, so stuff can tell if it even needs to bother generating an event...

Definition at line 67 of file EventRouter.h.

References EventRouter::EventManager::hasMapping(), listeners, and trappers.

7.61.4.21 `void EventRouter::postEvent (EventBase * e)` [inline]

recommended to create and post an event using current buffer setting

Definition at line 53 of file EventRouter.h.

References buffertime, events, and processEvent().

7.61.4.22 `void EventRouter::postEvent (EventBase::EventGeneratorID_t egid, unsigned int sid, EventBase::EventTypeID_t etid, unsigned int dur, const std::string & n, float m)` [inline]

recommended to create and post an event using current buffer setting

Definition at line 52 of file EventRouter.h.

References buffertime, events, and processEvent().

7.61.4.23 `void EventRouter::postEvent (EventBase::EventGeneratorID_t egid, unsigned int sid, EventBase::EventTypeID_t etid, unsigned int dur)` [inline]

recommended to create and post an event using current buffer setting

Definition at line 51 of file EventRouter.h.

References buffertime, events, and processEvent().

7.61.4.24 void EventRouter::processEvent (const [EventBase](#) & *e*) [virtual]

forces unbuffered - sends event *now*. Will clear the buffer first if needed to ensure proper event ordering

Implements [EventListener](#).

Definition at line 142 of file EventRouter.cc.

References [doSendBuffer\(\)](#), [doSendEvent\(\)](#), and [events](#).

7.61.4.25 void EventRouter::processEventBuffer ()

clears the event buffer, deletes events as it does so.

Definition at line 119 of file EventRouter.cc.

References [doSendBuffer\(\)](#), and [events](#).

7.61.4.26 void EventRouter::processTimers ()

determines if timers need to be posted, and posts them if so.

Call this often to ensure accurate timers. Also, will clear event buffer before sending timer events in order to ensure correct ordering of event reception

Definition at line 12 of file EventRouter.cc.

References [buffertime](#), [get_time\(\)](#), [lastBufClear](#), [processEventBuffer\(\)](#), [sort\(\)](#), [EventBase::statusETID](#), [timer_it_t](#), [EventBase::timerEGID](#), and [timers](#).

7.61.4.27 void EventRouter::removeAllTimers ()

clears all timers for all listeners

Definition at line 113 of file EventRouter.cc.

References [timer_it_t](#), and [timers](#).

7.61.4.28 void EventRouter::removeListener ([EventListener](#) * *el*) [inline]

stops sending ALL events to the listener

Definition at line 92 of file EventRouter.h.

References [EventRouter::EventManager::clean\(\)](#), [EventBase::EventGeneratorID_t](#), [listeners](#), [EventBase::numEGIDs](#), and [EventRouter::EventManager::removeMapping\(\)](#).

7.61.4.29 `void EventRouter::removeListener (EventListener * el,
 EventBase::EventGeneratorID_t egid, unsigned int sid,
 EventBase::EventTypeID_t etid)` [inline]

stops sending specified events from the generator to the listener.

Definition at line 90 of file EventRouter.h.

References EventRouter::EventManager::clean(), listeners, and EventRouter::EventManager::removeMapping().

7.61.4.30 `void EventRouter::removeListener (EventListener * el,
 EventBase::EventGeneratorID_t egid, unsigned int sid)` [inline]

stops sending specified events from the generator to the listener.

Definition at line 89 of file EventRouter.h.

References EventRouter::EventManager::clean(), EventBase::EventTypeID_t, listeners, EventBase::numETIDs, and EventRouter::EventManager::removeMapping().

7.61.4.31 `void EventRouter::removeListener (EventListener * el,
 EventBase::EventGeneratorID_t egid)` [inline]

stops sending specified events from the generator to the listener.

Definition at line 88 of file EventRouter.h.

References EventRouter::EventManager::clean(), listeners, and EventRouter::EventManager::removeMapping().

7.61.4.32 `void EventRouter::removeListener (EventListener * el, const
 EventBase & e)`

stops sending specified events from the generator to the listener. If a timer is passed it will invoke `removeTimer(el, e.getSourceID())`

Definition at line 86 of file EventRouter.cc.

References EventRouter::EventManager::clean(), EventBase::getGeneratorID(), EventBase::getSourceID(), EventBase::getTypeID(), listeners, EventRouter::EventManager::removeMapping(), `removeTimer()`, and EventBase::timerEGID.

7.61.4.33 `void EventRouter::removeTimer (EventListener * el, unsigned int sid)`

clears any pending timers with source id *sid* for listener *el*

Definition at line 104 of file EventRouter.cc.

References timer_it_t, and timers.

7.61.4.34 void EventRouter::removeTimer ([EventListener](#) * *el*)

clears all pending timers for listener *el*

Definition at line 95 of file EventRouter.cc.

References timer_it_t, and timers.

7.61.4.35 void EventRouter::removeTrapper ([EventTrapper](#) * *el*) [inline]

stops sending ALL events to the trapper

Definition at line 110 of file EventRouter.h.

References EventRouter::EventManager::clean(), EventBase::EventGeneratorID_t, EventBase::numEGIDs, EventRouter::EventManager::removeMapping(), and trappers.

7.61.4.36 void EventRouter::removeTrapper ([EventTrapper](#) * *el*, [EventBase::EventGeneratorID_t](#) *egid*, unsigned int *sid*, [EventBase::EventTypeID_t](#) *etid*) [inline]

stops sending specified events from the generator to the trapper.

Definition at line 108 of file EventRouter.h.

References EventRouter::EventManager::clean(), EventRouter::EventManager::removeMapping(), and trappers.

7.61.4.37 void EventRouter::removeTrapper ([EventTrapper](#) * *el*, [EventBase::EventGeneratorID_t](#) *egid*, unsigned int *sid*) [inline]

stops sending specified events from the generator to the trapper.

Definition at line 107 of file EventRouter.h.

References EventRouter::EventManager::clean(), EventBase::EventTypeID_t, EventBase::numETIDs, EventRouter::EventManager::removeMapping(), and trappers.

7.61.4.38 void EventRouter::removeTrapper ([EventTrapper](#) * *el*, [EventBase::EventGeneratorID_t](#) *egid*) [inline]

stops sending specified events from the generator to the trapper.

Definition at line 106 of file EventRouter.h.

References EventRouter::EventManager::clean(), EventRouter::EventManager::removeMapping(), and trappers.

7.61.4.39 void EventRouter::removeTrapper ([EventTrapper](#) * *el*, const [EventBase](#) & *e*) [inline]

stops sending specified events from the generator to the trapper.

Definition at line 105 of file EventRouter.h.

References EventRouter::EventManager::clean(), EventBase::getGeneratorID(), EventBase::getSourceID(), EventBase::getTypeID(), EventRouter::EventManager::removeMapping(), and trappers.

7.61.4.40 void EventRouter::reset () [inline]

erases all listeners, trappers and timers, resets EventRouter

Definition at line 44 of file EventRouter.h.

References EventRouter::EventManager::clear(), listeners, removeAllTimers(), and trappers.

7.61.4.41 void EventRouter::setBufferTime (unsigned int *t*) [inline]

sets the time to wait between buffer clears, see [EventRouter::buffertime](#)

Parameters:

t time to wait between buffer clears

See also:

[buffertime](#)

Definition at line 46 of file EventRouter.h.

References buffertime.

7.61.5 Member Data Documentation

7.61.5.1 unsigned int [EventRouter::buffertime](#) [protected]

The time between clearings of the buffer.

0 will not use the buffer, events are routed upon posting 1 will clear the buffer at next call to [processTimers\(\)](#) or [processEventBuffer\(\)](#) a larger value will cause a delay of that number of milliseconds since the last clearing

Definition at line 175 of file EventRouter.h.

7.61.5.2 **bool** [EventRouter::doSendBufferLock](#) [protected]

in case of recursive calls to [processEventBuffer\(\)](#)/[doSendBuffer\(\)](#)

Definition at line 173 of file EventRouter.h.

7.61.5.3 **std::vector<EventBase*>** [EventRouter::events](#) [protected]

used to store buffered events

Definition at line 172 of file EventRouter.h.

7.61.5.4 **unsigned int** [EventRouter::lastBufClear](#) [protected]

time of last event buffer clear

Definition at line 174 of file EventRouter.h.

7.61.5.5 **EventMapper** [EventRouter::listeners](#) [protected]

A mapping of which EventListener's should receive events.

Definition at line 235 of file EventRouter.h.

7.61.5.6 **std::vector<TimerEntry*>** [EventRouter::timers](#) [protected]

the list of timer entries being maintained, kept sorted by time they go active

Definition at line 147 of file EventRouter.h.

7.61.5.7 **EventMapper** [EventRouter::trappers](#) [protected]

A mapping of which EventTrapper's should get a chance to trap the event.

Definition at line 234 of file EventRouter.h.

The documentation for this class was generated from the following files:

- [EventRouter.h](#)
- [EventRouter.cc](#)

7.62 EventRouter::EventManager Class Reference

```
#include <EventRouter.h>
```

7.62.1 Detailed Description

Does the actual storage of the mapping between EventBase's and the Event-Listeners/EventTrappers who should receive them.

Actually only stores void*'s, so it's more general than just Listeners or Trappers

Definition at line 182 of file EventRouter.h.

Public Member Functions

- [EventManager](#) ()
constructor
- void [addMapping](#) (void *el, [EventBase::EventGeneratorID_t](#) egid)
Adds a listener for all events from a given event generator.
- void [addMapping](#) (void *el, [EventBase::EventGeneratorID_t](#) egid, unsigned int sid, [EventBase::EventTypeID_t](#) etid)
Adds a listener for a specific source id and type from a given event generator.
- void [removeMapping](#) (void *el, [EventBase::EventGeneratorID_t](#) egid)
Removes a listener for all events from a given event generator.
- void [removeMapping](#) (void *el, [EventBase::EventGeneratorID_t](#) egid, unsigned int sid, [EventBase::EventTypeID_t](#) etid)
Removes a listener for a specific source id and type from a given event generator.
- void [clean](#) ()
compresses empty data structures
- void [clear](#) ()
Resets the mapping.
- template<class T> void [getMapping](#) (const [EventBase](#) &e, std::vector< T * > &listeners)
builds a list of all listeners which should receive the event, templated to typecast the pointers for you

- bool [hasMapping](#) ([EventBase::EventGeneratorID_t](#) egid)
so stuff can tell if it even needs to bother generating an event...
- bool [hasMapping](#) ([EventBase::EventGeneratorID_t](#) egid, unsigned int sid)
so stuff can tell if it even needs to bother generating an event...
- bool [hasMapping](#) ([EventBase::EventGeneratorID_t](#) egid, unsigned int sid, [EventBase::EventTypeID_t](#) etid)
so stuff can tell if it even needs to bother generating an event...

Protected Types

- typedef std::map< unsigned int, std::vector< void * >, std::less< unsigned int > > [SIDtoListenerVectorMap_t](#)
a mapping from source IDs (unsigned ints), each to a vector of pointers to listeners

Protected Attributes

- std::vector< void * > [allevents](#) [[EventBase::numEGIDs](#)]
an array of vectors of pointers to listeners... in other words, a vector of listener pointers for each generator
- [SIDtoListenerVectorMap_t](#) * [filteredevents](#) [[EventBase::numEGIDs](#)][[EventBase::numETIDs](#)]
not for the faint of heart: a matrix of mappings to vectors of pointers to listeners

Private Member Functions

- [EventManager](#) (const [EventManager](#) &)
this shouldn't be called...
- [EventManager](#) operator= (const [EventManager](#) &)
this shouldn't be called...

7.62.2 Member Typedef Documentation

7.62.2.1 `typedef std::map<unsigned int,std::vector<void*>,std::less<unsigned int> > EventRouter::EventManager::SIDtoListenerVectorMap_t`
[protected]

a mapping from source IDs (unsigned ints), each to a vector of pointers to listeners
main use in `filterevents`

See also:

[filterevents](#)

Definition at line 222 of file `EventRouter.h`.

7.62.3 Constructor & Destructor Documentation

7.62.3.1 `EventRouter::EventManager::EventManager ()`

constructor

Definition at line 162 of file `EventRouter.cc`.

References `filterevents`, `EventBase::numEGIDs`, and `EventBase::numETIDs`.

7.62.3.2 `EventRouter::EventManager::EventManager (const EventManager &)`
[private]

this shouldn't be called...

7.62.4 Member Function Documentation

7.62.4.1 `void EventRouter::EventManager::addMapping (void * el,
EventBase::EventGeneratorID_t egid, unsigned int sid,
EventBase::EventTypeID_t etid)`

Adds a listener for a specific source id and type from a given event generator.

Definition at line 168 of file `EventRouter.cc`.

References `filterevents`, and `SIDtoListenerVectorMap_t`.

7.62.4.2 `void EventRouter::EventManager::addMapping (void * el,
EventBase::EventGeneratorID_t egid)` [inline]

Adds a listener for all events from a given event generator.

Definition at line 187 of file EventRouter.h.

References `allevents`.

7.62.4.3 void EventRouter::EventManager::clean ()

compresses empty data structures

Definition at line 212 of file EventRouter.cc.

References `filteredevents`, `EventBase::numEGIDs`, `EventBase::numETIDs`, and `SIDtoListenerVectorMap_t`.

7.62.4.4 void EventRouter::EventManager::clear ()

Resets the mapping.

Definition at line 247 of file EventRouter.cc.

References `filteredevents`, `EventBase::numEGIDs`, `EventBase::numETIDs`, and `SIDtoListenerVectorMap_t`.

7.62.4.5 template<class T> void EventRouter::EventManager::getMapping (const EventBase & e, std::vector< T * > & listeners)

builds a list of all listeners which should receive the event, templated to typecast the pointers for you

Parameters:

e the key event

listeners upon return, the resulting list of listeners *e* maps to

listeners is not cleared prior to building, new listeners are pushed on end

Results are in the order: all specific matches first, all generator listeners second, in order they were added to the [EventManager](#).

Definition at line 302 of file EventRouter.cc.

References `allevents`, `filteredevents`, `EventBase::getGeneratorID()`, `EventBase::getSourceID()`, `EventBase::getTypeID()`, and `SIDtoListenerVectorMap_t`.

7.62.4.6 bool EventRouter::EventManager::hasMapping (EventBase::EventGeneratorID_t egid, unsigned int sid, EventBase::EventTypeID_t etid)

so stuff can tell if it even needs to bother generating an event...

Returns:

true if it has any listeners, false otherwise

Definition at line 289 of file EventRouter.cc.

References allevents, filteredevents, and SIDtoListenerVectorMap_t.

7.62.4.7 **bool EventRouter::EventManager::hasMapping** ([EventBase::EventGeneratorID_t](#) *egid*, unsigned int *sid*)

so stuff can tell if it even needs to bother generating an event...

Returns:

true if it has any listeners, false otherwise

Definition at line 275 of file EventRouter.cc.

References allevents, filteredevents, EventBase::numETIDs, and SIDtoListenerVectorMap_t.

7.62.4.8 **bool EventRouter::EventManager::hasMapping** ([EventBase::EventGeneratorID_t](#) *egid*)

so stuff can tell if it even needs to bother generating an event...

Returns:

true if it has any listeners, false otherwise

Definition at line 260 of file EventRouter.cc.

References allevents, filteredevents, EventBase::numETIDs, and SIDtoListenerVectorMap_t.

7.62.4.9 [EventManager](#) EventRouter::EventManager::operator= (const [EventManager](#) &) [private]

this shouldn't be called...

7.62.4.10 **void EventRouter::EventManager::removeMapping** (void * *el*, [EventBase::EventGeneratorID_t](#) *egid*, unsigned int *sid*, [EventBase::EventTypeID_t](#) *etid*)

Removes a listener for a specific source id and type from a given event generator.

Doesn't necessarily remove the vector or mapping if this was the last listener, use [clean\(\)](#) to do that

Definition at line 201 of file EventRouter.cc.

References [filterevents](#), and [SIDtoListenerVectorMap_t](#).

7.62.4.11 **void EventRouter::EventManager::removeMapping (void * *el*, [EventBase::EventGeneratorID_t](#) *egid*)**

Removes a listener for all events from a given event generator.

Doesn't necessarily remove the vector or mapping if this was the last listener, use [clean\(\)](#) to do that

Definition at line 184 of file EventRouter.cc.

References [allevents](#), [filterevents](#), [EventBase::numETIDs](#), and [SIDtoListenerVectorMap_t](#).

7.62.5 Member Data Documentation

7.62.5.1 **std::vector<void*> [EventRouter::EventManager::allevents](#)[[EventBase::numEGIDs](#)] [protected]**

an array of vectors of pointers to listeners... in other words, a vector of listener pointers for each generator

Definition at line 225 of file EventRouter.h.

7.62.5.2 **[SIDtoListenerVectorMap_t](#)* [EventRouter::EventManager::filterevents](#)[[EventBase::numEGIDs](#)][[EventBase::numETIDs](#)] [protected]**

not for the faint of heart: a matrix of mappings to vectors of pointers to listeners

Definition at line 227 of file EventRouter.h.

The documentation for this class was generated from the following files:

- [EventRouter.h](#)
- [EventRouter.cc](#)

7.63 EventRouter::TimerEntry Struct Reference

```
#include <EventRouter.h>
```

7.63.1 Detailed Description

Contains all the information needed to maintain a timer by the [EventRouter](#).

Definition at line 118 of file EventRouter.h.

Public Member Functions

- [TimerEntry](#) (unsigned int nxt)
constructs an entry using the given value for next - useful for with [TimerEntryPtr-Cmp](#)
- [TimerEntry](#) ([EventListener](#) *e, unsigned int s, unsigned int d, bool r)
constructs with the given values, sets next field automatically next
- [TimerEntry](#) (const [TimerEntry](#) &t)
just does the default, i'm just being explicit since there's a pointer (no deep copy!)
- [TimerEntry](#) operator= (const [TimerEntry](#) &t)
just does the default, i'm just being explicit since there's a pointer (no deep copy!)
- void [Set](#) (unsigned int d, bool r)
will reset timer

Public Attributes

- [EventListener](#) * el
the listener to fire at
- unsigned int [sid](#)
the source id to fire with
- unsigned int [delay](#)
the delay until firing
- unsigned int [next](#)
the time at which this timer will go off next

- bool [repeat](#)

if true, will reset after firing, else will be deleted

7.63.2 Constructor & Destructor Documentation

7.63.2.1 EventRouter::TimerEntry::TimerEntry (unsigned int *nxt*) [inline, explicit]

constructors an entry using the given value for next - useful for with [TimerEntryPtrCmp](#)

Definition at line 120 of file EventRouter.h.

References [delay](#), [el](#), [next](#), [repeat](#), and [sid](#).

7.63.2.2 EventRouter::TimerEntry::TimerEntry ([EventListener](#) * *e*, unsigned int *s*, unsigned int *d*, bool *r*) [inline]

constructs with the given values, sets next field automatically next

Definition at line 122 of file EventRouter.h.

References [delay](#), [el](#), [get_time\(\)](#), [next](#), [repeat](#), and [sid](#).

7.63.2.3 EventRouter::TimerEntry::TimerEntry (const [TimerEntry](#) & *t*) [inline]

just does the default, i'm just being explicit since there's a pointer (no deep copy!)

Definition at line 124 of file EventRouter.h.

References [delay](#), [el](#), [next](#), [repeat](#), and [sid](#).

7.63.3 Member Function Documentation

7.63.3.1 [TimerEntry](#) EventRouter::TimerEntry::operator= (const [TimerEntry](#) & *t*) [inline]

just does the default, i'm just being explicit since there's a pointer (no deep copy!)

Definition at line 126 of file EventRouter.h.

References [delay](#), [el](#), [next](#), [repeat](#), and [sid](#).

7.63.3.2 void EventRouter::TimerEntry::Set (unsigned int *d*, bool *r*) [inline]

will reset timer

Parameters:

d the time from now when the timer should go off (in milliseconds)

r true if the timer should automatically repeat

Definition at line 130 of file EventRouter.h.

References `delay`, `get_time()`, `next`, and `repeat`.

7.63.4 Member Data Documentation

7.63.4.1 unsigned int EventRouter::TimerEntry::delay

the delay until firing

Definition at line 133 of file EventRouter.h.

7.63.4.2 EventListener* EventRouter::TimerEntry::el

the listener to fire at

Definition at line 131 of file EventRouter.h.

7.63.4.3 unsigned int EventRouter::TimerEntry::next

the time at which this timer will go off next

Definition at line 134 of file EventRouter.h.

7.63.4.4 bool EventRouter::TimerEntry::repeat

if true, will reset after firing, else will be deleted

Definition at line 135 of file EventRouter.h.

7.63.4.5 unsigned int EventRouter::TimerEntry::sid

the source id to fire with

Definition at line 132 of file EventRouter.h.

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

- [EventRouter.h](#)

7.64 EventRouter::TimerEntryPtrCmp Class Reference

```
#include <EventRouter.h>
```

7.64.1 Detailed Description

Used by STL to sort the timer list in order of activation time.

See also:

[EventRouter::timers](#)

Definition at line 139 of file EventRouter.h.

Public Member Functions

- bool [operator\(\)](#) (const [TimerEntry](#) *const a, const [TimerEntry](#) *const b) const

Used by STL to sort the timer list in order of activation time timers.

7.64.2 Member Function Documentation

7.64.2.1 bool EventRouter::TimerEntryPtrCmp::operator() (const [TimerEntry](#) *const a, const [TimerEntry](#) *const b) const [inline]

Used by STL to sort the timer list in order of activation time timers.

Since we remove NULLs before sorting, shouldn't need to check here (and I want to know if i'm wrong)

Returns:

(a->next<b->next)

Definition at line 144 of file EventRouter.h.

References [EventRouter::TimerEntry::next](#).

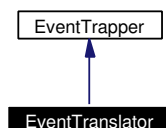
The documentation for this class was generated from the following file:

- [EventRouter.h](#)

7.65 EventTranslator Class Reference

```
#include <EventTranslator.h>
```

Inheritance diagram for EventTranslator:



7.65.1 Detailed Description

EventTranslator receives events from EventRouters in non-Main processes and adds them into a [SharedQueue](#) for Main to pick up.

The [SharedQueue](#) which the processes should set up and then pass to this object is defined by TranslatorSharedQueue.t - it allows 100 entries totalling 3KB of space. You can modify the type to change the capacity, but if you're making that many events that between Main's calls to [translateEvents\(\)](#) you might want to rethink a few things...

EventTranslator only handles [LocomotionEvent](#), [VisionEvent](#), and [TextMsgEvent](#) subtypes for the moment. Anything else is only handled as an [EventBase](#) class. (so extra fields aren't going to be stored) It's easy to add more types if you need to send them from other processes (that's the whole point of this class!)

Reason for doing it this way: Avoids OPENR message lag time (4-8ms in our testing), also avoids problems with RTTI stored in classes from different processes.

Definition at line 21 of file EventTranslator.h.

Public Types

- typedef [SharedQueue](#)< 3 * 1024, 100 > [Queue.t](#)
Use this type to set up the shared queue between processes.
- enum [TypeID.t](#) { [EventBase_ID](#), [LocomotionEvent_ID](#), [VisionEvent_ID](#), [TextMsgEvent_ID](#) }
an ID is inserted before the event data in the queue so we can tell which subclass it is

Public Member Functions

- [EventTranslator](#) ()

constructor

- void [setQueue](#) ([Queue.t](#) *q)
sets the [SharedQueue](#) which should be used
- virtual bool [trapEvent](#) (const [EventBase](#) &event)
called by the event router when something in this process sends an event
- void [translateEvents](#) ()
called whenever Main gets some processor time to check for events from other processes

Static Public Member Functions

- void [enqueue](#) (const [EventBase](#) &event, [Queue.t](#) *q)
called by [trapEvent](#) to do all the work, needed so [MotionCommands](#) can enqueue directly

Static Protected Member Functions

- void [sendEvent](#) (const void *buf, unsigned int size)
called by [translateEvents](#) for each event to be sent

Protected Attributes

- [Queue.t](#) * [queue](#)
pointer to queue of events to be sent

Private Member Functions

- [EventTranslator](#) (const [EventTranslator](#) &)
don't call
- [EventTranslator](#) operator= (const [EventTranslator](#) &)
don't call

7.65.2 Member Typedef Documentation

7.65.2.1 typedef [SharedQueue](#)<3*1024,100> [EventTranslator::Queue_t](#)

Use this type to set up the shared queue between processes.

Definition at line 27 of file EventTranslator.h.

7.65.3 Member Enumeration Documentation

7.65.3.1 enum [EventTranslator::TypeID_t](#)

an ID is inserted before the event data in the queue so we can tell which subclass it is (quickly, could look at the creator code which is stored, but that's a text string)

Enumeration values:

EventBase_ID

LocomotionEvent_ID

VisionEvent_ID

TextMsgEvent_ID

Definition at line 31 of file EventTranslator.h.

7.65.4 Constructor & Destructor Documentation

7.65.4.1 [EventTranslator::EventTranslator](#) () [inline]

constructor

Definition at line 24 of file EventTranslator.h.

References [queue](#).

7.65.4.2 [EventTranslator::EventTranslator](#) (const [EventTranslator](#) &) [private]

don't call

7.65.5 Member Function Documentation

7.65.5.1 `void EventTranslator::enqueue (const EventBase & event, Queue.t * q)` [static]

called by trapEvent to do all the work, needed so MotionCommands can enqueue directly

Definition at line 16 of file EventTranslator.cc.

References `ASSERT`, `SharedQueue< 3 *1024, 100 >::done()`, `EventBase_ID`, `EventBase::getBinSize()`, `LocomotionEvent_ID`, `SharedQueue< 3 *1024, 100 >::reserve()`, `EventBase::SaveBuffer()`, `TextMsgEvent_ID`, `TypeID_t`, and `VisionEvent_ID`.

7.65.5.2 `EventTranslator EventTranslator::operator= (const EventTranslator &)` [private]

don't call

7.65.5.3 `void EventTranslator::sendEvent (const void * buf, unsigned int size)` [static, protected]

called by translateEvents for each event to be sent

Definition at line 51 of file EventTranslator.cc.

References `ASSERT`, `erouter`, `EventBase_ID`, `EventBase::LoadBuffer()`, `TextMsgEvent::LoadBuffer()`, `LocomotionEvent::LoadBuffer()`, `LocomotionEvent_ID`, `EventRouter::postEvent()`, `TextMsgEvent_ID`, `TypeID_t`, `vision`, and `VisionEvent_ID`.

7.65.5.4 `void EventTranslator::setQueue (Queue.t * q)` [inline]

sets the [SharedQueue](#) which should be used

Definition at line 34 of file EventTranslator.h.

References `queue`.

7.65.5.5 `void EventTranslator::translateEvents ()`

called whenever Main gets some processor time to check for events from other processes

Definition at line 42 of file EventTranslator.cc.

References `SharedQueue< 3 *1024, 100 >::clear()`, `SharedQueue< 3 *1024, 100 >::data()`, `queue`, `sendEvent()`, and `SharedQueue< 3 *1024, 100 >::size()`.

7.65.5.6 `bool EventTranslator::trapEvent (const EventBase & event)` [virtual]

called by the event router when something in this process sends an event

Implements [EventTrapper](#).

Definition at line 10 of file EventTranslator.cc.

References [enqueue\(\)](#), and [queue](#).

7.65.6 Member Data Documentation

7.65.6.1 `Queue_t* EventTranslator::queue` [protected]

pointer to queue of events to be sent

Definition at line 50 of file EventTranslator.h.

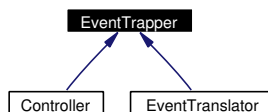
The documentation for this class was generated from the following files:

- [EventTranslator.h](#)
- [EventTranslator.cc](#)

7.66 EventTrapper Class Reference

```
#include <EventTrapper.h>
```

Inheritance diagram for EventTrapper:



7.66.1 Detailed Description

An interface to allow a standard method of trapping events.

Trappers get first dibs on events and can prevent the event from being sent any further

This is handy in situations where an event is more than a notification, and must be "handled" - the trapper which handles it returns true, otherwise it is passed to the next one

A trapper can filter any and all events, EXCEPT timers. This *could* be changed, if a good reason is presented.

Definition at line 12 of file EventTrapper.h.

Public Member Functions

- virtual [~EventTrapper](#) ()
destructor
- virtual bool [trapEvent](#) (const [EventBase](#) &event)=0
for receiving events - you must override this to inherit

7.66.2 Constructor & Destructor Documentation

7.66.2.1 virtual [EventTrapper::~EventTrapper](#) () [inline, virtual]

destructor

Definition at line 15 of file EventTrapper.h.

7.66.3 Member Function Documentation

7.66.3.1 `virtual bool EventTrapper::trapEvent (const EventBase & event)` [pure virtual]

for receiving events - you must override this to inherit

See also:

[EventRouter](#)

Parameters:

event the event being received

Returns:

`true` if the event was trapped (shouldn't be sent to listeners), `false` otherwise

Implemented in [Controller](#), and [EventTranslator](#).

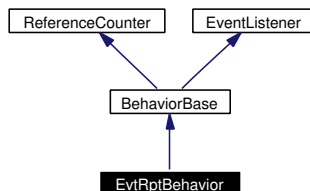
The documentation for this class was generated from the following file:

- [EventTrapper.h](#)

7.67 EvtRptBehavior Class Reference

```
#include <EvtRptBehavior.h>
```

Inheritance diagram for EvtRptBehavior:



7.67.1 Detailed Description

A simple behavior to test event reports.

Definition at line 8 of file EvtRptBehavior.h.

Public Member Functions

- [EvtRptBehavior](#) ()
constructor
- virtual [~EvtRptBehavior](#) ()
destructor
- virtual void [DoStart](#) ()
Subscribes to various events.
- virtual void [DoStop](#) ()
Cancels event subscriptions.
- virtual void [processEvent](#) (const [EventBase](#) &event)
prints out data on subscribed events
- virtual std::string [getName](#) () const
Identifies the behavior in menus and such.

Static Public Member Functions

- `std::string getClassDescription ()`

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

7.67.2 Constructor & Destructor Documentation

7.67.2.1 EvtRptBehavior::EvtRptBehavior () [inline]

constructor

Definition at line 11 of file EvtRptBehavior.h.

7.67.2.2 virtual EvtRptBehavior::~~EvtRptBehavior () [inline, virtual]

destructor

Definition at line 15 of file EvtRptBehavior.h.

7.67.3 Member Function Documentation

7.67.3.1 void EvtRptBehavior::DoStart () [virtual]

Subscribes to various events.

Reimplemented from [BehaviorBase](#).

Definition at line 7 of file EvtRptBehavior.cc.

References [EventManager::addListener\(\)](#), [BehaviorBase::DoStart\(\)](#), [Vision::enableEvents\(\)](#), [erouter](#), [EventBase::locomotionEGID](#), [vision](#), and [EventBase::visionEGID](#).

7.67.3.2 void EvtRptBehavior::DoStop () [virtual]

Cancels event subscriptions.

Reimplemented from [BehaviorBase](#).

Definition at line 18 of file EvtRptBehavior.cc.

References [Vision::disableEvents\(\)](#), [BehaviorBase::DoStop\(\)](#), [erouter](#), [EventManager::forgetListener\(\)](#), and [vision](#).

7.67.3.3 `std::string EvtRptBehavior::getClassDescription ()` [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 27 of file EvtRptBehavior.h.

7.67.3.4 `virtual std::string EvtRptBehavior::getName () const` [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 26 of file EvtRptBehavior.h.

7.67.3.5 `void EvtRptBehavior::processEvent (const EventBase & event)` [virtual]

prints out data on subscribed events

Reimplemented from [BehaviorBase](#).

Definition at line 28 of file EvtRptBehavior.cc.

References [EventBase::getGeneratorID\(\)](#), [EventBase::getSourceID\(\)](#), [EventBase::getTypeID\(\)](#), [EventBase::locomotionEGID](#), [EventBase::statusETID](#), and [EventBase::visionEGID](#).

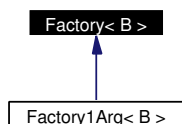
The documentation for this class was generated from the following files:

- [EvtRptBehavior.h](#)
- [EvtRptBehavior.cc](#)

7.68 Factory< B > Class Template Reference

```
#include <Factory.h>
```

Inheritance diagram for Factory< B >:



7.68.1 Detailed Description

```
template<class B> class Factory< B >
```

A lightweight class to override for constructing new objects (if you need to pass constructors parameters, etc.).

Say you don't want to construct your behavior at boot-up (if it's big and might not even be used) but your behavior needs special setup during creation (might be invoked several different ways for instance) then you'll want to subclass this to do the setup when your behavior is activated.

The default is to simply call the default constructor

Definition at line 12 of file Factory.h.

Static Public Member Functions

- B * [construct](#) ()
Just returns a new B.

7.68.2 Member Function Documentation

7.68.2.1 `template<class B> B* Factory< B >::construct () [inline, static]`

Just returns a new B.

Reimplemented in [Factory1Arg< B, A1, a1 >](#).

Definition at line 14 of file Factory.h.

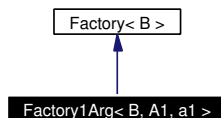
The documentation for this class was generated from the following file:

- [Factory.h](#)

7.69 Factory1Arg< B, A1, a1 > Class Template Reference

```
#include <Factory.h>
```

Inheritance diagram for Factory1Arg< B, A1, a1 >:



7.69.1 Detailed Description

```
template<class B, class A1, A1 a1> class Factory1Arg< B, A1, a1 >
```

Uses template to specify a constant parameter to the constructor.

Definition at line 19 of file Factory.h.

Static Public Member Functions

- B * [construct](#) ()
Just returns a new B constructed with arguments a1.

7.69.2 Member Function Documentation

7.69.2.1 `template<class B, class A1, A1 a1> B* Factory1Arg< B, A1, a1 >::construct () [inline, static]`

Just returns a new B constructed with arguments *a1*.

Reimplemented from [Factory< B >](#).

Definition at line 21 of file Factory.h.

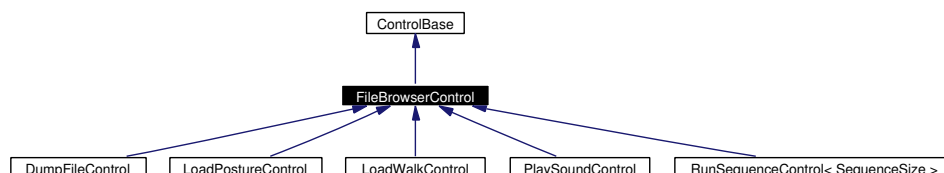
The documentation for this class was generated from the following file:

- [Factory.h](#)

7.70 FileBrowserControl Class Reference

```
#include <FileBrowserControl.h>
```

Inheritance diagram for FileBrowserControl:



7.70.1 Detailed Description

Displays the contents of a directory in a control menu, probably useful as a baseclass for other controls.

Causes the [selectedFile\(\)](#) function to be called on the root FileBrowserControl with the selected file

Definition at line 11 of file FileBrowserControl.h.

Public Member Functions

Constructors/Destructors

constructor

- [FileBrowserControl](#) ()
- [FileBrowserControl](#) (const std::string &nm, const std::string &desc, const std::string &path)

constructor pass name and root path

ControlBase Inheritance

- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#) display, [Socket](#) *gui)
Called when the control is activated (or the control system is reactivating).
- virtual [ControlBase](#) * [doSelect](#) ()
when the user has trigger an "open selection" - default is to return the hilighted control

Accessors

- void [setRecurse](#) (bool r)
sets recurse
- bool [getRecurse](#) () const
returns recurse
- void [setRoot](#) (const std::string &path)
sets root
- std::string [getRoot](#) () const
returns root
- void [setPath](#) (const std::string &path)
sets paths
- void [setFilter](#) (const std::string &filt)
*sets filter; remember can only use one wildcard, e.g. *.ext or filename.ext or filename**

Protected Member Functions

- virtual [ControlBase](#) * [selectedFile](#) (const std::string &)
the big money function - by default calls the parent if it exists, otherwise nothing
- std::string [makePath](#) ()
returns the path from root as a string, with no trailing '/'
- std::string [makePath](#) (const std::string &filename)
returns the path from root as a string, appends filename
- void [rebuildmenu](#) ()
rescans current directory and builds menus
- void [init](#) (std::string path)
sets a junk menu item to mark this as having submenus, and sets root to path

Static Protected Member Functions

- bool [match](#) (const std::string &file, const std::string &filt)
returns true if file matches filt

Protected Attributes

- bool [recurse](#)
if true (default), will show directories; if false, subdirectories are hidden
- std::string [root](#)
the path to browse, default "/"
- std::vector< std::string > [paths](#)
list of directories from root
- std::string [filter](#)
default "", only display matching files; only can use one wildcard, e.g. *.ext or filename.ext or filename**

7.70.2 Constructor & Destructor Documentation

7.70.2.1 FileBrowserControl::FileBrowserControl () [inline]

Definition at line 15 of file FileBrowserControl.h.

References [filter](#), [init\(\)](#), [paths](#), [recurse](#), and [root](#).

7.70.2.2 FileBrowserControl::FileBrowserControl (const std::string & nm, const std::string & desc, const std::string & path) [inline]

constructor pass name and root path

Definition at line 17 of file FileBrowserControl.h.

References [filter](#), [init\(\)](#), [paths](#), [recurse](#), and [root](#).

7.70.3 Member Function Documentation

7.70.3.1 [ControlBase](#) * FileBrowserControl::activate ([MotionManager::MC_ID](#) display, [Socket](#) * gui) [virtual]

Called when the control is activated (or the control system is reactivating).

Takes the id number of a [LedMC](#) which the control should use, maintained by [Controller](#). Controls share the display which is passed, and may use the socket *gui* to communicate with the GUI controller, if it is connected.

Returns:

a [ControlBase](#) pointer. Return:

- *this* if the control should stay active (if it's not a one-shot command)
- NULL to return to parent
- other address to spawn a child control

Reimplemented from [ControlBase](#).

Definition at line 7 of file FileBrowserControl.cc.

References [ControlBase::activate\(\)](#), and [rebuildmenu\(\)](#).

7.70.3.2 [ControlBase](#) * FileBrowserControl::doSelect () [virtual]

when the user has trigger an "open selection" - default is to return the hilighted control

Reimplemented from [ControlBase](#).

Definition at line 12 of file FileBrowserControl.cc.

References [ControlBase::doSelect\(\)](#), [ControlBase::getName\(\)](#), [ControlBase::hilights](#), [makePath\(\)](#), [ControlBase::options](#), [paths](#), [rebuildmenu\(\)](#), [ControlBase::refresh\(\)](#), and [selectedFile\(\)](#).

7.70.3.3 bool FileBrowserControl::getRecurse () const [inline]

returns [recurse](#)

Definition at line 27 of file FileBrowserControl.h.

References [recurse](#).

7.70.3.4 std::string FileBrowserControl::getRoot () const [inline]

returns [root](#)

Definition at line 30 of file FileBrowserControl.h.

References [root](#).

7.70.3.5 void FileBrowserControl::init (std::string path) [inline, protected]

sets a junk menu item to mark this as having submenus, and sets root to path

Definition at line 56 of file FileBrowserControl.h.

References [ControlBase::pushSlot\(\)](#), and [setRoot\(\)](#).

7.70.3.6 `std::string FileBrowserControl::makePath (const std::string & filename)` [protected]

returns the path from root as a string, appends filename

Definition at line 66 of file FileBrowserControl.cc.

References `makePath()`.

7.70.3.7 `std::string FileBrowserControl::makePath ()` [protected]

returns the path from root as a string, with no trailing '/'

Definition at line 56 of file FileBrowserControl.cc.

References `paths`, and `root`.

7.70.3.8 `bool FileBrowserControl::match (const std::string & file, const std::string & filt)` [static, protected]

returns true if *file* matches *filt*

Definition at line 73 of file FileBrowserControl.cc.

7.70.3.9 `void FileBrowserControl::rebuildmenu ()` [protected]

rescans current directory and builds menus

Definition at line 98 of file FileBrowserControl.cc.

References `ControlBase::clearSlots()`, `ControlBase::ControlBase()`, `filter`, `ControlBase::hilights`, `makePath()`, `match()`, `ControlBase::options`, `paths`, `ControlBase::pushSlot()`, and `recurse`.

7.70.3.10 `virtual ControlBase* FileBrowserControl::selectedFile (const std::string &)` [inline, protected, virtual]

the big money function - by default calls the parent if it exists, otherwise nothing

returning NULL means deactivate, this (default) to stay put, or a different Control if you want a submenu

Reimplemented in [DumpFileControl](#), [LoadPostureControl](#), [LoadWalkControl](#), [PlaySoundControl](#), and [RunSequenceControl< SequenceSize >](#).

Definition at line 41 of file FileBrowserControl.h.

7.70.3.11 void FileBrowserControl::setFilter (const std::string & *filt*)
[inline]

sets [filter](#); remember can only use one wildcard, e.g. *.ext or filename.ext or filename*

Definition at line 34 of file FileBrowserControl.h.

References [filter](#).

7.70.3.12 void FileBrowserControl::setPath (const std::string & *path*)

sets [paths](#)

Definition at line 47 of file FileBrowserControl.cc.

References [paths](#).

7.70.3.13 void FileBrowserControl::setRecurse (bool *r*) [inline]

sets [recurse](#)

Definition at line 26 of file FileBrowserControl.h.

References [recurse](#).

7.70.3.14 void FileBrowserControl::setRoot (const std::string & *path*)

sets [root](#)

Definition at line 39 of file FileBrowserControl.cc.

References [paths](#), and [root](#).

7.70.4 Member Data Documentation**7.70.4.1 std::string [FileBrowserControl::filter](#)** [protected]

default "*", only display matching files; only can use one wildcard, e.g. *.ext or filename.ext or filename*

Definition at line 62 of file FileBrowserControl.h.

7.70.4.2 std::vector<std::string> [FileBrowserControl::paths](#) [protected]

list of directories from root

Definition at line 60 of file FileBrowserControl.h.

7.70.4.3 **bool** [FileBrowserControl::recurse](#) [protected]

if true (default), will show directories; if false, subdirectories are hidden

Definition at line 58 of file FileBrowserControl.h.

7.70.4.4 **std::string** [FileBrowserControl::root](#) [protected]

the path to browse, default ""

Definition at line 59 of file FileBrowserControl.h.

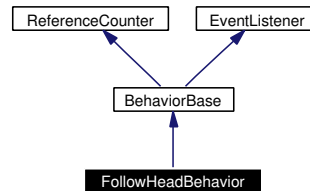
The documentation for this class was generated from the following files:

- [FileBrowserControl.h](#)
- [FileBrowserControl.cc](#)

7.71 FollowHeadBehavior Class Reference

```
#include <FollowHeadBehavior.h>
```

Inheritance diagram for FollowHeadBehavior:



7.71.1 Detailed Description

Will walk where the head is pointing.

Press the chin button to loosen the head to point it, release the button to lock it again

Tilt of head determines x axis (forward/backward)

Roll of head determines y axis (sideways strafing)

Pan of head determines z axis (rotational)

The zero point of joint position is zero motion. Since the tilt is asymmetric (can tilt down farther than it can tilt up), the full range of the down tilt isn't used - if you tilt down farther than you could tilt it back, it'll just clip the speed. Besides, if the head is all the way down, it screws up the walk because the center of balance is changed.

Definition at line 23 of file FollowHeadBehavior.h.

Public Member Functions

- [FollowHeadBehavior](#) ()
just sets up the variables
- virtual [~FollowHeadBehavior](#) ()
calls [DoStop\(\)](#) if [isActive\(\)](#)
- virtual void [DoStart](#) ()
Register for events and creates and adds two motion commands - a walker and a head pointer.
- virtual void [DoStop](#) ()

Removes its two motion commands.

- virtual void [processEvent](#) (const [EventBase](#) &e)

Handles event processing.

- virtual std::string [getName](#) () const

Identifies the behavior in menus and such.

Static Public Member Functions

- std::string [getClassDescription](#) ()

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- const [EventBase](#) [head_release](#)
event mask for releasing head (chin button down)

- const [EventBase](#) [head_lock](#)
event mask for locking head (chin button up)

- const [EventBase](#) [clock](#)
event mask for updating walk direction (every 150 ms)

- [MotionManager::MC_ID](#) [walker_id](#)
MC_ID for walker.

7.71.2 Constructor & Destructor Documentation

7.71.2.1 FollowHeadBehavior::FollowHeadBehavior ()

just sets up the variables

Definition at line 10 of file FollowHeadBehavior.cc.

References [ERS210Info::ChinButOffset](#).

7.71.2.2 FollowHeadBehavior::~FollowHeadBehavior () [virtual]

calls [DoStop\(\)](#) if [isActive\(\)](#)

Definition at line 18 of file FollowHeadBehavior.cc.

References [DoStop\(\)](#), and [BehaviorBase::isActive\(\)](#).

7.71.3 Member Function Documentation

7.71.3.1 void FollowHeadBehavior::DoStart () [virtual]

Register for events and creates and adds two motion commands - a walker and a head pointer.

Reimplemented from [BehaviorBase](#).

Definition at line 23 of file FollowHeadBehavior.cc.

References [EventRouter::addListener\(\)](#), [MotionManager::addMotion\(\)](#), [clock](#), [BehaviorBase::DoStart\(\)](#), [erouter](#), [head_lock](#), [head_release](#), [motman](#), [processEvent\(\)](#), and [walker_id](#).

7.71.3.2 void FollowHeadBehavior::DoStop () [virtual]

Removes its two motion commands.

Reimplemented from [BehaviorBase](#).

Definition at line 35 of file FollowHeadBehavior.cc.

References [BehaviorBase::DoStop\(\)](#), [erouter](#), [EventRouter::forgetListener\(\)](#), [MotionManager::invalid_MC_ID](#), [motman](#), [MotionManager::removeMotion\(\)](#), and [walker_id](#).

7.71.3.3 std::string FollowHeadBehavior::getClassDescription () [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 43 of file FollowHeadBehavior.h.

7.71.3.4 virtual std::string FollowHeadBehavior::getName () const [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 42 of file FollowHeadBehavior.h.

7.71.3.5 void FollowHeadBehavior::processEvent (const [EventBase](#) & e) [virtual]

Handles event processing.

After every clock pulse, sets walk in direction of head

Reimplemented from [BehaviorBase](#).

Definition at line 44 of file FollowHeadBehavior.cc.

References [EventRouter::addListener\(\)](#), [MotionManager::addMotion\(\)](#), [ASSERT](#), [clock](#), [erouter](#), [EventBase::getName\(\)](#), [head_lock](#), [head_release](#), [ERS210Info::HeadOffset](#), [WalkMC::MAX_DA](#), [WalkMC::MAX_DX](#), [WalkMC::MAX_DY](#), [MMAccessor< MC_t >::mc\(\)](#), [motman](#), [ERS210Info::NumHeadJoints](#), [ERS210Info::outputRanges](#), [WorldState::outputs](#), [ERS210Info::PanOffset](#), [EventRouter::removeListener\(\)](#), [ERS210Info::RollOffset](#), [MotionManager::setOutput\(\)](#), [state](#), [ERS210Info::TiltOffset](#), and [walker_id](#).

7.71.4 Member Data Documentation

7.71.4.1 const [EventBase](#) FollowHeadBehavior::clock [protected]

event mask for updating walk direction (every 150 ms)

Definition at line 48 of file FollowHeadBehavior.h.

7.71.4.2 const [EventBase](#) FollowHeadBehavior::head_lock [protected]

event mask for locking head (chin button up)

Definition at line 47 of file FollowHeadBehavior.h.

7.71.4.3 const [EventBase](#) FollowHeadBehavior::head_release [protected]

event mask for releasing head (chin button down)

Definition at line 46 of file FollowHeadBehavior.h.

7.71.4.4 [MotionManager::MC_ID FollowHeadBehavior::walker_id](#) [protected]

MC_ID for walker.

Definition at line 49 of file FollowHeadBehavior.h.

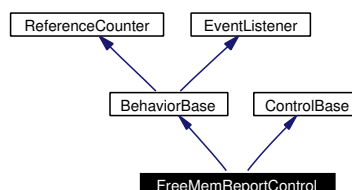
The documentation for this class was generated from the following files:

- [FollowHeadBehavior.h](#)
- [FollowHeadBehavior.cc](#)

7.72 FreeMemReportControl Class Reference

```
#include <FreeMemReportControl.h>
```

Inheritance diagram for FreeMemReportControl:



7.72.1 Detailed Description

gives reports on free memory size at variable rates, can also warn if free memory drops too low.

Definition at line 12 of file FreeMemReportControl.h.

Public Member Functions

- virtual void [DoStart](#) ()
By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.
- virtual void [DoStop](#) ()
By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).
- virtual void [processEvent](#) (const [EventBase](#) &e)
Allows you to get away with not supplying a [processEvent\(\)](#) function for the [EventListener](#) interface. By default, does nothing.
- virtual std::string [getName](#) () const
Identifies the behavior in menus and such.
- virtual void [refresh](#) ()
called when the child has died and this control should refresh its display
- void [report](#) ()

reports size of free memory - if this is below low_mem, also generates a warning

- void [resetTimerFreq](#) ()
resets timer delays

Constructors/Destructors

constructor

- [FreeMemReportControl](#) ()
- [FreeMemReportControl](#) (const std::string &n)
- [FreeMemReportControl](#) (const std::string &n, const std::string &d)
- virtual [~FreeMemReportControl](#) ()
destructor

Static Public Member Functions

- size_t [freeMem](#) ()
returns the size of the free memory

Protected Member Functions

- void [init](#) ()
builds the submenus

Protected Attributes

- unsigned int [report_freq](#)
how often to report memory size (in seconds - -IU turns off, 0 is as often as possible)
- unsigned int [low_mem](#)
threshold to trigger low memory warning (in kilobytes)
- unsigned int [monitor_freq](#)
how often to check for low memory (in milliseconds - -IU turns off, 0 is as often as possible)
- bool [isWarning](#)
true we already know we're below threshold

7.72.2 Constructor & Destructor Documentation

7.72.2.1 `FreeMemReportControl::FreeMemReportControl ()` [inline]

Definition at line 16 of file `FreeMemReportControl.h`.

References `init()`, `isWarning`, `low_mem`, `monitor_freq`, and `report_freq`.

7.72.2.2 `FreeMemReportControl::FreeMemReportControl (const std::string & n)` [inline]

Definition at line 17 of file `FreeMemReportControl.h`.

References `init()`, `isWarning`, `low_mem`, `monitor_freq`, and `report_freq`.

7.72.2.3 `FreeMemReportControl::FreeMemReportControl (const std::string & n, const std::string & d)` [inline]

Definition at line 18 of file `FreeMemReportControl.h`.

References `init()`, `isWarning`, `low_mem`, `monitor_freq`, and `report_freq`.

7.72.2.4 `virtual FreeMemReportControl::~FreeMemReportControl ()` [inline, virtual]

destructor

Definition at line 19 of file `FreeMemReportControl.h`.

References `ControlBase::clearSlots()`, and `ReferenceCounter::SetAutoDelete()`.

7.72.3 Member Function Documentation

7.72.3.1 `virtual void FreeMemReportControl::DoStart ()` [inline, virtual]

By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.

Reimplemented from [BehaviorBase](#).

Definition at line 22 of file `FreeMemReportControl.h`.

References `BehaviorBase::DoStart()`, and `resetTimerFreq()`.

7.72.3.2 virtual void FreeMemReportControl::DoStop () [inline, virtual]

By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).

Reimplemented from [BehaviorBase](#).

Definition at line 27 of file FreeMemReportControl.h.

References [BehaviorBase::DoStop\(\)](#), [erouter](#), [EventRouter::forgetListener\(\)](#), and [isWarning](#).

7.72.3.3 size_t FreeMemReportControl::freeMem () [static]

returns the size of the free memory

Definition at line 37 of file FreeMemReportControl.cc.

References [Socket::printf\(\)](#), and [sout](#).

7.72.3.4 virtual std::string FreeMemReportControl::getName () const [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 35 of file FreeMemReportControl.h.

References [ControlBase::getName\(\)](#), and [BehaviorBase::isActive\(\)](#).

7.72.3.5 void FreeMemReportControl::init () [inline, protected]

builds the submenus

Definition at line 50 of file FreeMemReportControl.h.

References [low_mem](#), [monitor_freq](#), [ControlBase::pushSlot\(\)](#), and [report_freq](#).

7.72.3.6 void FreeMemReportControl::processEvent (const [EventBase](#) & e) [virtual]

Allows you to get away with not supplying a [processEvent\(\)](#) function for the [Event-Listener](#) interface. By default, does nothing.

Reimplemented from [BehaviorBase](#).

Definition at line 5 of file FreeMemReportControl.cc.

References ASSERT, ASSERTRET, freeMem(), EventBase::getGeneratorID(), EventBase::getSourceID(), isWarning, low_mem, Socket::printf(), report(), serr, and EventBase::timerEGID.

7.72.3.7 **virtual void FreeMemReportControl::refresh ()** [inline, virtual]

called when the child has died and this control should refresh its display

Reimplemented from [ControlBase](#).

Definition at line 37 of file FreeMemReportControl.h.

References ControlBase::refresh(), and report().

7.72.3.8 **void FreeMemReportControl::report ()**

reports size of free memory - if this is below low_mem, also generates a warning

Definition at line 27 of file FreeMemReportControl.cc.

References freeMem(), isWarning, low_mem, Socket::printf(), resetTimerFreq(), serr, and sout.

7.72.3.9 **void FreeMemReportControl::resetTimerFreq ()**

resets timer delays

Definition at line 44 of file FreeMemReportControl.cc.

References EventRouter::addTimer(), erouter, monitor_freq, EventRouter::removeTimer(), and report_freq.

7.72.4 **Member Data Documentation**

7.72.4.1 **bool FreeMemReportControl::isWarning** [protected]

true we already know we're below threshold

Definition at line 60 of file FreeMemReportControl.h.

7.72.4.2 **unsigned int FreeMemReportControl::low_mem** [protected]

threshold to trigger low memory warning (in kilobytes)

Definition at line 58 of file FreeMemReportControl.h.

7.72.4.3 unsigned int [FreeMemReportControl::monitor_freq](#) [protected]

how often to check for low memory (in milliseconds - -1U turns off, 0 is as often as possible)

Definition at line 59 of file FreeMemReportControl.h.

7.72.4.4 unsigned int [FreeMemReportControl::report_freq](#) [protected]

how often to report memory size (in seconds - -1U turns off, 0 is as often as possible)

Definition at line 57 of file FreeMemReportControl.h.

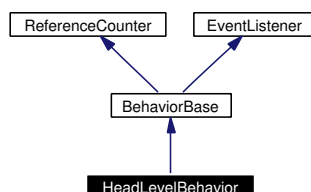
The documentation for this class was generated from the following files:

- [FreeMemReportControl.h](#)
- [FreeMemReportControl.cc](#)

7.73 HeadLevelBehavior Class Reference

```
#include <HeadLevelBehavior.h>
```

Inheritance diagram for HeadLevelBehavior:



7.73.1 Detailed Description

Tests the head leveling code of [HeadPointerMC](#).

Definition at line 13 of file HeadLevelBehavior.h.

Public Member Functions

- [HeadLevelBehavior](#) ()
constructor
- virtual [~HeadLevelBehavior](#) ()
destructor
- virtual void [DoStart](#) ()
By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.
- virtual void [DoStop](#) ()
By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).
- virtual void [processEvent](#) (const [EventBase](#) &event)
Allows you to get away with not supplying a [processEvent\(\)](#) function for the [EventListener](#) interface. By default, does nothing.
- virtual std::string [getName](#) () const
Identifies the behavior in menus and such.

Static Public Member Functions

- `std::string getClassDescription ()`
Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- `const EventBase head_release`
event mask for releasing head (chin button down)
- `const EventBase head_lock`
event mask for locking head (chin button up)
- `const SharedObject< HeadPointerMC > head`
might as well just hang on to the whole memory region and reuse it, we can peek for most of our stuff
- `MotionManager::MC_ID head_id`
MCID of headpointer.

7.73.2 Constructor & Destructor Documentation

7.73.2.1 HeadLevelBehavior::HeadLevelBehavior () [inline]

constructor

Definition at line 16 of file HeadLevelBehavior.h.

References `SharedObjectBase::getRegion()`, `head`, `head_id`, `head_lock`, `head_release`, and `ERS210Info::HeadFrButOffset`.

7.73.2.2 virtual HeadLevelBehavior::~~HeadLevelBehavior () [inline, virtual]

destructor

Definition at line 26 of file HeadLevelBehavior.h.

References `SharedObjectBase::getRegion()`, and `head`.

7.73.3 Member Function Documentation

7.73.3.1 `virtual void HeadLevelBehavior::DoStart () [inline, virtual]`

By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.

Reimplemented from [BehaviorBase](#).

Definition at line 30 of file `HeadLevelBehavior.h`.

References `EventRouter::addListener()`, `MotionManager::addMotion()`, `BehaviorBase::DoStart()`, `erouter`, `HeadPointerMC::GravityRelative`, `head`, `head_id`, `head_lock`, `head_release`, `ERS210Info::HeadOffset`, `motman`, `WorldState::outputs`, `ERS210Info::PanOffset`, `ERS210Info::RollOffset`, `state`, and `ERS210Info::TiltOffset`.

7.73.3.2 `virtual void HeadLevelBehavior::DoStop () [inline, virtual]`

By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).

Reimplemented from [BehaviorBase](#).

Definition at line 39 of file `HeadLevelBehavior.h`.

References `HeadPointerMC::BodyRelative`, `BehaviorBase::DoStop()`, `erouter`, `EventRouter::forgetListener()`, `head`, `head_id`, `motman`, and `MotionManager::removeMotion()`.

7.73.3.3 `std::string HeadLevelBehavior::getClassDescription () [inline, static]`

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 57 of file `HeadLevelBehavior.h`.

7.73.3.4 `virtual std::string HeadLevelBehavior::getName () const [inline, virtual]`

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 56 of file `HeadLevelBehavior.h`.

7.73.3.5 virtual void HeadLevelBehavior::processEvent (const [EventBase](#) & *event*) [inline, virtual]

Allows you to get away with not supplying a [processEvent\(\)](#) function for the [EventListener](#) interface. By default, does nothing.

Reimplemented from [BehaviorBase](#).

Definition at line 46 of file [HeadLevelBehavior.h](#).

References [MotionManager::addMotion\(\)](#), [ASSERTRET](#), [HeadPointerMC::Body-Relative](#), [EventBase::getName\(\)](#), [head](#), [head_lock](#), [head_release](#), [ERS210Info::Head-Offset](#), [motman](#), [ERS210Info::NumHeadJoints](#), [WorldState::outputs](#), [state](#), and [ERS210Info::TPROffset_t](#).

7.73.4 Member Data Documentation

7.73.4.1 const [SharedObject](#)<[HeadPointerMC](#)> [HeadLevelBehavior::head](#) [protected]

might as well just hang on to the whole memory region and reuse it, we can peek for most of our stuff

Definition at line 62 of file [HeadLevelBehavior.h](#).

7.73.4.2 [MotionManager::MC_ID](#) [HeadLevelBehavior::head_id](#) [protected]

MCID of headpointer.

Definition at line 63 of file [HeadLevelBehavior.h](#).

7.73.4.3 const [EventBase](#) [HeadLevelBehavior::head_lock](#) [protected]

event mask for locking head (chin button up)

Definition at line 61 of file [HeadLevelBehavior.h](#).

7.73.4.4 const [EventBase](#) [HeadLevelBehavior::head_release](#) [protected]

event mask for releasing head (chin button down)

Definition at line 60 of file [HeadLevelBehavior.h](#).

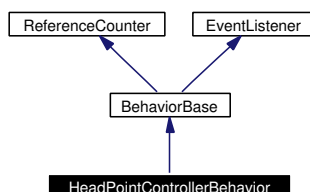
The documentation for this class was generated from the following file:

- [HeadLevelBehavior.h](#)

7.74 HeadPointControllerBehavior Class Reference

```
#include <HeadPointControllerBehavior.h>
```

Inheritance diagram for HeadPointControllerBehavior:



7.74.1 Detailed Description

Listens to control commands coming in from the command port for remotely controlling the head.

Definition at line 15 of file HeadPointControllerBehavior.h.

Public Member Functions

- [HeadPointControllerBehavior](#) ()
constructor
- virtual [~HeadPointControllerBehavior](#) ()
destructor
- virtual void [DoStart](#) ()
By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.
- virtual void [DoStop](#) ()
By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).
- virtual void [processEvent](#) (const [EventBase](#) &)
The only event we could possibly receive is the stop-if-no-heartbeat timer.
- virtual std::string [getName](#) () const
Identifies the behavior in menus and such.

Static Public Member Functions

- int [mechacmd_callback](#) (char *buf, int bytes)
called by wireless when there's new data
- std::string [getClassDescription](#) ()
Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Static Public Attributes

- [HeadPointControllerBehavior](#) * [theOne](#) = NULL

Protected Attributes

- [MotionManager::MC_ID](#) [head_id](#)
the [HeadPointerMC](#) to use

Private Member Functions

- void [runCommand](#) (unsigned char *command)
Executes a command. Called by [mechacmd_callback](#).
- [HeadPointControllerBehavior](#) (const [HeadPointControllerBehavior](#) &)
don't call
- [HeadPointControllerBehavior](#) operator= (const [HeadPointControllerBehavior](#) &)
don't call

Private Attributes

- float [t](#)
head parameter
- float [p](#)
head parameter

- float `r`
head parameter
- `HeadPointControllerBehavior * theLastOne`
- `Socket * cmdsock`
The input command stream socket.

Static Private Attributes

Command Bytes

- const char `CMD_tilt = 't'`
handy symbol for matching incoming communication
- const char `CMD_pan = 'p'`
handy symbol for matching incoming communication
- const char `CMD_roll = 'r'`
handy symbol for matching incoming communication

7.74.2 Constructor & Destructor Documentation

7.74.2.1 `HeadPointControllerBehavior::HeadPointControllerBehavior (const HeadPointControllerBehavior &) [private]`

don't call

7.74.2.2 `HeadPointControllerBehavior::HeadPointControllerBehavior () [inline]`

constructor

Definition at line 56 of file `HeadPointControllerBehavior.h`.

References `cmdsock`, `head_id`, `p`, `r`, `SocketNS::SOCK_STREAM`, `t`, `theLastOne`, `theOne`, and `wireless`.

7.74.2.3 `virtual HeadPointControllerBehavior::~~HeadPointControllerBehavior () [inline, virtual]`

destructor

Definition at line 64 of file HeadPointControllerBehavior.h.

References `theLastOne`, and `theOne`.

7.74.3 Member Function Documentation

7.74.3.1 void HeadPointControllerBehavior::DoStart () [virtual]

By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.

Reimplemented from [BehaviorBase](#).

Definition at line 52 of file HeadPointControllerBehavior.cc.

References `EventRouter::addListener()`, `MotionManager::addMotion()`, `cmdsock`, `config`, `BehaviorBase::DoStart()`, `erouter`, `head_id`, `Config::main_config::headControl_port`, `Wireless::listen()`, `Controller::loadGUI()`, `Config::main`, `mechacmd_callback()`, `motman`, `Wireless::setReceiver()`, `Socket::sock`, `EventBase::timerEGID`, and `wireless`.

7.74.3.2 void HeadPointControllerBehavior::DoStop () [virtual]

By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).

Reimplemented from [BehaviorBase](#).

Definition at line 66 of file HeadPointControllerBehavior.cc.

References `Wireless::close()`, `Controller::closeGUI()`, `cmdsock`, `BehaviorBase::DoStop()`, `erouter`, `EventRouter::forgetListener()`, `head_id`, `motman`, `MotionManager::removeMotion()`, and `wireless`.

7.74.3.3 std::string HeadPointControllerBehavior::getClassDescription () [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 74 of file HeadPointControllerBehavior.h.

References `config`, `Config::main_config::headControl_port`, and `Config::main`.

7.74.3.4 `virtual std::string HeadPointControllerBehavior::getName () const`
`[inline, virtual]`

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 73 of file HeadPointControllerBehavior.h.

7.74.3.5 `int HeadPointControllerBehavior::mechacmd_callback (char * buf, int`
`bytes) [static]`

called by wireless when there's new data

Definition at line 80 of file HeadPointControllerBehavior.cc.

References `runCommand()`, and `theOne`.

7.74.3.6 [HeadPointControllerBehavior](#) `HeadPointController-`
`Behavior::operator= (const HeadPointControllerBehavior &)`
`[private]`

don't call

7.74.3.7 `virtual void HeadPointControllerBehavior::processEvent (const`
`EventBase &) [inline, virtual]`

The only event we could possibly receive is the stop-if-no-heartbeat timer.

Reimplemented from [BehaviorBase](#).

Definition at line 71 of file HeadPointControllerBehavior.h.

7.74.3.8 `void HeadPointControllerBehavior::runCommand (unsigned char *`
`command) [private]`

Executes a command. Called by `mechacmd_callback`.

Definition at line 7 of file HeadPointControllerBehavior.cc.

References `EventRouter::addTimer()`, `CMD_pan`, `CMD_roll`, `CMD_tilt`, `erouter`, `head_-id`, `ERS210Info::HeadOffset`, `ERS210Info::MaxRange`, `ERS210Info::outputRanges`, `p`, `ERS210Info::PanOffset`, `r`, `EventRouter::removeTimer()`, `ERS210Info::RollOffset`, `t`, and `ERS210Info::TiltOffset`.

7.74.4 Member Data Documentation

7.74.4.1 `const char HeadPointControllerBehavior::CMD_pan = 'p'`
[static, private]

handy symbol for matching incoming communication

Definition at line 30 of file HeadPointControllerBehavior.h.

7.74.4.2 `const char HeadPointControllerBehavior::CMD_roll = 'r'` [static, private]

handy symbol for matching incoming communication

Definition at line 31 of file HeadPointControllerBehavior.h.

7.74.4.3 `const char HeadPointControllerBehavior::CMD_tilt = 't'` [static, private]

handy symbol for matching incoming communication

Definition at line 29 of file HeadPointControllerBehavior.h.

7.74.4.4 `Socket* HeadPointControllerBehavior::cmdsock` [private]

The input command stream socket.

Definition at line 46 of file HeadPointControllerBehavior.h.

7.74.4.5 `MotionManager::MC_ID HeadPointControllerBehavior::head_id`
[protected]

the [HeadPointerMC](#) to use

Definition at line 25 of file HeadPointControllerBehavior.h.

7.74.4.6 `float HeadPointControllerBehavior::p` [private]

head parameter

Definition at line 35 of file HeadPointControllerBehavior.h.

7.74.4.7 `float HeadPointControllerBehavior::r` [private]

head parameter

Definition at line 36 of file HeadPointControllerBehavior.h.

7.74.4.8 `float HeadPointControllerBehavior::t` `[private]`

head parameter

Definition at line 34 of file HeadPointControllerBehavior.h.

7.74.4.9 `HeadPointControllerBehavior* HeadPointControllerBehavior::the-
LastOne` `[private]`

The last HPCB object that was theOne, so we can restore it to prominence when we die. This is a nice gesture, but it doesn't really make sense since we're all using the same port. But just in case something changes and we don't do that, this mechanism is in place.

Definition at line 43 of file HeadPointControllerBehavior.h.

7.74.4.10 `HeadPointControllerBehavior * HeadPointControllerBehavior::the-
One = NULL` `[static]`

Points to the one HeadPointControllerBehavior object that the input command stream is talking to. A kludge. Dunno how you're gonna make sure you're not using this uninitialized.

Definition at line 5 of file HeadPointControllerBehavior.cc.

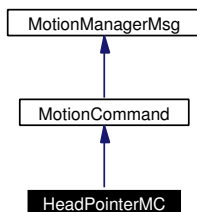
The documentation for this class was generated from the following files:

- [HeadPointControllerBehavior.h](#)
- [HeadPointControllerBehavior.cc](#)

7.75 HeadPointerMC Class Reference

```
#include <HeadPointerMC.h>
```

Inheritance diagram for HeadPointerMC:



7.75.1 Detailed Description

This class gives some quick and easy functions to point the head at things.

Definition at line 10 of file HeadPointerMC.h.

Public Types

- enum [CoordFrame_t](#) { [BodyRelative](#), [GravityRelative](#) }

Various modes the head can be in. In the future may want to add ability to explicitly track an object or point in the world model.

Public Member Functions

- [HeadPointerMC](#) ()
constructor, defaults to active, BodyRelative, all joints at 0
- virtual [~HeadPointerMC](#) ()
destructor
- void [setWeight](#) (double w)
sets the weight values for all the neck joints
- void [setWeight](#) (RobotInfo::TPROffset_t i, double weight)
set a specific head joint weight, pass one of RobotInfo::TPROffset_t, not a full offset!

- void [setActive](#) (bool a)
sets active flag, see [isDirty\(\)](#)
- bool [getActive](#) () const
returns active flag, see [isDirty\(\)](#)
- double [convert](#) (RobotInfo::TPROffset_t i, double v, [CoordFrame_t](#) srcmode, [CoordFrame_t](#) tgtmode) const
converts a value v in srcmode to a value in tgtmode that would leave the joint angle for joint i constant (you probably won't need to call this directly)

Joint Accessors

Note that none of these are virtual, so you don't have to checkout to use them, you should be able to use [MotionManager::peekMotion\(\)](#)

- void [setJoints](#) (double tilt, double pan, double roll)
Directly sets the neck values, uses current mode.
- void [setMode](#) ([CoordFrame_t](#) m, bool convert=true)
sets all the joints to the given mode, will convert the values to the new mode if convert is true
- void [setJointMode](#) (RobotInfo::TPROffset_t i, [CoordFrame_t](#) m, bool convert=true)
set a specific head joint's mode, will convert from previous mode's value to next mode's value if convert is true. Pass one of RobotInfo::TPROffset_t, not a full offset!
- void [setJointValue](#) (RobotInfo::TPROffset_t i, double value)
set a specific head joint's value (for whatever mode it's in), pass one of RobotInfo::TPROffset_t, not a full offset!
- void [setJointValueAndMode](#) (RobotInfo::TPROffset_t i, double value, [CoordFrame_t](#) m)
set a specific head joint, pass one of RobotInfo::TPROffset_t, not a full offset!
- void [setJointValueFromMode](#) (RobotInfo::TPROffset_t i, double value, [CoordFrame_t](#) m)
set a specific head joint auto converting value from mode m to the current mode, pass one of RobotInfo::TPROffset_t, not a full offset!
- [CoordFrame_t](#) [getJointMode](#) (RobotInfo::TPROffset_t i) const
returns the current mode for joint i (use RobotInfo::TPROffset_t, not global offset)
- double [getJointValue](#) (RobotInfo::TPROffset_t i) const

returns the current value (relative to the current mode) of joint *i*, use [getOutputCmd\(\)](#) if you want to know the actual **target** joint value (to get the actual current joint position, look in [WorldState](#))

Inherited:

- virtual int [updateOutputs](#) ()
Updates where the head is looking.
- virtual const [OutputCmd](#) & [getOutputCmd](#) (unsigned int *i*)
returns one of the [headJoints](#) entries or ::unusedJoint if not a head joint
- virtual int [isDirty](#) ()
true if a change has been made since the last [updateJointCmds\(\)](#) and we're active
- virtual int [isAlive](#) ()
Updates where the head is looking.

Protected Member Functions

- double [convToBodyRelative](#) (RobotInfo::TPROffset_t *i*, double *v*, [CoordFrame_t](#) *mode*) const
*converts to a body relative measurement for joint *i**
- double [convFromBodyRelative](#) (RobotInfo::TPROffset_t *i*, double *v*, [CoordFrame_t](#) *mode*) const
*converts from a body relative measurement for joint *i**

Protected Attributes

- bool [dirty](#)
true if a change has been made since last call to [updateJointCmds\(\)](#)
- bool [active](#)
set by accessor functions, defaults to true
- [OutputCmd](#) [headJoints](#) [NumHeadJoints]
stores the current target head angles
- double [headValues](#) [NumHeadJoints]

stores the target value of each joint, relative to [headModes](#)

- [CoordFrame_t headModes](#) [NumHeadJoints]

stores the current mode of each joint, for instance so tilt can be GravityRelative while pan is static

Static Protected Attributes

- const [OutputCmd unused](#)

handy when we need a reference to an unused joint

7.75.2 Member Enumeration Documentation

7.75.2.1 enum [HeadPointerMC::CoordFrame_t](#)

Various modes the head can be in. In the future may want to add ability to explicitly track an object or point in the world model.

Enumeration values:

BodyRelative holds neck at a specified position, like a [PostureEngine](#), but neck specific

GravityRelative uses accelerometers to keep a level head, doesn't apply for pan joint, but in future could use localization for pan

Definition at line 18 of file HeadPointerMC.h.

7.75.3 Constructor & Destructor Documentation

7.75.3.1 [HeadPointerMC::HeadPointerMC \(\)](#)

constructor, defaults to active, BodyRelative, all joints at 0

Definition at line 9 of file HeadPointerMC.cc.

References [BodyRelative](#), [headModes](#), [headValues](#), [ERS210Info::NumHeadJoints](#), and [setWeight\(\)](#).

7.75.3.2 [virtual HeadPointerMC::~~HeadPointerMC \(\)](#) [inline, virtual]

destructor

Definition at line 15 of file HeadPointerMC.h.

7.75.4 Member Function Documentation

7.75.4.1 `double HeadPointerMC::convert (RobotInfo::TPROffset_t i, double v, CoordFrame_t srcmode, CoordFrame_t tgtmode) const` [inline]

converts a value *v* in *srcmode* to a value in *tgtmode* that would leave the joint angle for joint *i* constant (you probably won't need to call this directly)

Definition at line 29 of file HeadPointerMC.h.

References convFromBodyRelative(), and convToBodyRelative().

7.75.4.2 `double HeadPointerMC::convFromBodyRelative (RobotInfo::TPROffset_t i, double v, CoordFrame_t mode) const` [protected]

converts from a body relative measurement for joint *i*

Todo

this is perhaps a bit amateurish - could be more accurate

Definition at line 95 of file HeadPointerMC.cc.

References ASSERT, ERS210Info::BAccelOffset, BodyRelative, ERS210Info::DAccelOffset, GravityRelative, ERS210Info::LAccelOffset, ERS210Info::PanOffset, ERS210Info::RollOffset, WorldState::sensors, state, and ERS210Info::TiltOffset.

7.75.4.3 `double HeadPointerMC::convToBodyRelative (RobotInfo::TPROffset_t i, double v, CoordFrame_t mode) const` [protected]

converts to a body relative measurement for joint *i*

Todo

this is perhaps a bit amateurish - could be more accurate

Definition at line 69 of file HeadPointerMC.cc.

References ASSERT, ERS210Info::BAccelOffset, BodyRelative, ERS210Info::DAccelOffset, GravityRelative, ERS210Info::LAccelOffset, ERS210Info::PanOffset, ERS210Info::RollOffset, WorldState::sensors, state, and ERS210Info::TiltOffset.

7.75.4.4 **bool HeadPointerMC::getActive () const** [inline]

returns [active](#) flag, see [isDirty\(\)](#)

Definition at line 26 of file HeadPointerMC.h.

References [active](#).

7.75.4.5 **CoordFrame_t HeadPointerMC::getJointMode (RobotInfo::TPROffset_t i) const** [inline]

returns the current mode for joint *i* (use RobotInfo::TPROffset_t, not global offset)

Definition at line 41 of file HeadPointerMC.h.

References [CoordFrame_t](#), and [headModes](#).

7.75.4.6 **double HeadPointerMC::getJointValue (RobotInfo::TPROffset_t i) const** [inline]

returns the current value (relative to the current mode) of joint *i*, use [getOutputCmd\(\)](#) if you want to know the actual **target** joint value (to get the actual current joint position, look in [WorldState](#))

Definition at line 42 of file HeadPointerMC.h.

References [headValues](#).

7.75.4.7 **const OutputCmd & HeadPointerMC::getOutputCmd (unsigned int i)** [virtual]

returns one of the [headJoints](#) entries or ::unusedJoint if not a head joint

Definition at line 60 of file HeadPointerMC.cc.

References [getActive\(\)](#), [headJoints](#), [ERS210Info::HeadOffset](#), [ERS210Info::NumHeadJoints](#), and [unused](#).

7.75.4.8 **virtual int HeadPointerMC::isAlive ()** [inline, virtual]

Updates where the head is looking.

Implements [MotionCommand](#).

Definition at line 49 of file HeadPointerMC.h.

7.75.4.9 virtual int HeadPointerMC::isDirty () [inline, virtual]

true if a change has been made since the last updateJointCmds() and we're active

Implements [MotionCommand](#).

Definition at line 48 of file HeadPointerMC.h.

References active, and dirty.

7.75.4.10 void HeadPointerMC::setActive (bool *a*) [inline]

sets [active](#) flag, see [isDirty\(\)](#)

Definition at line 25 of file HeadPointerMC.h.

References active.

**7.75.4.11 void HeadPointerMC::setJointMode (RobotInfo::TPROffset_t *i*,
[CoordFrame_t](#) *m*, bool *convert* = true)**

set a specific head joint's mode, will convert from previous mode's value to next mode's value if convert is true. Pass one of RobotInfo::TPROffset_t, not a full offset!

Definition at line 36 of file HeadPointerMC.cc.

References convert(), dirty, headModes, and headValues.

7.75.4.12 void HeadPointerMC::setJoints (double *tilt*, double *pan*, double *roll*)

Directly sets the neck values, uses current mode.

Definition at line 18 of file HeadPointerMC.cc.

References dirty, headValues, ERS210Info::PanOffset, ERS210Info::RollOffset, and ERS210Info::TiltOffset.

**7.75.4.13 void HeadPointerMC::setJointValue (RobotInfo::TPROffset_t *i*,
double *value*) [inline]**

set a specific head joint's value (for whatever mode it's in), pass one of RobotInfo::TPROffset_t, not a full offset!

Definition at line 38 of file HeadPointerMC.h.

References dirty, and headValues.

7.75.4.14 void HeadPointerMC::setJointValueAndMode
(RobotInfo::TPROffset_t *i*, double *value*, CoordFrame_t *m*)
 [inline]

set a specific head joint, pass one of RobotInfo::TPROffset_t, not a full offset!

Definition at line 39 of file HeadPointerMC.h.

References dirty, headModes, and headValues.

7.75.4.15 void HeadPointerMC::setJointValueFromMode
(RobotInfo::TPROffset_t *i*, double *value*, CoordFrame_t *m*)
 [inline]

set a specific head joint auto converting *value* from mode *m* to the current mode, pass one of RobotInfo::TPROffset_t, not a full offset!

Definition at line 40 of file HeadPointerMC.h.

References convert(), dirty, headModes, and headValues.

7.75.4.16 void HeadPointerMC::setMode (CoordFrame_t *m*, bool *convert* = true)

sets all the joints to the given mode, will convert the values to the new mode if *convert* is true

Definition at line 31 of file HeadPointerMC.cc.

References ERS210Info::NumHeadJoints, and setJointMode().

7.75.4.17 void HeadPointerMC::setWeight (RobotInfo::TPROffset_t *i*, double *weight*) [inline]

set a specific head joint weight, pass one of RobotInfo::TPROffset_t, not a full offset!

Definition at line 24 of file HeadPointerMC.h.

References dirty, headJoints, and OutputCmd::weight.

7.75.4.18 void HeadPointerMC::setWeight (double *w*)

sets the weight values for all the neck joints

Definition at line 25 of file HeadPointerMC.cc.

References dirty, headJoints, ERS210Info::NumHeadJoints, and OutputCmd::weight.

7.75.4.19 `int HeadPointerMC::updateOutputs()` [virtual]

Updates where the head is looking.

Implements [MotionCommand](#).

Definition at line 43 of file HeadPointerMC.cc.

References [BodyRelative](#), [convToBodyRelative\(\)](#), [dirty](#), [headJoints](#), [headModes](#), [ERS210Info::HeadOffset](#), [headValues](#), [isDirty\(\)](#), [motman](#), [ERS210Info::NumFrames](#), [ERS210Info::NumHeadJoints](#), [MotionManager::setOutput\(\)](#), and [OutputCmd::value](#).

7.75.5 Member Data Documentation**7.75.5.1** `bool HeadPointerMC::active` [protected]

set by accessor functions, defaults to true

Definition at line 57 of file HeadPointerMC.h.

7.75.5.2 `bool HeadPointerMC::dirty` [protected]

true if a change has been made since last call to [updateJointCmds\(\)](#)

Definition at line 56 of file HeadPointerMC.h.

7.75.5.3 `OutputCmd HeadPointerMC::headJoints[NumHeadJoints]`
[protected]

stores the current target head angles

Definition at line 58 of file HeadPointerMC.h.

7.75.5.4 `CoordFrame.t HeadPointerMC::headModes[NumHeadJoints]`
[protected]

stores the current mode of each joint, for instance so tilt can be GravityRelative while pan is static

Definition at line 60 of file HeadPointerMC.h.

7.75.5.5 `double HeadPointerMC::headValues[NumHeadJoints]`
[protected]

stores the target value of each joint, relative to [headModes](#)

Definition at line 59 of file HeadPointerMC.h.

7.75.5.6 `const OutputCmd HeadPointerMC::unused` `[static,`
`protected]`

handy when we need a reference to an unused joint

Definition at line 7 of file HeadPointerMC.cc.

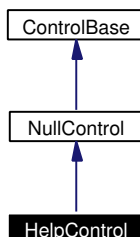
The documentation for this class was generated from the following files:

- [HeadPointerMC.h](#)
- [HeadPointerMC.cc](#)

7.76 HelpControl Class Reference

```
#include <HelpControl.h>
```

Inheritance diagram for HelpControl:



7.76.1 Detailed Description

Recurse through the menu system and outputs the name and description of each item.

Definition at line 8 of file HelpControl.h.

Public Member Functions

- [HelpControl](#) ([ControlBase](#) *r)
< constructor
- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#) disp_id, [Socket](#) *gui)
displays global [Controller](#) commands (hardcoded as strings, will need updates) as well as recursing the menu system (dynamic)
- void [report](#) ([ControlBase](#) *r, const std::string &prefix)
displays the menu items of r and their descriptions, recursing on submenus

Protected Attributes

- [ControlBase](#) * [root](#)
stores root node to begin recursion (this item is not displayed)

Static Protected Attributes

- const unsigned int `term_width` = 80
number of character to word wrap the display

Private Member Functions

- `HelpControl` (const `HelpControl` &)
don't call
- `HelpControl operator=` (const `HelpControl` &)
don't call

7.76.2 Constructor & Destructor Documentation

7.76.2.1 `HelpControl::HelpControl (ControlBase * r)` [inline]

< constructor

Definition at line 11 of file `HelpControl.h`.

References `ControlBase::description`, `ControlBase::name`, and `root`.

7.76.2.2 `HelpControl::HelpControl (const HelpControl &)` [private]

don't call

7.76.3 Member Function Documentation

7.76.3.1 `ControlBase * HelpControl::activate (MotionManager::MC_ID disp_id, Socket * gui)` [virtual]

displays global `Controller` commands (hardcoded as strings, will need updates) as well as recursing the menu system (dynamic)

Reimplemented from `NullControl`.

Definition at line 3 of file `HelpControl.cc`.

References `NullControl::activate()`, `config`, `Config::main`, `Socket::printf()`, `report()`, `root`, `sout`, and `Config::main_config::use_VT100`.

7.76.3.2 [HelpControl](#) [HelpControl::operator=](#) (const [HelpControl](#) &) [private]

don't call

7.76.3.3 void [HelpControl::report](#) ([ControlBase](#) * *r*, const std::string & *prefix*)

displays the menu items of *r* and their descriptions, recursing on submenus

prefix is what should be displayed before each menu item (like a bullet point) this is itself prefixed by 2 spaces for each level of recursion. Word wrapping is performed to maintain the clean indenting

Definition at line 28 of file [HelpControl.cc](#).

References [config](#), [ControlBase::getSlots\(\)](#), [Config::main](#), [Socket::printf\(\)](#), [sout](#), [term_width](#), and [Config::main_config::use_VT100](#).

7.76.4 Member Data Documentation

7.76.4.1 [ControlBase](#)* [HelpControl::root](#) [protected]

stores root node to begin recursion (this item is not displayed)

Definition at line 25 of file [HelpControl.h](#).

7.76.4.2 const unsigned int [HelpControl::term_width](#) = 80 [static, protected]

number of character to word wrap the display

Definition at line 23 of file [HelpControl.h](#).

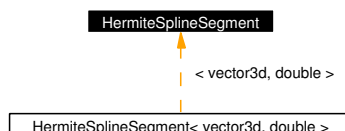
The documentation for this class was generated from the following files:

- [HelpControl.h](#)
- [HelpControl.cc](#)

7.77 HermiteSplineSegment Class Reference

```
#include <Spline.h>
```

Inheritance diagram for HermiteSplineSegment:



Public Member Functions

- [HermiteSplineSegment](#) ()
- void [create](#) (point x1, point x2, point dx1, point dx2)
- void [create](#) (point *pts, int num, int i)
- point [eval](#) (fnum u)
- point [eval_deriv](#) (fnum u)

Public Attributes

- point [a](#)
- point [b](#)
- point [c](#)
- point [d](#)

7.77.1 Constructor & Destructor Documentation

7.77.1.1 HermiteSplineSegment::HermiteSplineSegment () [inline]

Definition at line 36 of file Spline.h.

7.77.2 Member Function Documentation

7.77.2.1 void HermiteSplineSegment::create (point * *pts*, int *num*, int *i*)

7.77.2.2 void HermiteSplineSegment::create (point *x1*, point *x2*, point *dx1*, point *dx2*)

7.77.2.3 point HermiteSplineSegment::eval (fnum *u*)

7.77.2.4 point HermiteSplineSegment::eval_deriv (fnum *u*)

7.77.3 Member Data Documentation

7.77.3.1 point [HermiteSplineSegment::a](#)

Definition at line 37 of file Spline.h.

7.77.3.2 point [HermiteSplineSegment::b](#)

Definition at line 37 of file Spline.h.

7.77.3.3 point [HermiteSplineSegment::c](#)

Definition at line 37 of file Spline.h.

7.77.3.4 point [HermiteSplineSegment::d](#)

Definition at line 37 of file Spline.h.

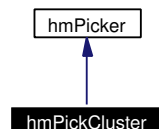
The documentation for this class was generated from the following file:

- [Spline.h](#)

7.78 hmPickCluster Struct Reference

```
#include <almStructures.h>
```

Inheritance diagram for hmPickCluster:



7.78.1 Detailed Description

Picks cluster membership out of the HHM or GHM. For completeness--not use!

Definition at line 88 of file almStructures.h.

Public Member Functions

- virtual float [operator\(\)](#) ([hm_cell](#) &c) const
functor - TSS_TODO

7.78.2 Member Function Documentation

7.78.2.1 virtual float [hmPickCluster::operator\(\)](#) ([hm_cell](#) &c) const
[inline, virtual]

functor - TSS_TODO

Implements [hmPicker](#).

Definition at line 90 of file almStructures.h.

References [_hm_cell::cluster](#).

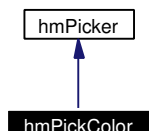
The documentation for this struct was generated from the following file:

- [almStructures.h](#)

7.79 hmPickColor Struct Reference

```
#include <almStructures.h>
```

Inheritance diagram for hmPickColor:



7.79.1 Detailed Description

Picks color measurements out of the HHM or GHM.

Definition at line 83 of file almStructures.h.

Public Member Functions

- virtual float [operator\(\)](#) ([hm_cell](#) &c) const
functor - TSS_TODO

7.79.2 Member Function Documentation

7.79.2.1 virtual float hmPickColor::operator() ([hm_cell](#) & c) const [inline, virtual]

functor - TSS_TODO

Implements [hmPicker](#).

Definition at line 85 of file almStructures.h.

References [_hm_cell::color](#).

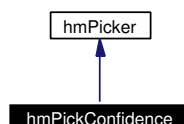
The documentation for this struct was generated from the following file:

- [almStructures.h](#)

7.80 hmPickConfidence Struct Reference

```
#include <almStructures.h>
```

Inheritance diagram for hmPickConfidence:



7.80.1 Detailed Description

Picks confidence values out of the HHM or GHM.

Definition at line 78 of file almStructures.h.

Public Member Functions

- virtual float [operator\(\)](#) ([hm_cell](#) &c) const
functor - TSS.TODO

7.80.2 Member Function Documentation

7.80.2.1 virtual float hmPickConfidence::operator() ([hm_cell](#) & c) const
[inline, virtual]

functor - TSS.TODO

Implements [hmPicker](#).

Definition at line 80 of file almStructures.h.

References [_hm_cell::confidence](#).

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

- [almStructures.h](#)

7.81 hmPicker Struct Reference

```
#include <almStructures.h>
```

Inheritance diagram for hmPicker:



7.81.1 Detailed Description

Abstract base class for height map data pickers. Implementations available.

Definition at line 57 of file almStructures.h.

Public Member Functions

- virtual float [operator\(\)](#) ([hm_cell](#) &c) const=0
functor - TSS_TODO

7.81.2 Member Function Documentation

7.81.2.1 virtual float [hmPicker::operator\(\)](#) ([hm_cell](#) &c) const [pure virtual]

functor - TSS_TODO

Implemented in [hmPickHeight](#), [hmPickTrav](#), [hmPickConfidence](#), [hmPickColor](#), and [hmPickCluster](#).

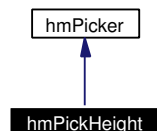
The documentation for this struct was generated from the following file:

- [almStructures.h](#)

7.82 hmPickHeight Struct Reference

```
#include <almStructures.h>
```

Inheritance diagram for hmPickHeight:



7.82.1 Detailed Description

Picks height measurements out of the HHM or GHM.

Definition at line 68 of file almStructures.h.

Public Member Functions

- virtual float [operator\(\)](#) ([hm.cell](#) &c) const
functor - TSS_TODO

7.82.2 Member Function Documentation

7.82.2.1 virtual float [hmPickHeight::operator\(\)](#) ([hm.cell](#) & c) const
[inline, virtual]

functor - TSS_TODO

Implements [hmPicker](#).

Definition at line 70 of file almStructures.h.

References [_hm.cell::height](#).

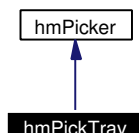
The documentation for this struct was generated from the following file:

- [almStructures.h](#)

7.83 hmPickTrav Struct Reference

```
#include <almStructures.h>
```

Inheritance diagram for hmPickTrav:



7.83.1 Detailed Description

Picks traversability values out of the HHM or GHM.

Definition at line 73 of file almStructures.h.

Public Member Functions

- virtual float [operator\(\)](#) ([hm_cell](#) &c) const
functor - TSS_TODO

7.83.2 Member Function Documentation

7.83.2.1 virtual float [hmPickTrav::operator\(\)](#) ([hm_cell](#) & c) const [inline, virtual]

functor - TSS_TODO

Implements [hmPicker](#).

Definition at line 75 of file almStructures.h.

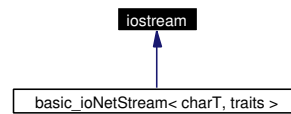
References [_hm_cell::trav](#).

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

- [almStructures.h](#)

7.84 iostream Class Reference

Inheritance diagram for iostream:

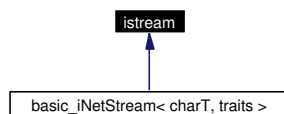


The documentation for this class was generated from the following file:

- [ionetstream.h](#)

7.85 istream Class Reference

Inheritance diagram for istream:



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

- [ionetstream.h](#)

7.86 `karmedbanditExp3` Class Reference

```
#include <karmedbandit.h>
```

7.86.1 Detailed Description

Makes decisions regarding an adversarial k-armed bandit.

Uses algorithms described in: The non-stochastic multi-armed bandit problem Auer, Cesa-Bianchi, Freund, and Schapire October 14, 2002

Definition at line 15 of file `karmedbandit.h`.

Public Member Functions

- `karmedbanditExp3` (unsigned int k, double gammap)
constructor, pass the number of arms
- unsigned int `decide` ()
returns the next choice, [0:k-1]
- void `reward` (bool r)
call this if you want to reward (r==true) or penalize (r==false) the previous decision
- void `reset` ()
resets weights
- double `getGamma` ()
gets gamma parameter
- void `setGamma` (double gammap)
sets gamma parameter
- unsigned int `getK` ()
gets k parameter

Protected Attributes

- `std::vector< double > w`
the weights

- double `lastp`
prob of last choice
- unsigned int `last`
the last choice
- double `g`
gamma

7.86.2 Constructor & Destructor Documentation

7.86.2.1 `karmedbanditExp3::karmedbanditExp3 (unsigned int k, double gammap)` [inline]

constructor, pass the number of arms

Definition at line 18 of file `karmedbandit.h`.

References `g`, `last`, `lastp`, and `w`.

7.86.3 Member Function Documentation

7.86.3.1 `unsigned int karmedbanditExp3::decide ()` [inline]

returns the next choice, `[0:k-1]`

Definition at line 23 of file `karmedbandit.h`.

References `g`, `last`, `lastp`, and `w`.

7.86.3.2 `double karmedbanditExp3::getGamma ()` [inline]

gets gamma parameter

Definition at line 65 of file `karmedbandit.h`.

References `g`.

7.86.3.3 `unsigned int karmedbanditExp3::getK ()` [inline]

gets k parameter

Definition at line 69 of file `karmedbandit.h`.

References `w`.

7.86.3.4 void karmedbanditExp3::reset () [inline]

resets weights

Definition at line 60 of file karmedbandit.h.

References w.

7.86.3.5 void karmedbanditExp3::reward (bool *r*) [inline]

call this if you want to reward (*r*==true) or penalize (*r*==false) the previous decision

Definition at line 52 of file karmedbandit.h.

References g, last, lastp, and w.

7.86.3.6 void karmedbanditExp3::setGamma (double *gammap*) [inline]

sets gamma parameter

Definition at line 67 of file karmedbandit.h.

References g.

7.86.4 Member Data Documentation**7.86.4.1 double [karmedbanditExp3::g](#) [protected]**

gamma

Definition at line 74 of file karmedbandit.h.

7.86.4.2 unsigned int [karmedbanditExp3::last](#) [protected]

the last choice

Definition at line 73 of file karmedbandit.h.

7.86.4.3 double [karmedbanditExp3::lastp](#) [protected]

prob of last choice

Definition at line 72 of file karmedbandit.h.

7.86.4.4 `std::vector<double>` [karmedbanditExp3::w](#) [protected]

the weights

Definition at line 71 of file karmedbandit.h.

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

- [karmedbandit.h](#)

7.87 `karmedbanditExp3_1` Class Reference

```
#include <karmedbandit.h>
```

7.87.1 Detailed Description

Makes decisions regarding an adversarial k-armed bandit.

Uses algorithms described in: The non-stochastic multi-armed bandit problem Auer, Cesa-Bianchi, Freund, and Schapire October 14, 2002

Definition at line 83 of file `karmedbandit.h`.

Public Member Functions

- `karmedbanditExp3_1` (unsigned int k)
constructor, pass the number of arms
- unsigned int `decide` ()
returns the next choice, [0:k-1]
- void `reward` (bool rew)
call this if you want to reward ($r==true$) or penalize ($r==false$) the previous decision

Protected Member Functions

- void `restart` ()
restarts exp3

Protected Attributes

- unsigned int `r`
the number of restarts
- double `gr`
the γ_r parameter
- unsigned int `last`
the last choice

- `std::vector< double > G`

the G -hat's

- `karmedbanditExp3 exp3`

runs `exp3` within this

7.87.2 Constructor & Destructor Documentation

7.87.2.1 `karmedbanditExp3_1::karmedbanditExp3_1 (unsigned int k)` `[inline]`

constructor, pass the number of arms

Definition at line 86 of file `karmedbandit.h`.

References `exp3`, `G`, `gr`, `last`, `r`, and `restart()`.

7.87.3 Member Function Documentation

7.87.3.1 `unsigned int karmedbanditExp3_1::decide ()` `[inline]`

returns the next choice, $[0:k-1]$

Definition at line 93 of file `karmedbandit.h`.

References `karmedbanditExp3::decide()`, `exp3`, `G`, `karmedbanditExp3::getGamma()`, `karmedbanditExp3::getK()`, `gr`, `last`, and `restart()`.

7.87.3.2 `void karmedbanditExp3_1::restart ()` `[inline, protected]`

restarts `exp3`

Definition at line 112 of file `karmedbandit.h`.

References `exp3`, `karmedbanditExp3::getK()`, `gr`, `r`, and `karmedbanditExp3::setGamma()`.

7.87.3.3 `void karmedbanditExp3_1::reward (bool rew)` `[inline]`

call this if you want to reward ($r==true$) or penalize ($r==false$) the previous decision

Definition at line 105 of file `karmedbandit.h`.

References `exp3`, `G`, `last`, and `karmedbanditExp3::reward()`.

7.87.4 Member Data Documentation

7.87.4.1 `karmedbanditExp3` `karmedbanditExp3_1::exp3` [protected]

runs exp3 within this

Definition at line 126 of file karmedbandit.h.

7.87.4.2 `std::vector<double>` `karmedbanditExp3_1::G` [protected]

the G-hat's

Definition at line 125 of file karmedbandit.h.

7.87.4.3 `double` `karmedbanditExp3_1::gr` [protected]

the gamma_r parameter

Definition at line 123 of file karmedbandit.h.

7.87.4.4 `unsigned int` `karmedbanditExp3_1::last` [protected]

the last choice

Definition at line 124 of file karmedbandit.h.

7.87.4.5 `unsigned int` `karmedbanditExp3_1::r` [protected]

the number of restarts

Definition at line 122 of file karmedbandit.h.

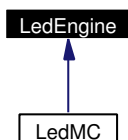
The documentation for this class was generated from the following file:

- [karmedbandit.h](#)

7.88 LedEngine Class Reference

```
#include <LedEngine.h>
```

Inheritance diagram for LedEngine:



7.88.1 Detailed Description

Provides basic LED effects to anything that inherits from (recommended method for MotionCommands) or instantiates it (just in case you have reason to).

Provides a collection of special effects so that the code can be reused various places as feedback to the user.

Cycling ("pulsing") and single-value setting are mutually exclusive - one will cut off the other

A flash will invert and override the current setting, so that it will "reset" after the flash. Flashes change mid-range values to extremes to make the flash visible (ie not just (1-current)) Normal invert will do simple inverses (just (1-current))

[getSetting\(\)](#) returns value of last [set\(\)](#); [getValue\(\)](#) returns what's actually being returned to Motion at the moment

There's some nice functions for using the LEDs to display numbers. This is handy for when you want to be free of the terminal.

Definition at line 31 of file LedEngine.h.

Public Types

- enum [numStyle_t](#) { [onedigit](#), [twodigit](#) }

Use these to specify a style for displaying numbers using [displayNumber\(\)](#).

- enum [percentStyle_t](#) { [major](#), [minor](#), [none](#) }

Use these to specify a style for displaying a percentage value [0-1] using [displayPercent\(\)](#).

Public Member Functions

- [LedEngine](#) ()
constructor - don't forget to call if you inherit
- virtual [~LedEngine](#) ()
destructor
- int [updateLEDs](#) (const [MotionCommand](#) *caller, LEDBitMask_t mask=All-LEDMask)
call this from a MotionCommand's updateOutputs() - makes calls to [MotionManager](#) to update LED values
- int [updateLEDs](#) ([OutputCmd](#) cmds[NumLEDs])
call this from a MotionCommand's updateOutputs() - performs the calculations to update LEDs' values
- int [updateLEDFrames](#) ([OutputCmd](#) cmds[NumLEDs][NumFrames])
call this from a MotionCommand's updateOutputs() - performs the calculations to update LEDs' values
- int [isDirty](#) ()
returns true if there are changes since the last [updateLEDs\(\)](#)
- void [invert](#) (LEDBitMask_t leds)
sets the leds specified by leds to the inverse of their current value
- void [cset](#) (LEDBitMask_t leds, float value)
sets the leds specified by leds to value, clears all the rest
- void [set](#) (LEDBitMask_t leds, float value)
sets the leds specified by leds to value
- void [cflash](#) (LEDBitMask_t leds, float value, unsigned int ms)
sets the leds specified by leds to value for ms milliseconds, then sets back. Clears ~leds
- void [flash](#) (LEDBitMask_t leds, unsigned int ms=500)
sets the leds specified by leds to value for ms milliseconds, then sets back.
- void [ccycle](#) (LEDBitMask_t leds, unsigned int period, float amp, float offset=0, int phase=0)
causes the leds specified by leds to cycle between low and high, clears others. See [cycle\(\)](#) for parameter documentation.

- void [cycle](#) (LEDBitMask_t leds, unsigned int period, float amp, float offset=0, int phase=0)
causes the leds specified by leds to cycle between low and high
- void [clear](#) ()
sets all leds to 0.
- float [getSetting](#) (LEDOffset_t led_id)
returns the current setting of the LED specified by led_id (the value you passed in [set\(\)](#))
- float [getValue](#) (LEDOffset_t led_id, unsigned int planahead=0)
returns the current value of the LED specified by led_id (the value being expressed - may change if cycling for instance)
- void [displayNumber](#) (int x, [numStyle_t](#) style)
Allows convenient display of numerical information to the LEDs on the face.
- void [displayPercent](#) (float x, [percentStyle_t](#) left_style, [percentStyle_t](#) right_style)
Allows convenient display of percentage information to the LEDs on the face.

Static Public Attributes

- const LEDBitMask_t [ERS210numMasks](#) [11]
holds a series of bit masks for the onedigit style of numerical display (0-10 and '.')
- const LEDBitMask_t [ERS220numMasks](#) [11]

Protected Member Functions

- float [calcValue](#) (unsigned int i, unsigned int t)
Calculates the current value of led i for current time t.
- void [setOneOfTwo](#) (unsigned int x, unsigned int low, unsigned int mid, unsigned int high)
used by [displayNumber\(\)](#) to determine settings of LEDs when using #numStyle_t::twodigit
- void [setColumn](#) (float x, unsigned int low, unsigned int mid, unsigned int high, unsigned int top)

used by [displayPercent\(\)](#) to determine settings of LEDs

Static Protected Member Functions

- float [calcInvert](#) (float value)
Performs the 'invert' calculation based on current value (returns 1-value).
- float [calcFlash](#) (float value)
Performs the 'flash' calculation based on current value (uses [calcInvert\(\)](#) if value upper or lower third, 0 or 1 otherwise).
- float [calcCycle](#) (unsigned int period, float amp, float offset, unsigned int t)
Performs the 'cycle' calculation based on desired period, amplituted, amplitude offset, and time since start. See [cycle\(\)](#).

Protected Attributes

- [LEDInfo infos](#) [NumLEDs]
the information regarding each of the LEDs
- unsigned int [numCycling](#)
the number of LEDs currently cycling (if non-zero, always dirty)
- bool [dirty](#)
true if changes since last updateLEDs
- unsigned int [dirtyTime](#)
the time at which it becomes dirty again (if flashing)

7.88.2 Member Enumeration Documentation

7.88.2.1 enum [LedEngine::numStyle_t](#)

Use these to specify a style for displaying numbers using [displayNumber\(\)](#).

Enumeration values:

onedigit can display a number -9 thru 9, using #numMask. For negative numbers, blinks the top bar - fast if it's supposed to be on, slow if it would otherwise be off

twodigit can display a number -99 thru 99, using [setOneOfTwo\(\)](#). For negative numbers, sets the top bar to 1 (off otherwise).

Definition at line 82 of file LedEngine.h.

7.88.2.2 enum [LedEngine::percentStyle_t](#)

Use these to specify a style for displaying a percentage value [0-1] using [displayPercent\(\)](#).

Enumeration values:

- major** shows overall value
- minor** shows value within major tick
- none** if you want to leave blank

Definition at line 87 of file LedEngine.h.

7.88.3 Constructor & Destructor Documentation

7.88.3.1 [LedEngine::LedEngine \(\)](#)

constructor - don't forget to call if you inherit

Definition at line 35 of file LedEngine.cc.

References [clear\(\)](#), [LedEngine::LEDInfo::flashtime](#), [infos](#), [ERS210Info::NumLEDs](#), and [LedEngine::LEDInfo::starttime](#).

7.88.3.2 [virtual LedEngine::~~LedEngine \(\)](#) [[inline](#), [virtual](#)]

destructor

Definition at line 36 of file LedEngine.h.

7.88.4 Member Function Documentation

7.88.4.1 [float LedEngine::calcCycle \(unsigned int *period*, float *amp*, float *offset*, unsigned int *t*\)](#) [[inline](#), [static](#), [protected](#)]

Performs the 'cycle' calculation based on desired period, amplituted, amplitude offset, and time since start. See [cycle\(\)](#).

Definition at line 121 of file LedEngine.h.

7.88.4.2 **float LedEngine::calcFlash (float *value*)** [inline, static, protected]

Performs the 'flash' calculation based on current value (uses [calcInvert\(\)](#) if value upper or lower third, 0 or 1 otherwise).

Definition at line 114 of file LedEngine.h.

References [calcInvert\(\)](#).

7.88.4.3 **float LedEngine::calcInvert (float *value*)** [inline, static, protected]

Performs the 'invert' calculation based on current value (returns 1-value).

Definition at line 110 of file LedEngine.h.

7.88.4.4 **float LedEngine::calcValue (unsigned int *i*, unsigned int *t*)** [inline, protected]

Calculates the current value of led *i* for current time *t*.

Definition at line 127 of file LedEngine.h.

References [calcCycle\(\)](#), [LedEngine::LEDInfo::flashtime](#), and [infos](#).

7.88.4.5 **void LedEngine::ccycle (LEDBitMask_t *leds*, unsigned int *period*, float *amp*, float *offset* = 0, int *phase* = 0)** [inline]

causes the leds specified by *leds* to cycle between low and high, clears others. See [cycle\(\)](#) for parameter documentation.

Definition at line 65 of file LedEngine.h.

References [clear\(\)](#), and [cycle\(\)](#).

7.88.4.6 **void LedEngine::cflash (LEDBitMask_t *leds*, float *value*, unsigned int *ms*)**

sets the leds specified by *leds* to *value* for *ms* milliseconds, then sets back. Clears ~leds

Definition at line 109 of file LedEngine.cc.

References [dirty](#), [dirtyTime](#), [LedEngine::LEDInfo::flashtime](#), [LedEngine::LEDInfo::flashvalue](#), [get_time\(\)](#), [infos](#), and [ERS210Info::NumLEDs](#).

7.88.4.7 void LedEngine::clear ()

sets all leds to 0.

Definition at line 167 of file LedEngine.cc.

References `dirty`, `infos`, `LedEngine::LEDInfo::isCycling`, `numCycling`, `ERS210Info::NumLEDs`, and `LedEngine::LEDInfo::value`.

7.88.4.8 void LedEngine::cset (LEDBitMask_t leds, float value) [inline]

sets the leds specified by *leds* to *value*, clears all the rest

Definition at line 57 of file LedEngine.h.

References `clear()`, and `set()`.

7.88.4.9 void LedEngine::cycle (LEDBitMask_t leds, unsigned int period, float amp, float offset = 0, int phase = 0)

causes the leds specified by *leds* to cycle between low and high

Parameters:

leds the bitmask of leds to apply this to

period the period of the cycle (milliseconds), includes an on and off

amp the amplitude of the cycle - note that this is clipped at 0 and 1.

offset the vertical offset of the cycle - simply shifts the baseline of the cycle up or down

phase the phase within the cycle to start at (specify in milliseconds)

When this function is called, the starting time is stored as current time + phase.

The equation used is

$$\cos\left(\frac{2\pi(t - starttime)}{period}\right) * \left(\frac{-amp}{2}\right) + .5 + offset$$

The idea is that with a amplitude=1 and offset=0, it will start at 0, ramp up to 1, and then ramp down again. The arguments to this function will let you control all parameters of the cycle.

You can get a blink-on/off instead of cycle on/off by using a very large amplitude.

Definition at line 150 of file LedEngine.cc.

References `LedEngine::LEDInfo::amp`, `dirty`, `get_time()`, `infos`, `LedEngine::LEDInfo::isCycling`, `numCycling`, `ERS210Info::NumLEDs`, `LedEngine::LEDInfo::offset`, `LedEngine::LEDInfo::period`, and `LedEngine::LEDInfo::starttime`.

7.88.4.10 void LedEngine::displayNumber (int *x*, [numStyle_t style](#))

Allows convenient display of numerical information to the LEDs on the face.

If overflow occurs, the face LEDs are set to flash on and off 3 every 333 milliseconds

Definition at line 176 of file LedEngine.cc.

References ERS210Info::BotLLEDOffset, ERS210Info::BotRLEDOffset, ccycle(), clear(), ERS210numMasks, WorldState::ERS220Mask, ERS220numMasks, ERS210Info::FaceLEDMask, infos, ERS210Info::LEDBitMask_t, ERS210Info::LEDOffset, ERS210Info::MidLLEDOffset, ERS210Info::MidRLEDOffset, onedigit, WorldState::robotDesign, set(), setOneOfTwo(), state, ERS210Info::TopBrLEDMask, ERS210Info::TopBrLEDOffset, ERS210Info::TopLLEDOffset, ERS210Info::TopRLEDOffset, and twodigit.

7.88.4.11 void LedEngine::displayPercent (float *x*, [percentStyle_t left_style](#), [percentStyle_t right_style](#))

Allows convenient display of percentage information to the LEDs on the face.

Besides allowing a two-digit display, the 'edge' bar for each type is blinked to denote how full it is. So you can get up to a two-digit, base 5 display, with an extra digit of estimated value.

If overflow (>1) occurs, sets everything to .75.

If underflow (<0) occurs, sets everything to .25.

The left and right columns are combined with an OR operation. (they overlap on the top bar) Left and right designations are *dog centric!*

Definition at line 234 of file LedEngine.cc.

References ERS210Info::BotLLEDMask, ERS210Info::BotRLEDMask, clear(), ERS210Info::FaceLEDMask, major, ERS210Info::MidLLEDMask, ERS210Info::MidRLEDMask, minor, set(), setColumn(), ERS210Info::TopBrLEDMask, ERS210Info::TopLLEDMask, and ERS210Info::TopRLEDMask.

7.88.4.12 void LedEngine::flash (LEDBitMask_t *leds*, unsigned int *ms* = 500)

sets the leds specified by *leds* to *value* for *ms* milliseconds, then sets back.

Definition at line 121 of file LedEngine.cc.

References calcFlash(), calcValue(), dirty, dirtyTime, LedEngine::LEDInfo::flashtime, LedEngine::LEDInfo::flashvalue, get_time(), infos, and ERS210Info::NumLEDs.

7.88.4.13 float LedEngine::getSetting (LEDOffset_t *led_id*) [inline]

returns the current setting of the LED specified by *led_id* (the value you passed in [set\(\)](#))

Definition at line 72 of file LedEngine.h.

References [infos](#), and [ERS210Info::LEDOffset](#).

7.88.4.14 float LedEngine::getValue (LEDOffset_t *led_id*, unsigned int *planahead* = 0) [inline]

returns the current value of the LED specified by *led_id* (the value being expressed - may change if cycling for instance)

Definition at line 74 of file LedEngine.h.

References [calcValue\(\)](#), [get_time\(\)](#), and [ERS210Info::LEDOffset](#).

7.88.4.15 void LedEngine::invert (LEDBitMask_t *leds*)

sets the leds specified by *leds* to the inverse of their current value

Definition at line 85 of file LedEngine.cc.

References [LedEngine::LEDInfo::amp](#), [dirty](#), [infos](#), [LedEngine::LEDInfo::isCycling](#), [ERS210Info::NumLEDs](#), and [LedEngine::LEDInfo::value](#).

7.88.4.16 int LedEngine::isDirty ()

returns true if there are changes since the last [updateLEDs\(\)](#)

Reimplemented in [LedMC](#).

Definition at line 43 of file LedEngine.cc.

References [dirty](#), [dirtyTime](#), [LedEngine::LEDInfo::flashtime](#), [get_time\(\)](#), [infos](#), [numCycling](#), and [ERS210Info::NumLEDs](#).

7.88.4.17 void LedEngine::set (LEDBitMask_t *leds*, float *value*)

sets the leds specified by *leds* to *value*

Definition at line 96 of file LedEngine.cc.

References [dirty](#), [infos](#), [LedEngine::LEDInfo::isCycling](#), [numCycling](#), [ERS210Info::NumLEDs](#), and [LedEngine::LEDInfo::value](#).

7.88.4.18 void LedEngine::setColumn (float *x*, unsigned int *low*, unsigned int *mid*, unsigned int *high*, unsigned int *top*) [protected]

used by [displayPercent\(\)](#) to determine settings of LEDs

Definition at line 256 of file LedEngine.cc.

References ERS210Info::LEDBitMask_t, and set().

7.88.4.19 void LedEngine::setOneOfTwo (unsigned int *x*, unsigned int *low*, unsigned int *mid*, unsigned int *high*) [protected]

used by [displayNumber\(\)](#) to determine settings of LEDs when using #numStyle_t::twodigit

Definition at line 210 of file LedEngine.cc.

References infos, and LedEngine::LEDInfo::value.

7.88.4.20 int LedEngine::updateLEDFrames ([OutputCmd](#) *cmds*[NumLEDs][NumFrames])

call this from a MotionCommand's updateOutputs() - performs the calculations to update LEDs' values

Parameters:

cmds on input, used for weight values - on return, holds the resulting OutputCmd's

Definition at line 75 of file LedEngine.cc.

References calcValue(), dirty, ERS210Info::FrameTime, get_time(), ERS210Info::NumFrames, and ERS210Info::NumLEDs.

7.88.4.21 int LedEngine::updateLEDs ([OutputCmd](#) *cmds*[NumLEDs])

call this from a MotionCommand's updateOutputs() - performs the calculations to update LEDs' values

Parameters:

cmds on input, used for weight values - on return, holds the resulting OutputCmd's

Definition at line 66 of file LedEngine.cc.

References calcValue(), dirty, get_time(), and ERS210Info::NumLEDs.

7.88.4.22 int LedEngine::updateLEDs (const [MotionCommand](#) * *caller*, LEDBitMask_t *mask* = AllLEDMask)

call this from a MotionCommand's updateOutputs() - makes calls to [MotionManager](#) to update LED values

Parameters:

caller pass the "parent" motioncommand's address here (usually will pass 'this')

mask a bitmask of which leds to update (uses weight of 1)

Definition at line 55 of file LedEngine.cc.

References [calcValue\(\)](#), [dirty](#), [ERS210Info::FrameTime](#), [get_time\(\)](#), [ERS210Info::LEDOffset](#), [motman](#), [ERS210Info::NumFrames](#), [ERS210Info::NumLEDs](#), and [MotionManager::setOutput\(\)](#).

7.88.5 Member Data Documentation

7.88.5.1 bool [LedEngine::dirty](#) [protected]

true if changes since last updateLEDs

Definition at line 154 of file LedEngine.h.

7.88.5.2 unsigned int [LedEngine::dirtyTime](#) [protected]

the time at which it becomes dirty again (if flashing)

Definition at line 155 of file LedEngine.h.

7.88.5.3 const LEDBitMask_t [LedEngine::ERS210numMasks](#) [static]

Initial value:

```
{
  ERS210Info::BotRLEDMask | ERS210Info::BotLLEDMask | ERS210Info::TopBrLEDMask,
  ERS210Info::BotLLEDMask | ERS210Info::MidLLEDMask | ERS210Info::TopLLEDMask,
  ERS210Info::BotRLEDMask | ERS210Info::BotLLEDMask | ERS210Info::TopLLEDMask | ERS210Info::TopBrLEDMask,
  ERS210Info::BotRLEDMask | ERS210Info::BotLLEDMask | ERS210Info::MidRLEDMask | ERS210Info::TopLLEDMask | ERS210Info::TopBrLEDMask,
  ERS210Info::BotLLEDMask | ERS210Info::MidLLEDMask | ERS210Info::TopRLEDMask | ERS210Info::TopLLEDMask,
  ERS210Info::BotRLEDMask | ERS210Info::BotLLEDMask | ERS210Info::TopRLEDMask | ERS210Info::TopBrLEDMask,
  ERS210Info::BotRLEDMask | ERS210Info::BotLLEDMask | ERS210Info::MidRLEDMask | ERS210Info::MidLLEDMask | ERS210Info::TopLLEDMask,
  ERS210Info::BotLLEDMask | ERS210Info::MidLLEDMask | ERS210Info::TopLLEDMask | ERS210Info::TopBrLEDMask,
  ERS210Info::BotRLEDMask | ERS210Info::BotLLEDMask | ERS210Info::MidRLEDMask | ERS210Info::MidLLEDMask | ERS210Info::TopLLEDMask,
  ERS210Info::BotLLEDMask | ERS210Info::MidLLEDMask | ERS210Info::TopRLEDMask | ERS210Info::TopLLEDMask | ERS210Info::TopBrLEDMask,
  ERS210Info::BotLLEDMask
}
```

holds a series of bit masks for the onedigit style of numerical display (0-10 and '.')

the hope is that these actually resemble the shapes of the numbers so people can recognize them more easily - without converting base 2 in their heads.

Definition at line 7 of file LedEngine.cc.

7.88.5.4 `const LEDBitMask_t LedEngine::ERS220numMasks` [static]

Initial value:

```
{
  ERS220Info::ModeLEDMask,
  ERS220Info::FaceBackLeftLEDMask,
  ERS220Info::FaceBackLeftLEDMask | ERS220Info::FaceCenterLeftLEDMask,
  ERS220Info::FaceBackLeftLEDMask | ERS220Info::FaceCenterLeftLEDMask | ERS220Info::FaceFrontLeftLEDMask,
  ERS220Info::FaceBackLeftLEDMask | ERS220Info::FaceCenterLeftLEDMask | ERS220Info::FaceFrontLeftLEDMask,
  ERS220Info::FaceBackLeftLEDMask | ERS220Info::FaceCenterLeftLEDMask | ERS220Info::FaceFrontLeftLEDMask,
  ERS220Info::FaceBackLeftLEDMask | ERS220Info::FaceCenterLeftLEDMask | ERS220Info::FaceFrontLeftLEDMask,
  ERS220Info::FaceBackLeftLEDMask | ERS220Info::FaceCenterLeftLEDMask | ERS220Info::FaceFrontLeftLEDMask,
  ERS220Info::FaceBackLeftLEDMask | ERS220Info::FaceCenterLeftLEDMask | ERS220Info::FaceFrontLeftLEDMask,
  ERS220Info::FaceBackLeftLEDMask | ERS220Info::FaceCenterLeftLEDMask | ERS220Info::FaceFrontLeftLEDMask,
  ERS220Info::FaceBackLeftLEDMask | ERS220Info::FaceCenterLeftLEDMask | ERS220Info::FaceFrontLeftLEDMask,
  ERS220Info::FaceBackLeftLEDMask | ERS220Info::FaceCenterLeftLEDMask | ERS220Info::FaceFrontLeftLEDMask,
  ERS220Info::FaceBackLeftLEDMask | ERS220Info::FaceCenterLeftLEDMask | ERS220Info::FaceFrontLeftLEDMask,
  ERS220Info::FaceFrontLeftLEDMask
}
```

bit masks for the ondigit style of numerical display - just count the LEDs on the head

Definition at line 20 of file LedEngine.cc.

7.88.5.5 `LEDInfo LedEngine::infos[NumLEDs]` [protected]

the information regarding each of the LEDs

Definition at line 152 of file LedEngine.h.

7.88.5.6 `unsigned int LedEngine::numCycling` [protected]

the number of LEDs currently cycling (if non-zero, always dirty)

Definition at line 153 of file LedEngine.h.

The documentation for this class was generated from the following files:

- [LedEngine.h](#)
- [LedEngine.cc](#)

7.89 LedEngine::LEDInfo Struct Reference

```
#include <LedEngine.h>
```

7.89.1 Detailed Description

Holds all the information needed by each of the LEDs.

Definition at line 141 of file LedEngine.h.

Public Attributes

- float [value](#)
the current value being expressed
- float [amp](#)
the amplitude of the cycle (if cycling)
- unsigned int [period](#)
the period of the cycle (if cycling)
- unsigned int [starttime](#)
the start time of the cycle (if cycling)
- float [offset](#)
the phase shift from normal of the cycle (if cycling)
- float [flashvalue](#)
the value being 'flashed' (only valid if current time is less than flashtime)
- unsigned int [flashtime](#)
the time the 'flash' should retire
- bool [isCycling](#)
true if in cycle mode

7.89.2 Member Data Documentation

7.89.2.1 float [LedEngine::LEDInfo::amp](#)

the amplitude of the cycle (if cycling)

Definition at line 143 of file LedEngine.h.

7.89.2.2 unsigned int [LedEngine::LEDInfo::flashtime](#)

the time the 'flash' should retire

Definition at line 148 of file LedEngine.h.

7.89.2.3 float [LedEngine::LEDInfo::flashvalue](#)

the value being 'flashed' (only valid if current time is less than flashtime)

Definition at line 147 of file LedEngine.h.

7.89.2.4 bool [LedEngine::LEDInfo::isCycling](#)

true if in cycle mode

Definition at line 149 of file LedEngine.h.

7.89.2.5 float [LedEngine::LEDInfo::offset](#)

the phase shift from normal of the cycle (if cycling)

Definition at line 146 of file LedEngine.h.

7.89.2.6 unsigned int [LedEngine::LEDInfo::period](#)

the period of the cycle (if cycling)

Definition at line 144 of file LedEngine.h.

7.89.2.7 unsigned int [LedEngine::LEDInfo::starttime](#)

the start time of the cycle (if cycling)

Definition at line 145 of file LedEngine.h.

7.89.2.8 float [LedEngine::LEDInfo::value](#)

the current value being expressed

Definition at line 142 of file LedEngine.h.

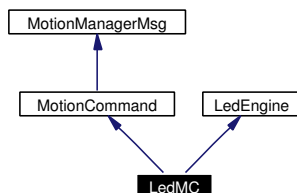
The documentation for this struct was generated from the following file:

- [LedEngine.h](#)

7.90 LedMC Class Reference

```
#include <LedMC.h>
```

Inheritance diagram for LedMC:



7.90.1 Detailed Description

This is just a simple wrapper - you probably want to be looking at [LedEngine.h](#).

This is handy if all you want to do is control the LED's, but since other Motion-Commands will probably also want to make use of the LEDs, they can just use the engine component to do all the work.

Definition at line 14 of file LedMC.h.

Public Member Functions

- [LedMC](#) ()
constructor
- virtual [~LedMC](#) ()
destructor
- virtual int [updateOutputs](#) ()
updates the cmds from [LedEngine::updateLEDs\(\)](#)
- virtual int [isDirty](#) ()
returns true if there are changes since the last [updateLEDs\(\)](#)
- virtual int [isAlive](#) ()
- void [setWeights](#) (LEDBitMask_t leds, float weight)
Sets the JointCmd::weight of the LEDs specified by leds to weight.

Protected Attributes

- [OutputCmd cmds](#) [NumLEDs][NumFrames]
needed to store weight values of LEDs (useful to mark LEDs as unused)

7.90.2 Constructor & Destructor Documentation

7.90.2.1 LedMC::LedMC () [inline]

constructor

Definition at line 17 of file LedMC.h.

References [ERS210Info::AllLEDMask](#), and [setWeights\(\)](#).

7.90.2.2 virtual LedMC::~LedMC () [inline, virtual]

destructor

Definition at line 19 of file LedMC.h.

7.90.3 Member Function Documentation

7.90.3.1 virtual int LedMC::isAlive () [inline, virtual]

Todo

let's make this smarter so you can flash the LED's and have it autoprune

Implements [MotionCommand](#).

Definition at line 30 of file LedMC.h.

7.90.3.2 virtual int LedMC::isDirty () [inline, virtual]

returns true if there are changes since the last [updateLEDs\(\)](#)

Reimplemented from [LedEngine](#).

Definition at line 29 of file LedMC.h.

References [LedEngine::isDirty\(\)](#).

7.90.3.3 void LedMC::setWeights (LEDBitMask_t *leds*, float *weight*) [inline]

Sets the JointCmd::weight of the LEDs specified by *leds* to *weight*.

Definition at line 33 of file LedMC.h.

References [cmds](#), [ERS210Info::NumFrames](#), [ERS210Info::NumLEDs](#), and [OutputCmd::weight](#).

7.90.3.4 virtual int LedMC::updateOutputs () [inline, virtual]

updates the cmds from [LedEngine::updateLEDs\(\)](#)

Implements [MotionCommand](#).

Definition at line 22 of file LedMC.h.

References [cmds](#), [ERS210Info::LEDOffset](#), [motman](#), [ERS210Info::NumLEDs](#), [MotionManager::setOutput\(\)](#), [LedEngine::updateLEDFrames\(\)](#), and [OutputCmd::weight](#).

7.90.4 Member Data Documentation

7.90.4.1 [OutputCmd](#) LedMC::cmds[NumLEDs][NumFrames] [protected]

needed to store weight values of LEDs (useful to mark LEDs as unused)

Definition at line 41 of file LedMC.h.

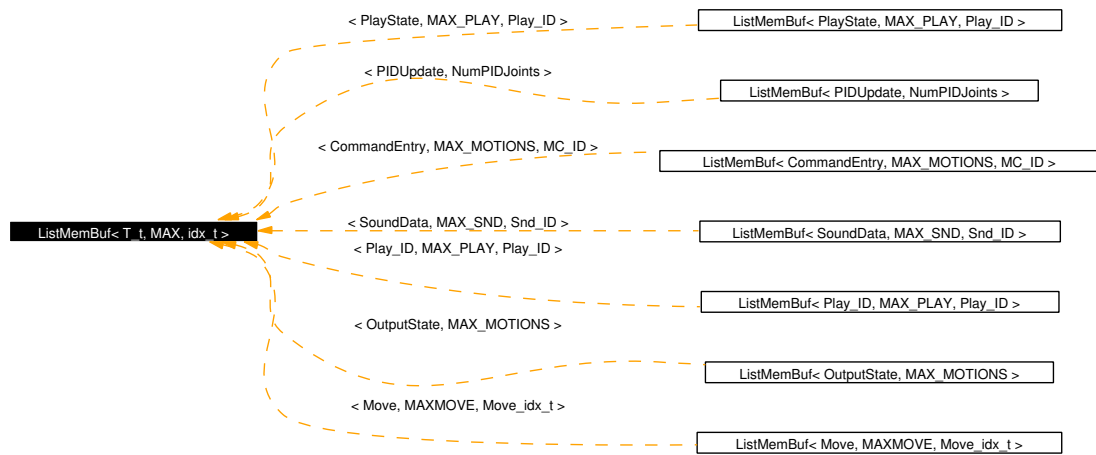
The documentation for this class was generated from the following file:

- [LedMC.h](#)

7.91 ListMemBuf< T.t, MAX, idx.t > Class Template Reference

```
#include <ListMemBuf.h>
```

Inheritance diagram for ListMemBuf< T.t, MAX, idx.t >:



7.91.1 Detailed Description

template<class T.t, unsigned int MAX, class idx.t = unsigned short> class ListMemBuf< T.t, MAX, idx.t >

Provides some degree of dynamic allocation of a templated type from a buffer of set size.

Think of this as a self-contained mini-malloc...

This is handy for classes which inhabit a shared memory region, where it's a bad idea to have pointers to other memory. By instantiating one of these in your class, you can allocate space internally for up to MAX objects of type T.t. ListMemBuf will worry about keeping track of which ones are in use or are free.

Each time you request a entry to be created, the destructor will be called followed by the the default constructor before it is given to you, so the fields should be reliably 'fresh', not what was in the entry last time it was used.

Definition at line 20 of file ListMemBuf.h.

Public Types

- typedef T_t [T](#)
Allows outside access to storage type.
- typedef idx_t [index_t](#)
Allows outside access to index type.

Public Member Functions

- [ListMemBuf](#) ()
constructor
- [index_t getMaxCapacity](#) () const
returns the maximum number of objects which can be used at any given time
- [index_t size](#) () const
returns the current number of objects in use
- [index_t countf](#) () const
for debugging, should equal size
- [index_t countb](#) () const
for debugging, should equal size
- bool [empty](#) () const
returns true if no objects are in use
- [T & operator\[\]](#) (unsigned int x)
allows direct access to elements - be careful, can access 'free' elements this way
- const [T & operator\[\]](#) (unsigned int x) const
allows direct access to elements - be careful, can access 'free' elements this way
- [index_t begin](#) () const
returns index of first used entry
- [T & front](#) ()
returns reference to first used entry
- const [T & front](#) () const

returns const reference to first used entry

- `index_t end () const`

returns the one-past-end index

- `T & back ()`

returns reference to last used entry

- `const T & back () const`

returns const reference to last used entry

- `index_t new_front ()`

pushes a 'blank' entry on the front of the used list

- `index_t push_front (const T &data)`

pushes an entry on the front of the used chain and assigns data to it

- `void pop_front ()`

pops the front of the used chain

- `void pop_front (T &ret)`

pops the front of the chain into ret

- `index_t new_back ()`

pushes a 'blank' entry on the back of the used list

- `index_t push_back (const T &data)`

pushes an entry on the back of the used chain and assigns data to it

- `void pop_back ()`

pops the last of the used chain

- `void pop_back (T &ret)`

pops the last of the used chain into ret

- `index_t new_before (index_t x)`

inserts a 'blank' entry before element x in the used chain

- `index_t push_before (index_t x, const T &data)`

inserts a 'blank' entry before element x in the used chain and assigns data to it

- `index_t new_after (index_t x)`

inserts a 'blank' entry after element x in the used chain

- `index_t push_after (index_t x, const T &data)`
inserts a 'blank' entry after element x in the used chain and assigns data to it
- `void erase (index_t x)`
removes element x from the used chain
- `void clear ()`
frees all used entries
- `void swap (index_t a, index_t b)`
swaps the two entries' position in the list
- `index_t next (index_t x) const`
returns the next used element following x
- `index_t prev (index_t x) const`
returns the preceeding used element following x

Static Public Attributes

- `const unsigned int MAX_ENTRIES = MAX`
Allows outside access to number of entries.

Protected Member Functions

- `index_t pop_free ()`
removes an element from the front of the free list, returns its index
- `void push_free (index_t x)`
pushes x onto the back of the free list

Protected Attributes

- `entry_t entries [MAX_ENTRIES]`
the main block of data

- [index.t activeBegin](#)
beginning of used chain
- [index.t activeBack](#)
end of used chain
- [index.t freeBegin](#)
beginning of free chain
- [index.t freeBack](#)
end of free chain
- [index.t cursize](#)
current number of used elements

7.91.2 Member Typedef Documentation

7.91.2.1 `template<class T.t, unsigned int MAX, class idx.t = unsigned short>
typedef idx.t ListMemBuf< T.t, MAX, idx.t >::index.t`

Allows outside access to index type.

Definition at line 30 of file ListMemBuf.h.

7.91.2.2 `template<class T.t, unsigned int MAX, class idx.t = unsigned short>
typedef T.t ListMemBuf< T.t, MAX, idx.t >::T`

Allows outside access to storage type.

Definition at line 26 of file ListMemBuf.h.

7.91.3 Constructor & Destructor Documentation

7.91.3.1 `template<class T, unsigned int MAX, class index.t> ListMemBuf< T,
MAX, index.t >::ListMemBuf ()`

constructor

Definition at line 94 of file ListMemBuf.h.

References [ListMemBuf](#)< T.t, MAX, idx.t >::cursize, [ListMemBuf](#)< T.t, MAX, idx.t >::freeBack, [ListMemBuf](#)< T.t, MAX, idx.t >::MAX_ENTRIES, and [ListMemBuf](#)< T.t, MAX, idx.t >::push_free().

7.91.4 Member Function Documentation

7.91.4.1 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
const T& ListMemBuf< T_t, MAX, idx_t >::back () const [inline]`

returns const reference to last used entry

Definition at line 47 of file ListMemBuf.h.

7.91.4.2 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
T& ListMemBuf< T_t, MAX, idx_t >::back () [inline]`

returns reference to last used entry

Definition at line 46 of file ListMemBuf.h.

7.91.4.3 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
idx_t ListMemBuf< T_t, MAX, idx_t >::begin () const [inline]`

returns index of first used entry

Definition at line 41 of file ListMemBuf.h.

7.91.4.4 `template<class T, unsigned int MAX, class index_t> void
ListMemBuf< T, MAX, index_t >::clear ()`

frees all used entries

Definition at line 210 of file ListMemBuf.h.

References ListMemBuf< T_t, MAX, idx_t >::activeBack, ListMemBuf< T_t, MAX, idx_t >::activeBegin, ListMemBuf< T_t, MAX, idx_t >::cursize, ListMemBuf< T_t, MAX, idx_t >::end(), ListMemBuf< T_t, MAX, idx_t >::entries, ListMemBuf< T_t, MAX, idx_t >::freeBack, and ListMemBuf< T_t, MAX, idx_t >::freeBegin.

7.91.4.5 `template<class T, unsigned int MAX, class index_t> index_t
ListMemBuf< T, MAX, index_t >::countb () const`

for debugging, should equal size

Definition at line 112 of file ListMemBuf.h.

References ListMemBuf< T_t, MAX, idx_t >::begin(), ListMemBuf< T_t, MAX, idx_t >::end(), ListMemBuf< T_t, MAX, idx_t >::index_t, and ListMemBuf< T_t, MAX, idx_t >::prev().

7.91.4.6 `template<class T, unsigned int MAX, class index_t> index_t
ListMemBuf< T, MAX, index_t >::countf () const`

for debugging, should equal size

Definition at line 103 of file ListMemBuf.h.

References [ListMemBuf](#)< T.t, MAX, idx.t >::begin(), [ListMemBuf](#)< T.t, MAX, idx.t >::end(), [ListMemBuf](#)< T.t, MAX, idx.t >::index_t, and [ListMemBuf](#)< T.t, MAX, idx.t >::next().

7.91.4.7 `template<class T.t, unsigned int MAX, class idx_t = unsigned short>
bool ListMemBuf< T.t, MAX, idx_t >::empty () const [inline]`

returns true if no objects are in use

Definition at line 36 of file ListMemBuf.h.

7.91.4.8 `template<class T.t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T.t, MAX, idx_t >::end () const [inline]`

returns the one-past-end index

Definition at line 45 of file ListMemBuf.h.

7.91.4.9 `template<class T, unsigned int MAX, class index_t> void
ListMemBuf< T, MAX, index_t >::erase (index_t x)`

removes element *x* from the used chain

Definition at line 194 of file ListMemBuf.h.

References [ListMemBuf](#)< T.t, MAX, idx.t >::activeBack, [ListMemBuf](#)< T.t, MAX, idx.t >::activeBegin, [ListMemBuf](#)< T.t, MAX, idx.t >::entries, [ListMemBuf](#)< T.t, MAX, idx.t >::entry_t::next, [ListMemBuf](#)< T.t, MAX, idx.t >::pop_back(), [ListMemBuf](#)< T.t, MAX, idx.t >::pop_front(), and [ListMemBuf](#)< T.t, MAX, idx.t >::push_free().

7.91.4.10 `template<class T.t, unsigned int MAX, class idx_t = unsigned
short> const T& ListMemBuf< T.t, MAX, idx_t >::front () const
[inline]`

returns const reference to first used entry

Definition at line 43 of file ListMemBuf.h.

7.91.4.11 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
T& ListMemBuf< T_t, MAX, idx_t >::front () [inline]`

returns reference to first used entry

Definition at line 42 of file ListMemBuf.h.

7.91.4.12 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T_t, MAX, idx_t >::getMaxCapacity () const
[inline]`

returns the maximum number of objects which can be used at any given time

Definition at line 32 of file ListMemBuf.h.

7.91.4.13 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T_t, MAX, idx_t >::new_after (index_t x)
[inline]`

inserts a 'blank' entry after element *x* in the used chain

Definition at line 62 of file ListMemBuf.h.

7.91.4.14 `template<class T, unsigned int MAX, class index_t> index_t
ListMemBuf< T, MAX, index_t >::new_back ()`

pushes a 'blank' entry on the back of the used list

Definition at line 137 of file ListMemBuf.h.

References ListMemBuf< T_t, MAX, idx_t >::activeBack, ListMemBuf< T_t, MAX, idx_t >::activeBegin, ListMemBuf< T_t, MAX, idx_t >::end(), ListMemBuf< T_t, MAX, idx_t >::entries, ListMemBuf< T_t, MAX, idx_t >::index_t, ListMemBuf< T_t, MAX, idx_t >::entry_t::next, ListMemBuf< T_t, MAX, idx_t >::pop_free(), and ListMemBuf< T_t, MAX, idx_t >::entry_t::prev.

7.91.4.15 `template<class T, unsigned int MAX, class index_t> index_t
ListMemBuf< T, MAX, index_t >::new_before (index_t x)`

inserts a 'blank' entry before element *x* in the used chain

Definition at line 153 of file ListMemBuf.h.

References ListMemBuf< T_t, MAX, idx_t >::end(), ListMemBuf< T_t, MAX, idx_t >::entries, ListMemBuf< T_t, MAX, idx_t >::index_t, ListMemBuf< T_t, MAX, idx_t >::new_back(), ListMemBuf< T_t, MAX, idx_t >::new_front(), ListMemBuf<

T.t, MAX, idx.t >::entry_t::next, ListMemBuf< T.t, MAX, idx.t >::pop_free(), and ListMemBuf< T.t, MAX, idx.t >::entry_t::prev.

7.91.4.16 `template<class T, unsigned int MAX, class index_t> index_t
ListMemBuf< T, MAX, index_t >::new_front ()`

pushes a 'blank' entry on the front of the used list

Definition at line 121 of file ListMemBuf.h.

References ListMemBuf< T.t, MAX, idx.t >::activeBack, ListMemBuf< T.t, MAX, idx.t >::activeBegin, ListMemBuf< T.t, MAX, idx.t >::end(), ListMemBuf< T.t, MAX, idx.t >::entries, ListMemBuf< T.t, MAX, idx.t >::index_t, ListMemBuf< T.t, MAX, idx.t >::entry_t::next, ListMemBuf< T.t, MAX, idx.t >::pop_free(), and ListMemBuf< T.t, MAX, idx.t >::entry_t::prev.

7.91.4.17 `template<class T.t, unsigned int MAX, class idx.t = unsigned short>
index_t ListMemBuf< T.t, MAX, idx.t >::next (index_t x) const
[inline]`

returns the next used element following x

Definition at line 70 of file ListMemBuf.h.

7.91.4.18]

`template<class T.t, unsigned int MAX, class idx.t = unsigned short> const T& ListMemBuf< T.t, MAX, idx.t >::operator[] (unsigned int x) const [inline]`

allows direct access to elements - be careful, can access 'free' elements this way

Definition at line 39 of file ListMemBuf.h.

7.91.4.19]

`template<class T.t, unsigned int MAX, class idx.t = unsigned short> T& ListMemBuf< T.t, MAX, idx.t >::operator[] (unsigned int x) [inline]`

allows direct access to elements - be careful, can access 'free' elements this way

Definition at line 38 of file ListMemBuf.h.

7.91.4.20 `template<class T_t, unsigned int MAX, class idx_t = unsigned short> void ListMemBuf< T_t, MAX, idx_t >::pop_back (T & ret)`
[inline]

pops the last of the used chain into *ret*

Definition at line 57 of file ListMemBuf.h.

7.91.4.21 `template<class T, unsigned int MAX, class index_t> void ListMemBuf< T, MAX, index_t >::pop_back ()`

pops the last of the used chain

Definition at line 182 of file ListMemBuf.h.

References ListMemBuf< T_t, MAX, idx_t >::activeBack, ListMemBuf< T_t, MAX, idx_t >::activeBegin, ListMemBuf< T_t, MAX, idx_t >::end(), ListMemBuf< T_t, MAX, idx_t >::entries, ListMemBuf< T_t, MAX, idx_t >::index_t, ListMemBuf< T_t, MAX, idx_t >::entry_t::prev, and ListMemBuf< T_t, MAX, idx_t >::push_free().

7.91.4.22 `template<class T, unsigned int MAX, class index_t> index_t ListMemBuf< T, MAX, index_t >::pop_free ()` [protected]

removes an element from the front of the free list, returns its index

free list is a queue... pop front, push back - hopefully more robust with multi-threads is purposely sloppy with unused links, a little faster

Definition at line 306 of file ListMemBuf.h.

References ListMemBuf< T_t, MAX, idx_t >::cursize, ListMemBuf< T_t, MAX, idx_t >::end(), ListMemBuf< T_t, MAX, idx_t >::entries, ListMemBuf< T_t, MAX, idx_t >::freeBack, ListMemBuf< T_t, MAX, idx_t >::freeBegin, ListMemBuf< T_t, MAX, idx_t >::index_t, ListMemBuf< T_t, MAX, idx_t >::entry_t::next, and ListMemBuf< T_t, MAX, idx_t >::T.

7.91.4.23 `template<class T_t, unsigned int MAX, class idx_t = unsigned short> void ListMemBuf< T_t, MAX, idx_t >::pop_front (T & ret)`
[inline]

pops the front of the chain into *ret*

Definition at line 52 of file ListMemBuf.h.

7.91.4.24 `template<class T, unsigned int MAX, class index_t> void
ListMemBuf< T, MAX, index_t >::pop_front ()`

pops the front of the used chain

Definition at line 170 of file ListMemBuf.h.

References ListMemBuf< T.t, MAX, idx.t >::activeBack, ListMemBuf< T.t, MAX, idx.t >::activeBegin, ListMemBuf< T.t, MAX, idx.t >::end(), ListMemBuf< T.t, MAX, idx.t >::entries, ListMemBuf< T.t, MAX, idx.t >::index_t, ListMemBuf< T.t, MAX, idx.t >::entry_t::next, and ListMemBuf< T.t, MAX, idx.t >::push_free().

7.91.4.25 `template<class T.t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T.t, MAX, idx_t >::prev (index_t x) const
[inline]`

returns the preceeding used element following *x*

Definition at line 71 of file ListMemBuf.h.

7.91.4.26 `template<class T.t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T.t, MAX, idx_t >::push_after (index_t x, const
T & data) [inline]`

inserts a 'blank' entry after element *x* in the used chain and assigns *data* to it

Definition at line 63 of file ListMemBuf.h.

7.91.4.27 `template<class T.t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T.t, MAX, idx_t >::push_back (const T &
data) [inline]`

pushes an entry on the back of the used chain and assigns *data* to it

Definition at line 55 of file ListMemBuf.h.

7.91.4.28 `template<class T.t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T.t, MAX, idx_t >::push_before (index_t x,
const T & data) [inline]`

inserts a 'blank' entry before element *x* in the used chain and assigns *data* to it

Definition at line 60 of file ListMemBuf.h.

7.91.4.29 `template<class T, unsigned int MAX, class index_t> void
ListMemBuf< T, MAX, index_t >::push_free (index_t x)
[protected]`

pushes *x* onto the back of the free list

See also:

[pop_free\(\)](#)

Definition at line 322 of file ListMemBuf.h.

References ListMemBuf< T_t, MAX, idx_t >::cursize, ListMemBuf< T_t, MAX, idx_t >::end(), ListMemBuf< T_t, MAX, idx_t >::entries, ListMemBuf< T_t, MAX, idx_t >::freeBack, ListMemBuf< T_t, MAX, idx_t >::freeBegin, and ListMemBuf< T_t, MAX, idx_t >::operator[]().

7.91.4.30 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T_t, MAX, idx_t >::push_front (const T &
data) [inline]`

pushes an entry on the front of the used chain and assigns *data* to it

Definition at line 50 of file ListMemBuf.h.

7.91.4.31 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T_t, MAX, idx_t >::size () const [inline]`

returns the current number of objects in use

Definition at line 33 of file ListMemBuf.h.

7.91.4.32 `template<class T, unsigned int MAX, class index_t> void
ListMemBuf< T, MAX, index_t >::swap (index_t a, index_t b)`

swaps the two entries' position in the list

Definition at line 224 of file ListMemBuf.h.

References ListMemBuf< T_t, MAX, idx_t >::activeBack, ListMemBuf< T_t, MAX, idx_t >::activeBegin, ListMemBuf< T_t, MAX, idx_t >::end(), ListMemBuf< T_t, MAX, idx_t >::entries, ListMemBuf< T_t, MAX, idx_t >::index_t, ListMemBuf< T_t, MAX, idx_t >::entry_t::next, and ListMemBuf< T_t, MAX, idx_t >::entry_t::prev.

7.91.5 Member Data Documentation

7.91.5.1 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T_t, MAX, idx_t >::activeBack` [protected]

end of used chain

Definition at line 87 of file ListMemBuf.h.

7.91.5.2 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T_t, MAX, idx_t >::activeBegin` [protected]

beginning of used chain

Definition at line 86 of file ListMemBuf.h.

7.91.5.3 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T_t, MAX, idx_t >::cursize` [protected]

current number of used elements

Definition at line 90 of file ListMemBuf.h.

7.91.5.4 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
entry_t ListMemBuf< T_t, MAX, idx_t >::entries[MAX_ENTRIES]` [protected]

the main block of data

Definition at line 85 of file ListMemBuf.h.

7.91.5.5 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T_t, MAX, idx_t >::freeBack` [protected]

end of free chain

Definition at line 89 of file ListMemBuf.h.

7.91.5.6 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
index_t ListMemBuf< T_t, MAX, idx_t >::freeBegin` [protected]

beginning of free chain

Definition at line 88 of file ListMemBuf.h.

7.91.5.7 `template<class T_t, unsigned int MAX, class idx_t = unsigned short>
const unsigned int ListMemBuf< T_t, MAX, idx_t >::MAX_ENTRIES
= MAX [static]`

Allows outside access to number of entries.

Definition at line 28 of file ListMemBuf.h.

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

- [ListMemBuf.h](#)

7.92 ListMemBuf< T_t, MAX, idx_t >::entry_t Struct Reference

```
#include <ListMemBuf.h>
```

7.92.1 Detailed Description

template<class T_t, unsigned int MAX, class idx_t = unsigned short> struct ListMemBuf< T_t, MAX, idx_t >::entry_t

holds data about an entry in the free/used lists

Definition at line 78 of file ListMemBuf.h.

Public Member Functions

- [entry_t\(\)](#)
constructor

Public Attributes

- [T data](#)
The data being stored, not actually an instantiation of T, but big enough to hold it.
- [index_t next](#)
The next element in the used or free chain.
- [index_t prev](#)
The previous element in the used chain, invalid if in the free chain.

7.92.2 Constructor & Destructor Documentation

7.92.2.1 **template<class T_t, unsigned int MAX, class idx_t = unsigned short>**
[ListMemBuf< T_t, MAX, idx_t >::entry_t::entry_t\(\)](#) [[inline](#)]

constructor

Definition at line 80 of file ListMemBuf.h.

References [ListMemBuf< T_t, MAX, idx_t >::entry_t::data](#).

7.92.3 Member Data Documentation

7.92.3.1 `template<class T_t, unsigned int MAX, class idx_t = unsigned short> T ListMemBuf< T_t, MAX, idx_t >::entry_t::data`

The data being stored, not actually an instantiation of T, but big enough to hold it.

Definition at line 81 of file ListMemBuf.h.

7.92.3.2 `template<class T_t, unsigned int MAX, class idx_t = unsigned short> index_t ListMemBuf< T_t, MAX, idx_t >::entry_t::next`

The next element in the used or free chain.

Definition at line 82 of file ListMemBuf.h.

7.92.3.3 `template<class T_t, unsigned int MAX, class idx_t = unsigned short> index_t ListMemBuf< T_t, MAX, idx_t >::entry_t::prev`

The previous element in the used chain, invalid if in the free chain.

Definition at line 83 of file ListMemBuf.h.

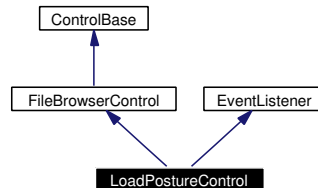
The documentation for this struct was generated from the following file:

- [ListMemBuf.h](#)

7.93 LoadPostureControl Class Reference

```
#include <LoadPostureControl.h>
```

Inheritance diagram for LoadPostureControl:



7.93.1 Detailed Description

Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...).

Should switch this to use a [MotionSequence](#) so it can move more leisurely and not "snap" to position

Definition at line 13 of file LoadPostureControl.h.

Public Member Functions

- [LoadPostureControl](#) (const std::string &n, [MotionManager::MC_ID](#) estop_id)
Constructor.
- virtual [~LoadPostureControl](#) ()
Destructor.
- virtual void [processEvent](#) (const [EventBase](#) &event)
this is to help reduce the twitch at the end (estop tries to go back to its position when this is removed)
- virtual void [deactivate](#) ()
called when this control is being popped from the control stack

Protected Member Functions

- virtual [ControlBase](#) * [selectedFile](#) (const std::string &f)

does the actual loading of the [MotionSequence](#)

Protected Attributes

- [MotionManager::MC_ID](#) `estopid`

MC_ID of the e-stop.

- `std::string` [file](#)

last posture file loaded

7.93.2 Constructor & Destructor Documentation

7.93.2.1 `LoadPostureControl::LoadPostureControl (const std::string & n, MotionManager::MC_ID estop_id)` [inline]

Constructor.

Definition at line 16 of file `LoadPostureControl.h`.

References `config`, `estopid`, `file`, `FileBrowserControl::root`, and `FileBrowserControl::setFilter()`.

7.93.2.2 `virtual LoadPostureControl::~~LoadPostureControl ()` [inline, virtual]

Destructor.

Definition at line 22 of file `LoadPostureControl.h`.

7.93.3 Member Function Documentation

7.93.3.1 `virtual void LoadPostureControl::deactivate ()` [inline, virtual]

called when this control is being popped from the control stack

Reimplemented from [ControlBase](#).

Definition at line 37 of file `LoadPostureControl.h`.

References `erouter`, and `EventRouter::forgetListener()`.

7.93.3.2 **virtual void LoadPostureControl::processEvent (const [EventBase](#) & event)** [inline, virtual]

this is to help reduce the twitch at the end (estop tries to go back to its position when this is removed)

Implements [EventListener](#).

Definition at line 25 of file LoadPostureControl.h.

References [erouter](#), [estopid](#), [file](#), and [EventRouter::removeListener\(\)](#).

7.93.3.3 **virtual [ControlBase](#)* LoadPostureControl::selectedFile (const [std::string](#) & f)** [inline, protected, virtual]

does the actual loading of the [MotionSequence](#)

Reimplemented from [FileBrowserControl](#).

Definition at line 43 of file LoadPostureControl.h.

References [EventRouter::addListener\(\)](#), [MotionManager::addMotion\(\)](#), [EventBase::deactivateETID](#), [erouter](#), [estopid](#), [file](#), [MotionManager::kEmergencyPriority](#), [MotionManager::MC_ID](#), [motman](#), and [EventBase::motmanEGID](#).

7.93.4 Member Data Documentation

7.93.4.1 **[MotionManager::MC_ID](#) LoadPostureControl::estopid** [protected]

MC_ID of the e-stop.

Definition at line 53 of file LoadPostureControl.h.

7.93.4.2 **[std::string](#) LoadPostureControl::file** [protected]

last posture file loaded

Definition at line 54 of file LoadPostureControl.h.

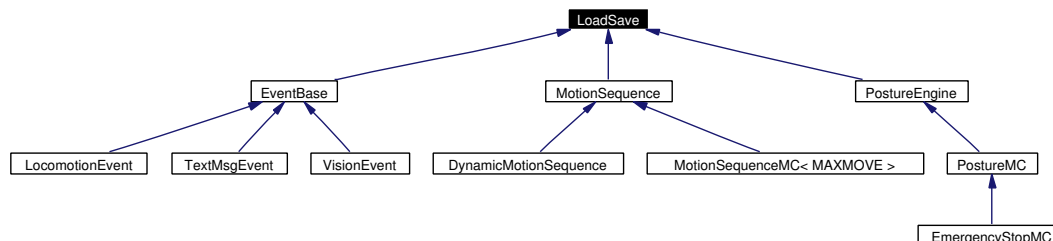
The documentation for this class was generated from the following file:

- [LoadPostureControl.h](#)

7.94 LoadSave Class Reference

```
#include <LoadSave.h>
```

Inheritance diagram for LoadSave:



7.94.1 Detailed Description

Intended as an interface to allow easy and uniform file operations.

Be mindful of version differences - better safe than sorry - put a version number as the first field, just in case

Definition at line 10 of file LoadSave.h.

Public Member Functions

Constructors/Destructors

- [LoadSave](#) ()
constructor
- [LoadSave](#) (const char *filename)
constructor
- virtual [~LoadSave](#) ()
destructor

Buffer Operations

These are useful for sending the data across a network as well as to a file.

These are the only ones that MUST be overridden, as the file ops can be based on calling these, tho feel free to override the file ops as well if speed or temp. memory is tight.

- virtual unsigned int [getBinSize](#) () const=0
calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.
- virtual unsigned int [LoadBuffer](#) (const char buf[], unsigned int len)=0
Load from a saved buffer.
- virtual unsigned int [SaveBuffer](#) (char buf[], unsigned int len) const=0
Save to a given buffer.

File Operations

These are called to load and save to files

- virtual unsigned int [LoadFile](#) (const char *filename)
initiate opening of the specified file and loading/saving of all appropriate information.
- virtual unsigned int [SaveFile](#) (const char *filename) const
initiate opening of the specified file and loading/saving of all appropriate information.
- virtual unsigned int [LoadFile](#) (FILE *f)
Used recursively on member objects once a file is already open - DON'T CLOSE the file in your overridden functions.
- virtual unsigned int [SaveFile](#) (FILE *f) const
Used recursively on member objects once a file is already open - DON'T CLOSE the file in your overridden functions.

Creator Utilities

These are for putting creator codes at the beginning of your data to check for sanity, just optional

- virtual unsigned int [creatorSize](#) (const char creator[]) const
Returns size of the creator code.
- virtual unsigned int [checkCreator](#) (const char *creator, const char buf[], unsigned int len, bool isLoading=true) const
Compares the creator code in the buffer to the one given.
- virtual unsigned int [checkCreator](#) (const char *creator, FILE *f, bool isLoading=true) const
Compares the creator code in the file to the one given, will attempt to reset the file position if fails (so you can check for one of several types).

- virtual unsigned int [saveCreator](#) (const char *creator, char buf[], unsigned int len) const
Saves a creator code to a buffer.
- virtual unsigned int [saveCreator](#) (const char *creator, FILE *f) const
Saves a creator code directly to a file.

Static Public Member Functions

Encode/Decode Utils

encode/decode cross-platform compatible (byte order consistency)

- unsigned int [encode](#) (const [LoadSave](#) &x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) ([LoadSave](#) &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const [LoadSave](#) &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) ([LoadSave](#) &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const double x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (double &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const double x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) (double &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const float x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (float &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const float x, FILE *f)
encode or decode with byte order consistency

- unsigned int [decode](#) (float &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const long x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (long &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const long x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) (long &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const unsigned long x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (unsigned long &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const unsigned long x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) (unsigned long &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const int x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (int &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const int x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) (int &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const unsigned int x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (unsigned int &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const unsigned int x, FILE *f)

encode or decode with byte order consistency

- unsigned int [decode](#) (unsigned int &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const short x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (short &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const short x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) (short &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const unsigned short x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (unsigned short &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const unsigned short x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) (unsigned short &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const std::string &x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (std::string &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const std::string &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) (std::string &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const char *x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (char *&x, const char buf[], unsigned int cap)
encode or decode with byte order consistency

- unsigned int [encode](#) (const char *x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) (char *&x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const char x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (char &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const char x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) (char &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const unsigned char x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (unsigned char &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const unsigned char x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) (unsigned char &x, FILE *f)
encode or decode with byte order consistency
- unsigned int [encode](#) (const bool x, char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [decode](#) (bool &x, const char buf[], unsigned int cap)
encode or decode with byte order consistency
- unsigned int [encode](#) (const bool x, FILE *f)
encode or decode with byte order consistency
- unsigned int [decode](#) (bool &x, FILE *f)
encode or decode with byte order consistency

Static Public Attributes

- const unsigned int [stringpad](#) = sizeof(unsigned int)+1

This is the amount of extra space needed to store a string (int for len of string plus 1 for null term.

Static Protected Member Functions

- `template<class T> void byteswap (T &dstc, const T &srcc)`

templated function to swap byte ordering, should allow compiler to unroll the loop don't use this if src==dst!!!

7.94.2 Constructor & Destructor Documentation

7.94.2.1 LoadSave::LoadSave () [inline]

constructor

Definition at line 17 of file LoadSave.h.

7.94.2.2 LoadSave::LoadSave (const char *filename) [inline]

constructor

Definition at line 18 of file LoadSave.h.

References LoadFile().

7.94.2.3 LoadSave::~LoadSave () [virtual]

destructor

Definition at line 5 of file LoadSave.cc.

7.94.3 Member Function Documentation

7.94.3.1 `template<class T> void LoadSave::byteswap (T &dstc, const T &srcc) [inline, static, protected]`

templated function to swap byte ordering, should allow compiler to unroll the loop don't use this if src==dst!!!

Definition at line 217 of file LoadSave.h.

7.94.3.2 unsigned int LoadSave::checkCreator (const char * *creator*, FILE * *f*, bool *isLoading* = true) const [virtual]

Compares the creator code in the file to the one given, will attempt to reset the file position if fails (so you can check for one of several types).

Parameters:

creator what the creator should be

f the file pointer to check

isLoading set this to true if you want to output a warning if it doesn't match

Returns:

the number of bytes consumed by the creator code, or 0 if it didn't match

Definition at line 20 of file LoadSave.cc.

References decode().

7.94.3.3 unsigned int LoadSave::checkCreator (const char * *creator*, const char *buf*[], unsigned int *len*, bool *isLoading* = true) const [virtual]

Compares the creator code in the buffer to the one given.

Parameters:

creator what the creator should be

buf the buffer to check

len the size remaining in the buffer

isLoading set this to true if you want to output a warning if it doesn't match

Returns:

the number of bytes used by the creator, or 0 if it didn't match

Definition at line 7 of file LoadSave.cc.

References decode().

7.94.3.4 virtual unsigned int LoadSave::creatorSize (const char *creator*[]) const [inline, virtual]

Returns size of the creator code.

Parameters:

creator a string to use for the creator

Returns:

the size to leave for the creator code

Definition at line 72 of file LoadSave.h.

References stringpad.

7.94.3.5 `unsigned int LoadSave::decode (bool & x, FILE *f) [inline, static]`

encode or decode with byte order consistency

Definition at line 213 of file LoadSave.h.

7.94.3.6 `unsigned int LoadSave::decode (bool & x, const char buf[], unsigned int cap) [inline, static]`

encode or decode with byte order consistency

Definition at line 211 of file LoadSave.h.

7.94.3.7 `unsigned int LoadSave::decode (unsigned char & x, FILE *f) [inline, static]`

encode or decode with byte order consistency

Definition at line 208 of file LoadSave.h.

7.94.3.8 `unsigned int LoadSave::decode (unsigned char & x, const char buf[], unsigned int cap) [inline, static]`

encode or decode with byte order consistency

Definition at line 206 of file LoadSave.h.

7.94.3.9 `unsigned int LoadSave::decode (char & x, FILE *f) [inline, static]`

encode or decode with byte order consistency

Definition at line 204 of file LoadSave.h.

7.94.3.10 unsigned int LoadSave::decode (char & *x*, const char *buf*[], unsigned int *cap*) [inline, static]

encode or decode with byte order consistency

Definition at line 202 of file LoadSave.h.

7.94.3.11 unsigned int LoadSave::decode (char *& *x*, FILE **f*) [inline, static]

encode or decode with byte order consistency

Definition at line 199 of file LoadSave.h.

References decode(), and stringpad.

7.94.3.12 unsigned int LoadSave::decode (char *& *x*, const char *buf*[], unsigned int *cap*) [inline, static]

encode or decode with byte order consistency

Definition at line 197 of file LoadSave.h.

References decode(), and stringpad.

7.94.3.13 unsigned int LoadSave::decode (std::string & *x*, FILE **f*) [inline, static]

encode or decode with byte order consistency

Definition at line 194 of file LoadSave.h.

References decode(), and stringpad.

7.94.3.14 unsigned int LoadSave::decode (std::string & *x*, const char *buf*[], unsigned int *cap*) [inline, static]

encode or decode with byte order consistency

Definition at line 192 of file LoadSave.h.

References decode(), and stringpad.

7.94.3.15 unsigned int LoadSave::decode (unsigned short & *x*, FILE **f*) [inline, static]

encode or decode with byte order consistency

Definition at line 188 of file LoadSave.h.

References `byteswap()`.

7.94.3.16 `unsigned int LoadSave::decode (unsigned short & x, const char buf[], unsigned int cap)` `[inline, static]`

encode or decode with byte order consistency

Definition at line 186 of file LoadSave.h.

References `byteswap()`.

7.94.3.17 `unsigned int LoadSave::decode (short & x, FILE *f)` `[inline, static]`

encode or decode with byte order consistency

Definition at line 184 of file LoadSave.h.

References `byteswap()`.

7.94.3.18 `unsigned int LoadSave::decode (short & x, const char buf[], unsigned int cap)` `[inline, static]`

encode or decode with byte order consistency

Definition at line 182 of file LoadSave.h.

References `byteswap()`.

7.94.3.19 `unsigned int LoadSave::decode (unsigned int & x, FILE *f)` `[inline, static]`

encode or decode with byte order consistency

Definition at line 179 of file LoadSave.h.

References `byteswap()`.

7.94.3.20 `unsigned int LoadSave::decode (unsigned int & x, const char buf[], unsigned int cap)` `[inline, static]`

encode or decode with byte order consistency

Definition at line 177 of file LoadSave.h.

References `byteswap()`.

7.94.3.21 `unsigned int LoadSave::decode (int & x, FILE * f) [inline, static]`

encode or decode with byte order consistency

Definition at line 175 of file LoadSave.h.

References `byteswap()`.

7.94.3.22 `unsigned int LoadSave::decode (int & x, const char buf[], unsigned int cap) [inline, static]`

encode or decode with byte order consistency

Definition at line 173 of file LoadSave.h.

References `byteswap()`.

7.94.3.23 `unsigned int LoadSave::decode (unsigned long & x, FILE * f) [inline, static]`

encode or decode with byte order consistency

Definition at line 171 of file LoadSave.h.

References `byteswap()`.

7.94.3.24 `unsigned int LoadSave::decode (unsigned long & x, const char buf[], unsigned int cap) [inline, static]`

encode or decode with byte order consistency

Definition at line 169 of file LoadSave.h.

References `byteswap()`.

7.94.3.25 `unsigned int LoadSave::decode (long & x, FILE * f) [inline, static]`

encode or decode with byte order consistency

Definition at line 167 of file LoadSave.h.

References `byteswap()`.

7.94.3.26 `unsigned int LoadSave::decode (long & x, const char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 165 of file LoadSave.h.

References `byteswap()`.

7.94.3.27 `unsigned int LoadSave::decode (float & x, FILE *f)` [inline, static]

encode or decode with byte order consistency

Definition at line 162 of file LoadSave.h.

References `byteswap()`.

7.94.3.28 `unsigned int LoadSave::decode (float & x, const char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 160 of file LoadSave.h.

References `byteswap()`.

7.94.3.29 `unsigned int LoadSave::decode (double & x, FILE *f)` [inline, static]

encode or decode with byte order consistency

Definition at line 157 of file LoadSave.h.

References `byteswap()`.

7.94.3.30 `unsigned int LoadSave::decode (double & x, const char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 155 of file LoadSave.h.

References `byteswap()`.

7.94.3.31 `unsigned int LoadSave::decode (LoadSave & x, FILE * f)`
[inline, static]

encode or decode with byte order consistency

Definition at line 112 of file LoadSave.h.

References LoadFile().

7.94.3.32 `unsigned int LoadSave::decode (LoadSave & x, const char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 110 of file LoadSave.h.

References LoadBuffer().

7.94.3.33 `unsigned int LoadSave::encode (const bool x, FILE * f)` [inline, static]

encode or decode with byte order consistency

Definition at line 212 of file LoadSave.h.

7.94.3.34 `unsigned int LoadSave::encode (const bool x, char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 210 of file LoadSave.h.

7.94.3.35 `unsigned int LoadSave::encode (const unsigned char x, FILE * f)`
[inline, static]

encode or decode with byte order consistency

Definition at line 207 of file LoadSave.h.

7.94.3.36 `unsigned int LoadSave::encode (const unsigned char x, char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 205 of file LoadSave.h.

7.94.3.37 `unsigned int LoadSave::encode (const char x, FILE *f)` [`inline`, `static`]

encode or decode with byte order consistency

Definition at line 203 of file LoadSave.h.

7.94.3.38 `unsigned int LoadSave::encode (const char x, char buf[], unsigned int cap)` [`inline`, `static`]

encode or decode with byte order consistency

Definition at line 201 of file LoadSave.h.

7.94.3.39 `unsigned int LoadSave::encode (const char * x, FILE *f)`
[`inline`, `static`]

encode or decode with byte order consistency

Definition at line 198 of file LoadSave.h.

References `encode()`.

7.94.3.40 `unsigned int LoadSave::encode (const char * x, char buf[], unsigned int cap)` [`inline`, `static`]

encode or decode with byte order consistency

Definition at line 196 of file LoadSave.h.

References `encode()`, and `stringpad`.

7.94.3.41 `unsigned int LoadSave::encode (const std::string & x, FILE *f)`
[`inline`, `static`]

encode or decode with byte order consistency

Definition at line 193 of file LoadSave.h.

References `encode()`.

7.94.3.42 `unsigned int LoadSave::encode (const std::string & x, char buf[], unsigned int cap)` [`inline`, `static`]

encode or decode with byte order consistency

Definition at line 191 of file LoadSave.h.

References encode(), and stringpad.

7.94.3.43 `unsigned int LoadSave::encode (const unsigned short x, FILE *f)`
[inline, static]

encode or decode with byte order consistency

Definition at line 187 of file LoadSave.h.

References byteswap().

7.94.3.44 `unsigned int LoadSave::encode (const unsigned short x, char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 185 of file LoadSave.h.

References byteswap().

7.94.3.45 `unsigned int LoadSave::encode (const short x, FILE *f)` [inline, static]

encode or decode with byte order consistency

Definition at line 183 of file LoadSave.h.

References byteswap().

7.94.3.46 `unsigned int LoadSave::encode (const short x, char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 181 of file LoadSave.h.

References byteswap().

7.94.3.47 `unsigned int LoadSave::encode (const unsigned int x, FILE *f)`
[inline, static]

encode or decode with byte order consistency

Definition at line 178 of file LoadSave.h.

References byteswap().

7.94.3.48 `unsigned int LoadSave::encode (const unsigned int x, char buf[], unsigned int cap) [inline, static]`

encode or decode with byte order consistency

Definition at line 176 of file LoadSave.h.

References `byteswap()`.

7.94.3.49 `unsigned int LoadSave::encode (const int x, FILE *f) [inline, static]`

encode or decode with byte order consistency

Definition at line 174 of file LoadSave.h.

References `byteswap()`.

7.94.3.50 `unsigned int LoadSave::encode (const int x, char buf[], unsigned int cap) [inline, static]`

encode or decode with byte order consistency

Definition at line 172 of file LoadSave.h.

References `byteswap()`.

7.94.3.51 `unsigned int LoadSave::encode (const unsigned long x, FILE *f) [inline, static]`

encode or decode with byte order consistency

Definition at line 170 of file LoadSave.h.

References `byteswap()`.

7.94.3.52 `unsigned int LoadSave::encode (const unsigned long x, char buf[], unsigned int cap) [inline, static]`

encode or decode with byte order consistency

Definition at line 168 of file LoadSave.h.

References `byteswap()`.

7.94.3.53 `unsigned int LoadSave::encode (const long x, FILE *f)` [inline, static]

encode or decode with byte order consistency

Definition at line 166 of file LoadSave.h.

References `byteswap()`.

7.94.3.54 `unsigned int LoadSave::encode (const long x, char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 164 of file LoadSave.h.

References `byteswap()`.

7.94.3.55 `unsigned int LoadSave::encode (const float x, FILE *f)` [inline, static]

encode or decode with byte order consistency

Definition at line 161 of file LoadSave.h.

References `byteswap()`.

7.94.3.56 `unsigned int LoadSave::encode (const float x, char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 159 of file LoadSave.h.

References `byteswap()`.

7.94.3.57 `unsigned int LoadSave::encode (const double x, FILE *f)` [inline, static]

encode or decode with byte order consistency

Definition at line 156 of file LoadSave.h.

References `byteswap()`.

7.94.3.58 `unsigned int LoadSave::encode (const double x, char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 154 of file LoadSave.h.

References `byteswap()`.

7.94.3.59 `unsigned int LoadSave::encode (const LoadSave & x, FILE *f)` [inline, static]

encode or decode with byte order consistency

Definition at line 111 of file LoadSave.h.

References `SaveFile()`.

7.94.3.60 `unsigned int LoadSave::encode (const LoadSave & x, char buf[], unsigned int cap)` [inline, static]

encode or decode with byte order consistency

Definition at line 109 of file LoadSave.h.

References `SaveBuffer()`.

7.94.3.61 `virtual unsigned int LoadSave::getBinSize () const` [pure virtual]

calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.

Returns:

number of bytes read/written, 0 if error (or empty)

Implemented in [EventBase](#), [LocomotionEvent](#), [TextMsgEvent](#), [VisionEvent](#), [MotionSequence](#), and [PostureEngine](#).

7.94.3.62 `virtual unsigned int LoadSave::LoadBuffer (const char buf[], unsigned int len)` [pure virtual]

Load from a saved buffer.

Parameters:

buf pointer to the memory where you should begin loading

len length of *buf* available (this isn't all yours, might be more stuff saved after yours)

Returns:

the number of bytes actually used

Implemented in [EventBase](#), [LocomotionEvent](#), [TextMsgEvent](#), [VisionEvent](#), [MotionSequence](#), [PostureEngine](#), and [PostureMC](#).

7.94.3.63 unsigned int LoadSave::LoadFile (FILE **f*) [virtual]

Used recursively on member objects once a file is already open - DON'T CLOSE the file in your overridden functions.

Parameters:

f a pointer to the file to load

Warning:

could potentially be very inefficient if root-level objects override LoadFile but leaf-level ones use this implementation, but leaf-level ones won't even get this call unless you override the ones above them - hence, this is all or nothing

Returns:

number of bytes read, 0 if error (or empty)

Definition at line 84 of file LoadSave.cc.

References LoadBuffer().

7.94.3.64 unsigned int LoadSave::LoadFile (const char **filename*) [virtual]

initiate opening of the specified file and loading/saving of all appropriate information.

Parameters:

filename the file to load/save

Returns:

number of bytes read/written, 0 if error (or empty)

Definition at line 47 of file LoadSave.cc.

7.94.3.65 `virtual unsigned int LoadSave::SaveBuffer (char buf [], unsigned int len) const` [pure virtual]

Save to a given buffer.

Parameters:

buf pointer to the memory where you should begin writing

len length of *buf* available. (this isn't all yours, constrain yourself to what you returned in [getBinSize\(\)](#))

Returns:

the number of bytes actually used

Implemented in [EventBase](#), [LocomotionEvent](#), [TextMsgEvent](#), [VisionEvent](#), [MotionSequence](#), and [PostureEngine](#).

7.94.3.66 `unsigned int LoadSave::saveCreator (const char * creator, FILE * f) const` [virtual]

Saves a creator code directly to a file.

Parameters:

creator the string to use for the creator code

f the file to save the code into

Returns:

the number of bytes consumed

Definition at line 43 of file LoadSave.cc.

References [encode\(\)](#).

7.94.3.67 `unsigned int LoadSave::saveCreator (const char * creator, char buf [], unsigned int len) const` [virtual]

Saves a creator code to a buffer.

Parameters:

creator the string to use for the creator code

buf the buffer to save the code into

len the space available in the buffer

Returns:

the number of bytes consumed

Definition at line 39 of file LoadSave.cc.

References encode().

7.94.3.68 unsigned int LoadSave::SaveFile (FILE **f*) const [virtual]

Used recursively on member objects once a file is already open - DON'T CLOSE the file in your overridden functions.

Parameters:

f a pointer to the file to save

Warning:

could potentially be very inefficient if root-level objects override SaveFile but leaf-level ones use this implementation, but leaf-level ones won't even get this call unless you override the ones above them - hence, this is all or nothing

Returns:

number of bytes written, 0 if error (or empty)

Definition at line 114 of file LoadSave.cc.

References getBinSize(), and SaveBuffer().

7.94.3.69 unsigned int LoadSave::SaveFile (const char **filename*) const [virtual]

initiate opening of the specified file and loading/saving of all appropriate information.

Parameters:

filename the file to load/save

Returns:

number of bytes read/written, 0 if error (or empty)

Definition at line 65 of file LoadSave.cc.

7.94.4 Member Data Documentation

7.94.4.1 const unsigned int LoadSave::stringpad = sizeof(unsigned int)+1 [static]

This is the amount of extra space needed to store a string (int for len of string plus 1 for null term).

Definition at line 13 of file LoadSave.h.

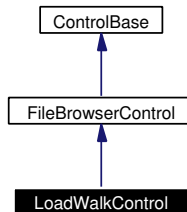
The documentation for this class was generated from the following files:

- [LoadSave.h](#)
- [LoadSave.cc](#)

7.95 LoadWalkControl Class Reference

```
#include <LoadWalkControl.h>
```

Inheritance diagram for LoadWalkControl:



7.95.1 Detailed Description

When activated, loads a set of walk parameters from a file specified by user.

Definition at line 10 of file LoadWalkControl.h.

Public Member Functions

- [LoadWalkControl](#) (const std::string &n, [MotionManager::MC_ID](#) w)
constructor, pass the MC_ID of the [WalkMC](#) which you want to save
- virtual [~LoadWalkControl](#) ()
destructor

Protected Member Functions

- virtual [ControlBase](#) * [selectedFile](#) (const std::string &f)
does the actual loading of the [MotionSequence](#)

Protected Attributes

- [MotionManager::MC_ID](#) [walk_id](#)
the MC_ID of the walk to load into

7.95.2 Constructor & Destructor Documentation

7.95.2.1 `LoadWalkControl::LoadWalkControl (const std::string & n, MotionManager::MC_ID w)` [inline]

constructor, pass the MC_ID of the [WalkMC](#) which you want to save

Definition at line 13 of file `LoadWalkControl.h`.

References `config`, `FileBrowserControl::root`, `FileBrowserControl::setFilter()`, and `walk_id`.

7.95.2.2 `virtual LoadWalkControl::~LoadWalkControl ()` [inline, virtual]

destructor

Definition at line 19 of file `LoadWalkControl.h`.

7.95.3 Member Function Documentation

7.95.3.1 `virtual ControlBase* LoadWalkControl::selectedFile (const std::string & f)` [inline, protected, virtual]

does the actual loading of the [MotionSequence](#)

Reimplemented from [FileBrowserControl](#).

Definition at line 23 of file `LoadWalkControl.h`.

References `MotionManager::checkinMotion()`, `MotionManager::checkoutMotion()`, `WalkMC::load()`, `motman`, and `walk_id`.

7.95.4 Member Data Documentation

7.95.4.1 `MotionManager::MC_ID LoadWalkControl::walk_id` [protected]

the MC.ID of the walk to load into

Definition at line 30 of file `LoadWalkControl.h`.

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

- [LoadWalkControl.h](#)

7.96 LockScope< num_doors > Class Template Reference

```
#include <LockScope.h>
```

7.96.1 Detailed Description

`template<unsigned int num_doors> class LockScope< num_doors >`

Locks a [MutexLock](#) until the LockScope goes out of scope.

This can help prevent forgetting to do it if you function has multiple return points

Definition at line 10 of file LockScope.h.

Public Member Functions

- [LockScope](#) ([MutexLock](#)< num_doors > &lock, int id)
constructor, locks lock with id
- [~LockScope](#) ()
destructor, releases lock received in constructor

Protected Attributes

- [MutexLock](#)< num_doors > &l
the lock

7.96.2 Constructor & Destructor Documentation

7.96.2.1 `template<unsigned int num_doors> LockScope< num_doors >::LockScope (MutexLock< num_doors > &lock, int id) [inline]`

constructor, locks *lock* with *id*

Definition at line 13 of file LockScope.h.

References `LockScope< num_doors >::l`.

7.96.2.2 `template<unsigned int num_doors> LockScope< num_doors
>::~~LockScope () [inline]`

destructor, releases lock received in constructor

Definition at line 15 of file `LockScope.h`.

References `LockScope< num_doors >::l`.

7.96.3 Member Data Documentation

7.96.3.1 `template<unsigned int num_doors> MutexLock<num_doors>&
LockScope< num_doors >::l [protected]`

the lock

Definition at line 17 of file `LockScope.h`.

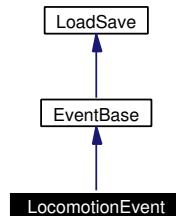
The documentation for this class was generated from the following file:

- [LockScope.h](#)

7.97 LocomotionEvent Class Reference

```
#include <LocomotionEvent.h>
```

Inheritance diagram for LocomotionEvent:



7.97.1 Detailed Description

Gives updates regarding the current movement of the robot through the world.

An activate event will be sent when a potential source of motion is created, and a deactivate when it is destroyed. Status events will be sent at any change of direction/speed.

The source ID field will hold the [MotionManager::MC_ID](#) of the sending [MotionCommand](#)

Definition at line 13 of file LocomotionEvent.h.

Public Member Functions

- [LocomotionEvent](#) & [setXYA](#) (float X, float Y, float A)
Allows you to set the new X, Y, and A components.
- virtual unsigned int [getBinSize](#) () const
calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.
- virtual unsigned int [LoadBuffer](#) (const char buf[], unsigned int len)
Load from a saved buffer.
- virtual unsigned int [SaveBuffer](#) (char buf[], unsigned int len) const
Save to a given buffer.

Constructors

- [LocomotionEvent \(\)](#)
constructor
- [LocomotionEvent \(EventGeneratorID.t gid, unsigned int sid, EventTypeID.t tid, unsigned int dur=0\)](#)
constructor
- [LocomotionEvent \(EventGeneratorID.t gid, unsigned int sid, EventTypeID.t tid, unsigned int dur, const std::string &n, float mag\)](#)
constructor

Public Attributes

- float [x](#)
the new x component (body relative)
- float [y](#)
the new y component (body relative)
- float [a](#)
the new angular component (body relative)

7.97.2 Constructor & Destructor Documentation

7.97.2.1 LocomotionEvent::LocomotionEvent () [inline]

constructor

Definition at line 19 of file LocomotionEvent.h.

References [a](#), [x](#), and [y](#).

7.97.2.2 LocomotionEvent::LocomotionEvent ([EventGeneratorID.t gid](#), unsigned int *sid*, [EventTypeID.t tid](#), unsigned int *dur* = 0) [inline]

constructor

Definition at line 20 of file LocomotionEvent.h.

References [a](#), [x](#), and [y](#).

7.97.2.3 LocomotionEvent::LocomotionEvent ([EventGeneratorID_t](#) *gid*, unsigned int *sid*, [EventTypeID_t](#) *tid*, unsigned int *dur*, const std::string & *n*, float *mag*) [inline]

constructor

Definition at line 21 of file LocomotionEvent.h.

References [a](#), [x](#), and [y](#).

7.97.3 Member Function Documentation

7.97.3.1 virtual unsigned int LocomotionEvent::getBinSize () const
[inline, virtual]

calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.

Returns:

number of bytes read/written, 0 if error (or empty)

Reimplemented from [EventBase](#).

Definition at line 32 of file LocomotionEvent.h.

References [a](#), [LoadSave::creatorSize\(\)](#), [EventBase::getBinSize\(\)](#), [x](#), and [y](#).

7.97.3.2 virtual unsigned int LocomotionEvent::LoadBuffer (const char *buf*[], unsigned int *len*) [inline, virtual]

Load from a saved buffer.

Parameters:

buf pointer to the memory where you should begin loading

len length of *buf* available (this isn't all yours, might be more stuff saved after yours)

Returns:

the number of bytes actually used

Reimplemented from [EventBase](#).

Definition at line 41 of file LocomotionEvent.h.

References [a](#), [LoadSave::checkCreator\(\)](#), [LoadSave::decode\(\)](#), [EventBase::LoadBuffer\(\)](#), [x](#), and [y](#).

7.97.3.3 **virtual unsigned int LocomotionEvent::SaveBuffer** (char *buf*[], unsigned int *len*) const [inline, virtual]

Save to a given buffer.

Parameters:

buf pointer to the memory where you should begin writing

len length of *buf* available. (this isn't all yours, constrain yourself to what you returned in [getBinSize\(\)](#))

Returns:

the number of bytes actually used

Reimplemented from [EventBase](#).

Definition at line 57 of file LocomotionEvent.h.

References [a](#), [LoadSave::encode\(\)](#), [EventBase::SaveBuffer\(\)](#), [LoadSave::save-Creator\(\)](#), [x](#), and [y](#).

7.97.3.4 **[LocomotionEvent](#)& LocomotionEvent::setXYA** (float *X*, float *Y*, float *A*) [inline]

Allows you to set the new X, Y, and A components.

Definition at line 25 of file LocomotionEvent.h.

References [a](#), [x](#), and [y](#).

7.97.4 Member Data Documentation

7.97.4.1 **float [LocomotionEvent::a](#)**

the new angular component (body relative)

Definition at line 75 of file LocomotionEvent.h.

7.97.4.2 **float [LocomotionEvent::x](#)**

the new x component (body relative)

Definition at line 73 of file LocomotionEvent.h.

7.97.4.3 **float [LocomotionEvent::y](#)**

the new y component (body relative)

Definition at line 74 of file LocomotionEvent.h.

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

- [LocomotionEvent.h](#)

7.98 Marker Struct Reference

```
#include <Vision.h>
```

Public Attributes

- [region](#) * [regs](#) [3]
- int [cen_x](#)
- int [cen_y](#)
- int [marker](#)

7.98.1 Member Data Documentation

7.98.1.1 int [Marker::cen_x](#)

Definition at line 81 of file Vision.h.

7.98.1.2 int [Marker::cen_y](#)

Definition at line 82 of file Vision.h.

7.98.1.3 int [Marker::marker](#)

Definition at line 83 of file Vision.h.

7.98.1.4 [region](#)* [Marker::regs](#)[3]

Definition at line 80 of file Vision.h.

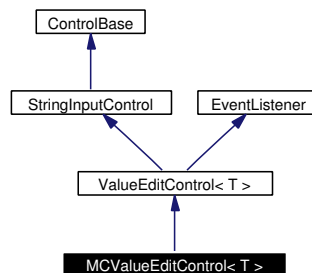
The documentation for this struct was generated from the following file:

- [Vision.h](#)

7.99 MCValueEditControl< T > Class Template Reference

```
#include <MCValueEditControl.h>
```

Inheritance diagram for MCValueEditControl< T >:



7.99.1 Detailed Description

template<class T> class MCValueEditControl< T >

allows you to modify a value in memory, much like [ValueEditControl](#), but will check out a [MotionCommand](#) first to maintain proper mutual exclusion.

Definition at line 8 of file `MCValueEditControl.h`.

Public Member Functions

- [MCValueEditControl](#) (const std::string &n, T *t, [MotionManager::MC_ID](#) id)
constructor
- virtual [~MCValueEditControl](#) ()
destructor
- virtual [ControlBase](#) * [doSelect](#) ()
if the value of the [target](#)!=[cur](#), assigns the current value to the target and all the [copies](#)

Protected Attributes

- [MotionManager::MC_ID](#) mcid

the id of the [MotionCommand](#) which should be checked out

7.99.2 Constructor & Destructor Documentation

7.99.2.1 `template<class T> MCValueEditControl< T >::MCValueEditControl(const std::string & n, T * t, MotionManager::MC_ID id) [inline]`

constructor

Definition at line 11 of file [MCValueEditControl.h](#).

References [MCValueEditControl](#)< T >::mcid.

7.99.2.2 `template<class T> virtual MCValueEditControl< T >::~~MCValueEditControl() [inline, virtual]`

destructor

Definition at line 14 of file [MCValueEditControl.h](#).

7.99.3 Member Function Documentation

7.99.3.1 `template<class T> virtual ControlBase* MCValueEditControl< T >::doSelect() [inline, virtual]`

if the value of the [target](#)!=[cur](#), assigns the current value to the target and all the [copies](#)

Reimplemented from [ValueEditControl](#)< T >.

Definition at line 16 of file [MCValueEditControl.h](#).

References [MotionManager::checkinMotion\(\)](#), [MotionManager::checkoutMotion\(\)](#), [ValueEditControl](#)< T >::doSelect(), [MCValueEditControl](#)< T >::mcid, and [motman](#).

7.99.4 Member Data Documentation

7.99.4.1 `template<class T> MotionManager::MC_ID MCValueEditControl< T >::mcid [protected]`

the id of the [MotionCommand](#) which should be checked out

Definition at line 24 of file [MCValueEditControl.h](#).

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

- [MCValueEditControl.h](#)

7.100 MMAccessor< MC_t > Class Template Reference

```
#include <MMAccessor.h>
```

7.100.1 Detailed Description

template<class MC_t> class MMAccessor< MC_t >

This class allows convenient ways of accessing a motion command.

Since MotionCommands must be checked out of the motion manager and then checked back in when they are done, this is a common source of errors, leading to deadlock. This class will check the motion out when it's created, and check it back in when it goes out of scope

It supports recursive checkin/checkouts.

Uses global [motman](#)

So, instead of doing things like this:

```
YourMotionCommand* ymc = dynamic_cast<YourMotionCommand*>(motman->checkoutMotion(your_mc_id));
//do 'stuff' with ymc, e.g.: ymc->rollOver();
motman->checkinMotion(your_mc_id);
```

...which can be error prone in many regards - if 'stuff' returns without checking in, or you forget to check in, or you get lazy and leave it checked out longer than you should, which can cause general performance issues (or worse, deadlock) Using MMAccessor makes it much easier to solve these problems, and is easier to code:

```
MMAccessor<YourMotionCommand> mma(your_mc_id);
//do 'stuff' with mma, e.g.: mma->rollOver();
```

We can call a return at any point and the motion command will automatically be checked in, and since C++ guarantees that the destructor of mma will be called, we don't have to worry about forgetting about it. We can limit the scope by placing {}'s around the segment in question:

```
//pre-stuff
{
    MMAccessor<YourMotionCommand> mma(your_mc_id);
    //do 'stuff' with mma, e.g.: mma->rollOver();
}
//post-stuff - has no knowledge of mma, out of its scope
```

And, for those astute enough to notice that the theoretical *rollOver()* function is called on MMAccessor when it's actually a member of YourMotionCommand, this is because

MMAccessor behaves as a 'smart pointer', which overloads `operator → ()` so it is fairly transparent to use.

See also the templated `checkin(Ret.t ret)` function for more examples of streamlined usage.

MMAccessor is a small class, you may consider passing it around instead of a `MotionManager::MC_ID` if appropriate. (Would be appropriate to avoid multiple checkin/outs in a row from different functions, but not as appropriate for storage and reuse of the same MMAccessor.

Definition at line 49 of file MMAccessor.h.

Public Member Functions

- `MMAccessor (MotionManager::MC_ID id, bool ckout=true)`
constructor, checks out by default
- `MMAccessor (const MMAccessor &a)`
copy constructor - will reference the same mc_id - checkin/checkouts are NOT independent between this and a - they will be linked
- `~MMAccessor ()`
destructor, checks in if needed
- `MMAccessor< MC.t > operator= (const MMAccessor< MC.t > &a)`
allows assignment of MMAccessor's, similar to the copy constructor - the two MMAccessor's will control the same MotionCommand
- `MC.t * checkout ()`
So you can check out if not done by default (or you checked in already).
- `MC.t * mc () const`
Returns the motion command's address so you can call functions.
- `void checkin ()`
Checks in the motion.
- `template<class Ret.t> Ret.t checkin (Ret.t ret)`
Checks in the motion, passing through the value it is passed.
- `MC.t * operator → ()`
smart pointer to the underlying MotionCommand
- `const MC.t * operator → () const`

smart pointer to the underlying [MotionCommand](#)

- MC_t & [operator *](#) ()
smart pointer to the underlying [MotionCommand](#)
- const MC_t & [operator *](#) () const
smart pointer to the underlying [MotionCommand](#)
- MC_t & [operator\[\]](#) (int i)
smart pointer to the underlying [MotionCommand](#)
- const MC_t & [operator\[\]](#) (int i) const
smart pointer to the underlying [MotionCommand](#)

Protected Attributes

- [MotionManager::MC_ID mc_id](#)
the MC_ID that this Accessor was constructed with
- MC_t * [mcptr](#)
a pointer to the motion command, should always be valid even when not checked out so you can access member fields (which is reasonably safe)

7.100.2 Constructor & Destructor Documentation

7.100.2.1 `template<class MC_t> MMAccessor< MC_t >::MMAccessor(MotionManager::MC_ID id, bool ckout = true) [inline]`

constructor, checks out by default

Parameters:

- id* the motion command to check out
- ckout* if true (default) will checkout upon creation. otherwise it just gets current address (so you can peek at member fields, which should be safe)

Definition at line 55 of file MMAccessor.h.

References [MMAccessor< MC_t >::checkout\(\)](#), [MMAccessor< MC_t >::mc_id](#), [MMAccessor< MC_t >::mcptr](#), [motman](#), and [MotionManager::peekMotion\(\)](#).

7.100.2.2 `template<class MC_t> MMAccessor< MC_t >::MMAccessor (const MMAccessor< MC_t > & a) [inline]`

copy constructor - will reference the same mc_id - checkin/checkouts are NOT independent between this and *a* - they will be linked

Definition at line 63 of file MMAccessor.h.

References MMAccessor< MC_t >::mc_id, and MMAccessor< MC_t >::mcptr.

7.100.2.3 `template<class MC_t> MMAccessor< MC_t >::~~MMAccessor () [inline]`

destructor, checks in if needed

Definition at line 66 of file MMAccessor.h.

References MMAccessor< MC_t >::checkin(), MotionManager::checkoutLevel(), MMAccessor< MC_t >::mc_id, and motman.

7.100.3 Member Function Documentation

7.100.3.1 `template<class MC_t> template<class Ret_t> Ret_t MMAccessor< MC_t >::checkin (Ret_t ref) [inline]`

Checks in the motion, passing through the value it is passed.

Returns:

the same value it's passed

Useful in situations like this:

```
MMAccessor<myMC> mine(myMC_id);
if(mine.mc()->foo())
    //do something with motman here
```

But we want to check in *mine* ASAP - if we don't reference it anywhere in the if statement, we're leaving the MC locked longer than we need to. How about instead doing this:

```
bool cond;
{MMAccessor<myMC> mine(myMC_id); cond=mine.mc()->foo();}
if(cond)
    //do something with motman here
```

But that uses an extra variable... ewwww... so use this function as a pass through to checkin the MC:

```

MMAccessor<myMC> mine(myMC_id);
if(mine.checkin(mine.mc()->foo()))
    //do something with motman here

```

Definition at line 117 of file MMAccessor.h.

References MMAccessor< MC_t >::checkin(), and MMAccessor< MC_t >::mcptr.

7.100.3.2 `template<class MC_t> void MMAccessor< MC_t >::checkin ()` [inline]

Checks in the motion.

Don't forget, you can also just limit the scope using extra { }'s

Definition at line 88 of file MMAccessor.h.

References MotionManager::checkinMotion(), MMAccessor< MC_t >::mc_id, and motman.

7.100.3.3 `template<class MC_t> MC_t* MMAccessor< MC_t >::checkout ()` [inline]

So you can check out if not done by default (or you checked in already).

<

Test

can this be a dynamic_cast?

Definition at line 79 of file MMAccessor.h.

References MotionManager::checkoutMotion(), MMAccessor< MC_t >::mc_id, MMAccessor< MC_t >::mcptr, and motman.

7.100.3.4 `template<class MC_t> MC_t* MMAccessor< MC_t >::mc () const` [inline]

Returns the motion command's address so you can call functions.

Definition at line 84 of file MMAccessor.h.

References MMAccessor< MC_t >::mcptr.

7.100.3.5 `template<class MC_t> const MC_t& MMAccessor< MC_t >::operator * () const` [inline]

smart pointer to the underlying [MotionCommand](#)

Definition at line 126 of file MMAccessor.h.

References MMAccessor< MC_t >::mc().

7.100.3.6 `template<class MC_t> MC_t& MMAccessor< MC_t >::operator * ()`
[inline]

smart pointer to the underlying [MotionCommand](#)

Definition at line 125 of file MMAccessor.h.

References MMAccessor< MC_t >::mc().

7.100.3.7 `template<class MC_t> const MC_t* MMAccessor< MC_t >::operator → () const` [inline]

smart pointer to the underlying [MotionCommand](#)

Definition at line 124 of file MMAccessor.h.

References MMAccessor< MC_t >::mc().

7.100.3.8 `template<class MC_t> MC_t* MMAccessor< MC_t >::operator → ()` [inline]

smart pointer to the underlying [MotionCommand](#)

Definition at line 123 of file MMAccessor.h.

References MMAccessor< MC_t >::mc().

7.100.3.9 `template<class MC_t> MMAccessor<MC_t> MMAccessor< MC_t >::operator= (const MMAccessor< MC_t > &a)` [inline]

allows assignment of MMAccessor's, similar to the copy constructor - the two MMAccessor's will control the same [MotionCommand](#)

Definition at line 72 of file MMAccessor.h.

References MMAccessor< MC_t >::mc_id, and MMAccessor< MC_t >::mcptr.

7.100.3.10]

`template<class MC_t> const MC_t& MMAccessor< MC_t >::operator[] (int i) const`
[inline]

smart pointer to the underlying [MotionCommand](#)

Definition at line 128 of file MMAccessor.h.

References MMAccessor< MC_t >::mc().

7.100.3.11]

```
template<class MC_t> MC_t& MMAccessor< MC_t >::operator[] (int i)
[inline]
```

smart pointer to the underlying [MotionCommand](#)

Definition at line 127 of file MMAccessor.h.

References MMAccessor< MC_t >::mc().

7.100.4 Member Data Documentation

7.100.4.1 `template<class MC_t> MotionManager::MC_ID MMAccessor< MC_t >::mc_id` [protected]

the MC_ID that this Accessor was constructed with

Definition at line 131 of file MMAccessor.h.

7.100.4.2 `template<class MC_t> MC_t* MMAccessor< MC_t >::mcptr` [protected]

a pointer to the motion command, should always be valid even when not checked out so you can access member fields (which is reasonably safe)

Definition at line 132 of file MMAccessor.h.

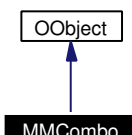
The documentation for this class was generated from the following file:

- [MMAccessor.h](#)

7.101 MMCombo Class Reference

```
#include <MMCombo.h>
```

Inheritance diagram for MMCombo:



7.101.1 Detailed Description

Contains code for both MainObj and MotoObj processes.

Why go to all this trouble? Virtual functions and polymorphism! Instead of writing my own object typing and serialization system, I would rather just use C++'s. But function lookups of the run time type information (RTTI) will break unless the object that created the object and the object that's actually calling the function agree on what object A's information is.

The easiest way to guarantee this is to compile them as one object, and then replace the strings in the source binary with strings for each of the final objects so they'll each have their own identity, but share the same code.

This is as close as I can get to a "fork", which is what i really want.

Definition at line 29 of file MMCombo.h.

Public Member Functions

- [MMCombo](#) ()
constructor
- virtual [~MMCombo](#) ()
destructor
- virtual OStatus [DoInit](#) (const OSystemEvent &)
first call (after constructor), set up memory
- virtual OStatus [DoStart](#) (const OSystemEvent &)
second call, ask for messages

- virtual OStatus [DoStop](#) (const OSystemEvent &)
next to last call, stop sending and receiving messages
- virtual OStatus [DoDestroy](#) (const OSystemEvent &)
last call (before destructor), clean up memory here
- void [ReadyRegisterWorldState](#) (const OReadyEvent &)
main only, send out the state global
- void [GotWorldState](#) (const ONotifyEvent &event)
motion only, called when state global is received
- void [ReadyRegisterMotionManager](#) (const OReadyEvent &)
motion only, send out motman global
- void [GotMotionManager](#) (const ONotifyEvent &event)
main only, called when motman global is received
- void [ReadyRegisterEventTranslatorQueue](#) (const OReadyEvent &)
main only, send out the EventTranslatorQueue
- void [GotEventTranslatorQueue](#) (const ONotifyEvent &event)
motion only, called when EventTranslatorQueue is received
- void [ReadySendJoints](#) (const OReadyEvent &event)
motion only (until main does ears again, then both) calls SendJoints, if DoStart has already been called
- void [GotSensorFrame](#) (const ONotifyEvent &event)
main only, called when new sensor information is available
- void [GotImage](#) (const ONotifyEvent &event)
main only, called when a new image is available
- void [GotPowerEvent](#) (void *msg)
main only, called when a power event occurs (can be just status events)
- void [GotMotionMsg](#) (const ONotifyEvent &event)
both, called when a new [MotionManagerMsg](#) has been received
- void [GotSoundManager](#) (const ONotifyEvent &event)
both, called when the sndman global is received

- void [ListenCont](#) (void *msg)
main only, called when //ALTODD
- void [BindCont](#) (void *msg)
main only, called when //ALTODD
- void [ConnectCont](#) (void *msg)
main only, called when //ALTODD
- void [SendCont](#) (void *msg)
main only, called when //ALTODD
- void [ReceiveCont](#) (void *msg)
main only, called when //ALTODD
- void [CloseCont](#) (void *msg)
main only, called when //ALTODD
- bool [RPOPENR_isReady](#) ()
main only, called when //ALTODD
- int [RPOPENR_send](#) (char *buf, int bufsize)
main only, called when //ALTODD
- void [RPOPENR_ready](#) (const OReadyEvent &)
main only, called when //ALTODD
- void [RPOPENR_notify](#) (const ONotifyEvent &event)
main only, called when //ALTODD

Public Attributes

- OSubject * [subject](#) [numOfSubject]
holds information for each of our subjects (data we provide)
- OObserver * [observer](#) [numOfObserver]
holds information for each of the sources we're observing

Protected Member Functions

- void [OpenPrimitives](#) ()
both, called from [SetupOutputs\(\)](#) (mostly for motion, but main does ears), uses [open](#) to tell which to open
- void [SetupOutputs](#) (const bool to_open[NumOutputs])
both, called from [DoInit\(\)](#) (mostly for motion, but main does ears)
- RCRegion * [InitRegion](#) (unsigned int size)
both, called to set up a shared memory region of a given size
- void [addRunLevel](#) ()
Main, checks runLevel and creates StartBehavior when ready.
- OLEDValue [calcLEDValue](#) (unsigned int i, float x)
Motion only, maintains the activation level of the LEDs, returns whether it should be 'fired'.

Protected Attributes

- RCRegion * [motmanMemRgn](#)
Motion creates, Main receives.
- RCRegion * [worldStateMemRgn](#)
Main creates, Motion receives.
- RCRegion * [soundManagerMemRgn](#)
[SoundPlay](#) creates, Main & Motion receives.
- RCRegion * [eventTranslatorQueueMemRgn](#)
Main creates, Motion (& [SoundPlay](#)) receive.
- OPrimitiveID [primIDs](#) [NumOutputs]
both, Main ears only, Motion the rest
- RCRegion * [region](#) [NUM_COMMAND_VECTOR]
both, the actual buffers
- float [ledActivation](#) [NumLEDs]
Motion, used for partial LED activation.

- unsigned int [runLevel](#)
Main, incremented until all sections are ready.
- bool [open](#) [NumOutputs]
both, holds information regarding which outputs are open in ("controlled by") this process
- unsigned int [num_open](#)
both, count of how many are open
- [EventTranslator](#) [etrans](#)
both, allows events to be sent between processes (from other processes besides these two too)
- bool [RPOPENR_isready](#)
true if we've received a ready message from a remote process
- bool [isStopped](#)
true if we've received a DoStart and no DoStop - we need this because sometimes an extra message seems to slip in after we've been told to stop, in which case we should ignore it

Static Protected Attributes

- const unsigned int [NUM_COMMAND_VECTOR](#) = 2
both, for double buffering
- const unsigned int [readyLevel](#) = 5
Main, runLevel at which StartBehavior is created. (1st power event, 1st sensor event, motman init, sndman init, MainObj::DoStart()).

Private Member Functions

- [MMCombo](#) (const [MMCombo](#) &)
should never be called...
- [MMCombo](#) & [operator=](#) (const [MMCombo](#) &)
should never be called...

7.101.2 Constructor & Destructor Documentation

7.101.2.1 MMCombo::MMCombo ()

constructor

Definition at line 26 of file MMCombo.cc.

References Profiler::initBuckets(), ERS210Info::NumOutputs, open, and primIDs.

7.101.2.2 virtual MMCombo::~MMCombo () [inline, virtual]

destructor

Definition at line 33 of file MMCombo.h.

7.101.2.3 MMCombo::MMCombo (const MMCombo &) [private]

should never be called...

7.101.3 Member Function Documentation

7.101.3.1 void MMCombo::addRunLevel () [protected]

Main, checks runLevel and creates StartBehavior when ready.

Definition at line 591 of file MMCombo.cc.

References BehaviorBase::DoStart(), readyLevel, runLevel, and startupBehavior.

7.101.3.2 void MMCombo::BindCont (void * msg) [inline]

main only, called when //ALTODD

Definition at line 60 of file MMCombo.h.

References Wireless::BindCont(), and wireless.

7.101.3.3 OLEDValue MMCombo::calcLEDValue (unsigned int i, float x) [inline, protected]

Motion only, maintains the activation level of the LEDs, returns whether it should be 'fired'.

Definition at line 102 of file MMCombo.h.

References ledActivation.

7.101.3.4 void MMCombo::CloseCont (void * *msg*) [inline]

main only, called when //ALTODD

Definition at line 64 of file MMCombo.h.

References Wireless::CloseCont(), and wireless.

7.101.3.5 void MMCombo::ConnectCont (void * *msg*) [inline]

main only, called when //ALTODD

Definition at line 61 of file MMCombo.h.

References Wireless::ConnectCont(), and wireless.

**7.101.3.6 OStatus MMCombo::DoDestroy (const OSystemEvent &)
[virtual]**

last call (before destructor), clean up memory here

Definition at line 158 of file MMCombo.cc.

References erouter, eventTranslatorQueueMemRgn, motmanMemRgn, ReferenceCounter::RemoveReference(), soundManagerMemRgn, startupBehavior, and worldStateMemRgn.

7.101.3.7 OStatus MMCombo::DoInit (const OSystemEvent &) [virtual]

first call (after constructor), set up memory

Definition at line 40 of file MMCombo.cc.

References ReferenceCounter::AddReference(), config, erouter, etrans, eventTranslatorQueueMemRgn, MotionManager::InitAccess(), InitRegion(), ERS210Info::IsFastOutput, isStopped, ProcessID::MainProcess, MotionManager::MAX_MOTIONS, ProcessID::MotionProcess, motman, motmanMemRgn, ERS210Info::NumOutputs, observer, serr, Socket::setFlushType(), Socket::setForward(), ProcessID::setID(), MotionCommand::setQueue(), EventTranslator::setQueue(), Socket::setTextForward(), SetupOutputs(), Wireless::socket(), sout, startupBehavior, state, subject, vision, wireless, Wireless::WIRELESS_DEF_RECV_SIZE, Wireless::WIRELESS_DEF_SEND_SIZE, and worldStateMemRgn.

7.101.3.8 OStatus MMCombo::DoStart (const OSystemEvent &) [virtual]

second call, ask for messages

Definition at line 117 of file MMCombo.cc.

References `addRunLevel()`, `config`, `Config::main_config::console_port`, `erouter`, `isStopped`, `Wireless::listen()`, `Config::main`, `WorldState::read()`, `serr`, `sout`, `state`, `Config::main_config::stderr_port`, and `wireless`.

7.101.3.9 OStatus MMCombo::DoStop (const OSystemEvent &) [virtual]

next to last call, stop sending and receiving messages

Definition at line 142 of file MMCombo.cc.

References `Wireless::close()`, `BehaviorBase::DoStop()`, `isStopped`, `serr`, `sout`, `startupBehavior`, and `wireless`.

7.101.3.10 void MMCombo::GotEventTranslatorQueue (const ONotifyEvent & event)

motion only, called when EventTranslatorQueue is received

Definition at line 258 of file MMCombo.cc.

References `EventRouter::addTrapper()`, `ASSERT`, `erouter`, `etrans`, `eventTranslatorQueueMemRgn`, `observer`, `MotionCommand::setQueue()`, and `EventTranslator::setQueue()`.

7.101.3.11 void MMCombo::GotImage (const ONotifyEvent & event)

main only, called when a new image is available

Definition at line 410 of file MMCombo.cc.

References `erouter`, `etrans`, `isStopped`, `WorldState::mainProfile`, `observer`, `Vision::processFrame()`, `EventRouter::processTimers()`, `PROFSECTION`, `state`, `EventTranslator::translateEvents()`, and `vision`.

7.101.3.12 void MMCombo::GotMotionManager (const ONotifyEvent & event)

main only, called when motman global is received

Definition at line 224 of file MMCombo.cc.

References `addRunLevel()`, `ASSERT`, `MotionManager::InitAccess()`, `motman`, `motmanMemRgn`, `observer`, and `subject`.

7.101.3.13 void MCombo::GotMotionMsg (const ONotifyEvent & event)

both, called when a new [MotionManagerMsg](#) has been received

Definition at line 472 of file MCombo.cc.

References [isStopped](#), [motman](#), [observer](#), and [MotionManager::receivedMsg\(\)](#).

7.101.3.14 void MCombo::GotPowerEvent (void * msg)

main only, called when a power event occurs (can be just status events)

Definition at line 441 of file MCombo.cc.

References [addRunLevel\(\)](#), [erouter](#), [etrans](#), [isStopped](#), [WorldState::mainProfile](#), [WorldState::powerFlags](#), [EventRouter::processTimers\(\)](#), [PROFSECTION](#), [WorldState::read\(\)](#), [state](#), and [EventTranslator::translateEvents\(\)](#).

7.101.3.15 void MCombo::GotSensorFrame (const ONotifyEvent & event)

main only, called when new sensor information is available

Definition at line 382 of file MCombo.cc.

References [addRunLevel\(\)](#), [erouter](#), [etrans](#), [isStopped](#), [WorldState::mainProfile](#), [observer](#), [EventRouter::processTimers\(\)](#), [PROFSECTION](#), [WorldState::read\(\)](#), [WorldStateSerializer::serialize\(\)](#), [state](#), [EventTranslator::translateEvents\(\)](#), and [WorldState::wsser](#).

7.101.3.16 void MCombo::GotSoundManager (const ONotifyEvent & event)

both, called when the sndman global is received

Definition at line 488 of file MCombo.cc.

References [addRunLevel\(\)](#), [ASSERT](#), [SoundManager::InitAccess\(\)](#), [observer](#), [sndman](#), [soundManagerMemRgn](#), and [subject](#).

7.101.3.17 void MCombo::GotWorldState (const ONotifyEvent & event)

motion only, called when state global is received

Definition at line 193 of file MCombo.cc.

References [ASSERT](#), [observer](#), [state](#), and [worldStateMemRgn](#).

7.101.3.18 RCTRegion * MMCombo::InitRegion (unsigned int *size*)
[protected]

both, called to set up a shared memory region of a given size

Will round up size to the nearest page

Definition at line 582 of file MMCombo.cc.

References ASSERT.

7.101.3.19 void MMCombo::ListenCont (void * *msg*) [inline]

main only, called when //ALTODD

Definition at line 59 of file MMCombo.h.

References Wireless::ListenCont(), and wireless.

7.101.3.20 void MMCombo::OpenPrimitives () [protected]

both, called from [SetupOutputs\(\)](#) (mostly for motion, but main does ears), uses [open](#) to tell which to open

Definition at line 502 of file MMCombo.cc.

References ERS210Info::NumOutputs, open, primIDs, and ERS210Info::Primitive-Name.

7.101.3.21 MMCombo& MMCombo::operator= (const MMCombo &)
[private]

should never be called...

7.101.3.22 void MMCombo::ReadyRegisterEventTranslatorQueue (const OReadyEvent &)

main only, send out the EventTranslatorQueue

Called when MotoObj is initially ready as well as when it has finished processing the previous message - we only want to do this the first time otherwise we infinite loop.

Definition at line 245 of file MMCombo.cc.

References eventTranslatorQueueMemRgn, and subject.

7.101.3.23 void MMCombo::ReadyRegisterMotionManager (const OReadyEvent &)

motion only, send out motman global

Called when MainObj is initially ready as well as when it has finished processing the previous message - we only want to do this the first time otherwise we infinite loop.

Definition at line 211 of file MMCombo.cc.

References motmanMemRgn, and subject.

7.101.3.24 void MMCombo::ReadyRegisterWorldState (const OReadyEvent &)

main only, send out the state global

Called when MotoObj is initially ready as well as when it has finished processing the previous message - we only want to do this the first time otherwise we infinite loop.

Definition at line 180 of file MMCombo.cc.

References subject, and worldStateMemRgn.

7.101.3.25 void MMCombo::ReadySendJoints (const OReadyEvent & event)

motion only (until main does ears again, then both) calls SendJoints, if DoStart has already been called

Definition at line 276 of file MMCombo.cc.

References ASSERTRET, ERS210Info::BinJointOffset, calcLEDValue(), ProcessID::getID(), Profiler::getNewID(), MotionManager::getOutputs(), isStopped, ERS210Info::LEDOffset, ProcessID::MainProcess, WorldState::mainProfile, ProcessID::MotionProcess, WorldState::motionProfile, motman, NUM_COMMAND_VECTOR, num_open, ERS210Info::NumBinJoints, ERS210Info::NumFrames, ERS210Info::NumLEDs, ERS210Info::NumOutputs, ERS210Info::NumPIDJoints, ERS210Info::NumSlowFrames, open, WorldState::outputs, ERS210Info::PIDJointOffset, primIDs, region, Profiler::Timer::setID(), state, subject, MotionManager::updatePIDs(), and MotionManager::updateWorldState().

7.101.3.26 void MMCombo::ReceiveCont (void * msg) [inline]

main only, called when //ALTODD

Definition at line 63 of file MMCombo.h.

References Wireless::ReceiveCont(), and wireless.

7.101.3.27 bool MMCombo::ROPENR_isReady () [inline]

main only, called when //ALTODD

Definition at line 66 of file MMCombo.h.

References ROPENR_isready.

7.101.3.28 void MMCombo::ROPENR_notify (const ONotifyEvent & event)

main only, called when //ALTODD

Definition at line 601 of file MMCombo.cc.

References erouter, observer, and EventRouter::postEvent().

**7.101.3.29 void MMCombo::ROPENR_ready (const OReadyEvent &)
[inline]**

main only, called when //ALTODD

Definition at line 69 of file MMCombo.h.

References ROPENR_isready.

7.101.3.30 int MMCombo::ROPENR_send (char * buf, int bufsize)

main only, called when //ALTODD

Definition at line 616 of file MMCombo.cc.

References ROPENR_isready, and subject.

7.101.3.31 void MMCombo::SendCont (void * msg) [inline]

main only, called when //ALTODD

Definition at line 62 of file MMCombo.h.

References Wireless::SendCont(), and wireless.

**7.101.3.32 void MMCombo::SetupOutputs (const bool to_open[NumOutputs])
[protected]**

both, called from [DoInit\(\)](#) (mostly for motion, but main does ears)

Definition at line 513 of file MMCombo.cc.

References ASSERT, ERS210Info::BinJointOffset, ERS210Info::LEDOffset, NUM_COMMAND_VECTOR, num_open, ERS210Info::NumBinJoints, ERS210Info::NumFrames, ERS210Info::NumLEDs, ERS210Info::NumOutputs, ERS210Info::NumPIDJoints, ERS210Info::NumSlowFrames, open, OpenPrimitives(), ERS210Info::PIDJointOffset, primIDs, and region.

7.101.4 Member Data Documentation

7.101.4.1 [EventTranslator MMCombo::etrans](#) [protected]

both, allows events to be sent between processes (from other processes besides these two too)

Definition at line 95 of file MMCombo.h.

7.101.4.2 [RCRegion* MMCombo::eventTranslatorQueueMemRgn](#) [protected]

Main creates, Motion (& [SoundPlay](#)) receive.

Definition at line 80 of file MMCombo.h.

7.101.4.3 [bool MMCombo::isStopped](#) [protected]

true if we've received a DoStart and no DoStop - we need this because sometimes an extra message seems to slip in after we've been told to stop, in which case we should ignore it

Definition at line 99 of file MMCombo.h.

7.101.4.4 [float MMCombo::ledActivation](#)[NumLEDs] [protected]

Motion, used for partial LED activation.

Definition at line 86 of file MMCombo.h.

7.101.4.5 [RCRegion* MMCombo::motmanMemRgn](#) [protected]

Motion creates, Main receives.

Definition at line 77 of file MMCombo.h.

7.101.4.6 `const unsigned int MMCombo::NUM_COMMAND_VECTOR = 2`
[static, protected]

both, for double buffering

Definition at line 83 of file MMCombo.h.

7.101.4.7 `unsigned int MMCombo::num_open` [protected]

both, count of how many are open

Definition at line 93 of file MMCombo.h.

7.101.4.8 `OObserver* MMCombo::observer[numOfObserver]`

holds information for each of the sources we're observing

Definition at line 36 of file MMCombo.h.

7.101.4.9 `bool MMCombo::open[NumOutputs]` [protected]

both, holds information regarding which outputs are open in ("controlled by") this process

Definition at line 92 of file MMCombo.h.

7.101.4.10 `OPrimitiveID MMCombo::primIDs[NumOutputs]` [protected]

both, Main ears only, Motion the rest

Definition at line 82 of file MMCombo.h.

7.101.4.11 `const unsigned int MMCombo::readyLevel = 5` [static, protected]

Main, runLevel at which StartBehavior is created. (1st power event, 1st sensor event, motman init, sndman init, MainObj::DoStart()).

Definition at line 89 of file MMCombo.h.

7.101.4.12 `RCRegion* MMCombo::region[NUM_COMMAND_VECTOR]`
[protected]

both, the actual buffers

Definition at line 84 of file MMCombo.h.

7.101.4.13 **bool** [MMCombo::RPOPENR_isready](#) [protected]

true if we've received a ready message from a remote process

Definition at line 97 of file MMCombo.h.

7.101.4.14 **unsigned int** [MMCombo::runLevel](#) [protected]

Main, incremented until all sections are ready.

Definition at line 88 of file MMCombo.h.

7.101.4.15 **RCRegion*** [MMCombo::soundManagerMemRgn](#) [protected]

[SoundPlay](#) creates, Main & Motion receives.

Definition at line 79 of file MMCombo.h.

7.101.4.16 **OSubject*** [MMCombo::subject](#)[numOfSubject]

holds information for each of our subjects (data we provide)

Definition at line 35 of file MMCombo.h.

7.101.4.17 **RCRegion*** [MMCombo::worldStateMemRgn](#) [protected]

Main creates, Motion receives.

Definition at line 78 of file MMCombo.h.

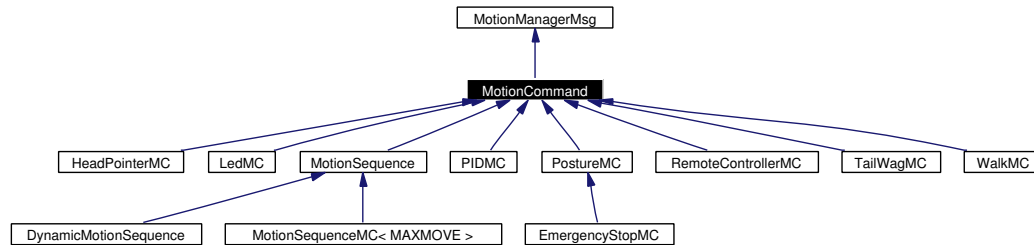
The documentation for this class was generated from the following files:

- [MMCombo.h](#)
- [MMCombo.cc](#)

7.102 MotionCommand Class Reference

```
#include <MotionCommand.h>
```

Inheritance diagram for MotionCommand:



7.102.1 Detailed Description

The abstract base class for motions, provides common interface. All motions should inherit from this.

For instructions on how to create:

- **a new subclass** of MotionCommand, read on. Also see the step-by-step [guide](#)
- **an instantiation** of a MotionCommand subclass, see [MotionManager](#)

To create a new type of motion, you'll want to subclass this. You don't need to do anything fancy, but just be sure to override the 3 abstract functions.

When an output is set to a value, that value is retained until it is set to a new value, even if the MotionCommand that set it is pruned or stops using the output. Outputs never "reset" to 0 or some other relatively arbitrary base value.

However, PID values will reset to the default values if pruned or not set since these values do have a base value which you will want to use 99% of the time.

Be aware that there is a delay between when you set a joint to a value and that actually is taken into account by the system - it's on the order of $\text{RobotInfo::FrameTime} * \text{RobotInfo::NumFrames}$ (currently $8 * 4 = 32$ ms, at most $2 * 8 * 4 = 64$ ms) This is because the commands are double buffered. PIDs, on the other hand, seem to take effect more quickly. This un-synchronization can sometimes cause annoying jerkiness (mainly on startup, where there's a large difference between desired and target values.)

Here is the cycle of calls made by [MotionManager](#) to your command:

1. [shouldPrune\(\)](#) (by default, this will call [isAlive\(\)](#) iff `autoprune==true`)

2. `updateJointCmds()` (assuming the MC wasn't pruned after the previous step)

So, if you want to hold a joint at a value, each time your `updateJointCmds()` function is called, you should tell the [MotionManager](#) to keep the joint there (using one of [MotionManager::setOutput\(\)](#)'s). If you do not set a joint after a call to `updateJointCmds`, the [MotionManager](#) will assume you are no longer using that joint and a lower priority `MotionCommand` may inherit it.

`MotionCommands` which generate events should use the inherited [postEvent\(\)](#) instead of trying to access a global erouter - the inherited version will properly handle sending the events, trying to access a non-shared global like erouter could cause problems.

Warning:

Be careful what you call in [MotionManager](#)

Some functions are marked `MotionCommand-safe` - this is another issue due to our "fake" fork. In short, when a function is called on a `MotionCommand`, it thinks it is still running in whatever process created it, not the process that actually made the function call. Thus, when Motion calls [updateOutputs\(\)](#), calls that the `MotionCommand` makes are indistinguishable from concurrent calls from Main. This can cause deadlock if a function is called which locks the [MotionManager](#). To get around this, we need to pass the 'this' parameter on functions that require a lock, namely [MotionManager::setOutputs\(\)](#). This allows the [MotionManager](#) to figure out that it's the same `MotionCommand` it previously called [updateOutputs\(\)](#) on, and thus avoid trying to lock itself again.

Don't store pointers in motion commands!

Since motion commands are in shared memory, and these shared memory regions can have different base pointers in each process, pointers will only be valid in the process from which they were assigned. In other processes, that address may point to something else, especially if it was pointing outside of the shared memory regions.

There are convoluted ways of getting around this. If needed, [MotionManager](#) could be modified to hand out shared memory regions upon request. Let's try to avoid this for now. Keep `MotionCommands` simple, without dynamic memory. Do more complicated stuff with behaviors, which only have to run in Main.

See also:

REGDEF
REGIMP

Definition at line 66 of file `MotionCommand.h`.

*** **INHERITED:** ***

- [MotionCommand](#) ()

Constructor: Defaults to `kStdPriority` and `autoprune==true`.

- virtual [~MotionCommand](#) ()
Destructor.
- virtual void [DoStart](#) ()
called after this is added to [MotionManager](#)
- virtual void [DoStop](#) ()
called after this is removed from [MotionManager](#)
- virtual bool [isActive](#) () const
returns true if the MotionCommand is currently running (although it may be overridden by a higher priority MotionCommand)
- virtual bool [getAutoPrune](#) ()
- virtual void [setAutoPrune](#) (bool ap)
- virtual bool [shouldPrune](#) ()
whether this motion should be removed from its motion group automatically ([MotionCommand::autoprune](#) && !isAlive())
- void [setQueue](#) ([EventTranslator::Queue_t](#) *q)
only called from [MMCombo](#) during process setup, allows MotionCommands to send events
- double [interpolate](#) (double a, double b, double x)
this utility function will probably be of use to a lot of MotionCommand's
- float [interpolate](#) (float a, float b, float x)
this utility function will probably be of use to a lot of MotionCommand's
- void [interpolate](#) (const [OutputCmd](#) &a, const [OutputCmd](#) &b, float x, [OutputCmd](#) &r)
this utility function will probably be of use to a lot of MotionCommand's, see [interpolate\(double a,double b,double r\)](#)

Public Member Functions

*** **ABSTRACT:** *** (must be defined by subclasses)

- virtual int [updateOutputs](#) ()=0
is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)

- virtual int `isDirty` ()=0
not used by `MotionManager` at the moment, but could be used to reduce recomputation, and you may find it useful
- virtual int `isAlive` ()=0
used to prune "dead" motions from the `MotionManager`

Static Protected Member Functions

- void `postEvent` (const `EventBase` &event)
calls `EventTranslator::enqueue` directly (avoids needing `erouter`, which is a non-shared global, causes problems with context, grr, silly OS)

Protected Attributes

- int `autoprune`
default true, autoprune setting, if this is true and `isAlive()` returns false, `MotionManager` will attempt to remove the MC automatically
- bool `started`
true if the `MotionCommand` is currently running (although it may be overridden by a higher priority `MotionCommand`)

Static Protected Attributes

- `EventTranslator::Queue_t * queue` = NULL
queue to store outgoing events in - call the `MotionCommand::postEvent`

7.102.2 Constructor & Destructor Documentation

7.102.2.1 `MotionCommand::MotionCommand ()` [inline]

Constructor: Defaults to `kStdPriority` and `autoprune==true`.

Definition at line 96 of file `MotionCommand.h`.

References `autoprune`, and `started`.

7.102.2.2 `virtual MotionCommand::~~MotionCommand ()` [inline, virtual]

Destructor.

Definition at line 98 of file MotionCommand.h.

7.102.3 Member Function Documentation

7.102.3.1 `virtual void MotionCommand::DoStart ()` [inline, virtual]

called after this is added to [MotionManager](#)

Reimplemented in [WalkMC](#).

Definition at line 101 of file MotionCommand.h.

References started.

7.102.3.2 `virtual void MotionCommand::DoStop ()` [inline, virtual]

called after this is removed from [MotionManager](#)

Reimplemented in [WalkMC](#).

Definition at line 104 of file MotionCommand.h.

References started.

7.102.3.3 `virtual bool MotionCommand::getAutoPrune ()` [inline, virtual]

Returns:

current setting of autopruning - used to remove motion from groups when !isAlive()

Definition at line 110 of file MotionCommand.h.

References autoprun.

7.102.3.4 `void MotionCommand::interpolate (const OutputCmd & a, const OutputCmd & b, float x, OutputCmd & r)` [inline, static, protected]

this utility function will probably be of use to a lot of MotionCommand's, see [interpolate\(double a,double b,double r\)](#)

interpolates both value and weights of JointCmd's

Parameters:

- a* first joint cmd
- b* second joint cmd
- x* weight to favor b's value and weight
- r* joint cmd to store the result

Definition at line 149 of file MotionCommand.h.

References `interpolate()`, `OutputCmd::set()`, `OutputCmd::value`, and `OutputCmd::weight`.

7.102.3.5 `float MotionCommand::interpolate (float a, float b, float x)` `[inline, static, protected]`

this utility function will probably be of use to a lot of MotionCommand's

Does a weighted average of *a* and *b*, favoring *b* by *x* percent (so *x*==0 results in *a*, *x*==1 results in *b*)

Parameters:

- a* first value
- b* second value
- x* weight of second value as opposed to first

Returns:

$$a * (1.0 - x) + b * x$$

Todo

- replace with a more fancy spline based thing?

Definition at line 140 of file MotionCommand.h.

7.102.3.6 `double MotionCommand::interpolate (double a, double b, double x)` `[inline, static, protected]`

this utility function will probably be of use to a lot of MotionCommand's

Does a weighted average of *a* and *b*, favoring *b* by *x* percent (so *x*==0 results in *a*, *x*==1 results in *b*)

Parameters:

- a* first value
- b* second value
- x* weight of second value as opposed to first

Returns:

$$a * (1.0 - x) + b * x$$

Todo

- replace with a more fancy spline based thing?

Definition at line 130 of file MotionCommand.h.

7.102.3.7 virtual bool MotionCommand::isActive () const [inline, virtual]

returns true if the MotionCommand is currently running (although it may be overridden by a higher priority MotionCommand)

Definition at line 107 of file MotionCommand.h.

References started.

7.102.3.8 virtual int MotionCommand::isAlive () [pure virtual]

used to prune "dead" motions from the [MotionManager](#)

note that a motion could be "paused" or inactive and therefore not dirty, but still alive, biding its time to "strike" ;)

Returns:

zero if the motion is still processing, non-zero otherwise

Implemented in [HeadPointerMC](#), [LedMC](#), [MotionSequence](#), [PIDMC](#), [PostureMC](#), [RemoteControllerMC](#), [TailWagMC](#), and [WalkMC](#).

7.102.3.9 virtual int MotionCommand::isDirty () [pure virtual]

not used by [MotionManager](#) at the moment, but could be used to reduce recomputation, and you may find it useful

Returns:

zero if none of the commands have changed since last getJointCmd(), else non-zero

Implemented in [HeadPointerMC](#), [LedMC](#), [MotionSequence](#), [PIDMC](#), [PostureMC](#), [RemoteControllerMC](#), [TailWagMC](#), and [WalkMC](#).

7.102.3.10 `void MotionCommand::postEvent (const EventBase & event)`
[inline, static, protected]

calls [EventTranslator::enqueue](#) directly (avoids needing erouter, which is a non-shared global, causes problems with context, grr, silly OS)

Definition at line 155 of file MotionCommand.h.

References [EventTranslator::enqueue\(\)](#), and [queue](#).

7.102.3.11 `virtual void MotionCommand::setAutoPrune (bool ap)` [inline, virtual]

Parameters:

ap bool representing requested autopruning setting

Definition at line 113 of file MotionCommand.h.

References [autoprun](#).

7.102.3.12 `void MotionCommand::setQueue (EventTranslator::Queue_t * q)`
[inline, static]

only called from [MMCombo](#) during process setup, allows MotionCommands to send events

Definition at line 120 of file MotionCommand.h.

References [queue](#).

7.102.3.13 `virtual bool MotionCommand::shouldPrune ()` [inline, virtual]

whether this motion should be removed from its motion group automatically ([MotionCommand::autoprun](#) && !isAlive())

Returns:

([MotionCommand::autoprun](#) && !isAlive())

Definition at line 117 of file MotionCommand.h.

References [autoprun](#), and [isAlive\(\)](#).

7.102.3.14 **virtual int MotionCommand::updateOutputs ()** [pure virtual]

is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)

Returns:

zero if no changes were made, non-zero otherwise

See also:

[RobotInfo::NumFrames](#)
[RobotInfo::FrameTime](#)

Implemented in [DynamicMotionSequence](#), [EmergencyStopMC](#), [HeadPointerMC](#), [LedMC](#), [MotionSequence](#), [MotionSequenceMC< MAXMOVE >](#), [PIDMC](#), [PostureMC](#), [RemoteControllerMC](#), [TailWagMC](#), and [WalkMC](#).

7.102.4 Member Data Documentation

7.102.4.1 **int MotionCommand::autoprune** [protected]

default true, autoprune setting, if this is true and [isAlive\(\)](#) returns false, [MotionManager](#) will attempt to remove the MC automatically

Definition at line 159 of file [MotionCommand.h](#).

7.102.4.2 **EventTranslator::Queue.t * MotionCommand::queue = NULL** [static, protected]

queue to store outgoing events in - call the [MotionCommand::postEvent](#)

Definition at line 3 of file [MotionCommand.cc](#).

7.102.4.3 **bool MotionCommand::started** [protected]

true if the MotionCommand is currently running (although it may be overridden by a higher priority MotionCommand)

Definition at line 160 of file [MotionCommand.h](#).

The documentation for this class was generated from the following files:

- [MotionCommand.h](#)
- [MotionCommand.cc](#)

7.103 MotionManager Class Reference

```
#include <MotionManager.h>
```

7.103.1 Detailed Description

The purpose of this class is to serialize access to the MotionCommands and simplify their sharing between memory spaces.

Since MotionObject and MainObject run as separate processes, they could potentially try to access the same motion command at the same time, leading to unpredictable behavior. The MotionManager enforces a set of locks to solve this

The other problem is that we are sharing between processes. MotionManager will do what's necessary to distribute new MotionCommand's to all the processes (currently just MainObj and MotoObj)

You should be able to create and add a new motion in one line:

```
motman->addMotion(SharedObject<YourMC>([arg1,[arg2,...]]) [, priority [, autoprune] ]);
```

But if you want to do some more initializations not handled by the constructor (the arg1, arg2, ... params) then you would want to do something like the following:

```
SharedObject<YourMC> tmpvar([arg1,[arg2,...]]);
tmpvar->cmd1();
tmpvar->cmd2();
//...
motman->addMotion(tmpvar [, ...]);
```

Notice that tmpvar is of type [SharedObject](#), but you're calling YourMC functions on it... [SharedObject](#) is a "smart pointer" which will pass your function calls on to the underlying templated type. Isn't C++ great? :)

Warning:

Once the [MotionCommand](#) has been added you must check it out to modify it or risk concurrent access problems.

See also:

[MotionCommand](#) for information on creating new motion primitives.

[MMAccessor](#) for information on accessing motions after you've added them to MotionManager.

Definition at line 56 of file MotionManager.h.

Priority Level Constants

Just to give you a general idea what values to use for different priority levels

- const float `kIgnoredPriority` = -1
won't be expressed, handy if you want to temporarily pause something
- const float `kBackgroundPriority` = 0
*will only be expressed if **nothing** else is using that joint*
- const float `kLowPriority` = 5
for stuff that's not background but lower than standard
- const float `kStdPriority` = 10
for every-day commands
- const float `kHighPriority` = 50
for stuff that should over ride standard stuff
- const float `kEmergencyPriority` = 100
for really important stuff, such as the emergency stop
- typedef unsigned short `accID_t`
type to use to refer to accessors of MotionManager (or its locks)
- typedef `ListMemBuf< OutputState, MAX_MOTIONS > cmdstatelist_t`
shorthand for a list of OutputState's
- void `setPID` (unsigned int j, const float p[3])
LOCKS MotionManager you can call this to set the PID values directly (instead of using a motion command) Be careful though
- void `func_begin` ()
called at the begining of many functions to lock MotionManager
- void `func_end` ()
called at the end of a function which called `func_begin()` to release it
- template<class T> T `func_end` (T val)
same as `func_end()`, except passes return value through
- `MC_ID skip_ahead` (`MC_ID` mcid) const

during iteration, skips over motioncommands which are still in transit from on *OOb-ject* to another

- **MC_ID pop_free ()**
pulls an entry from cmdlist's free section and returns its index
- **void push_free (MC_ID a)**
puts an entry back into cmdlist's free section
- **ListMemBuf< PIDUpdate, NumPIDJoints > pidchanges**
stores PID updates, up to one per joint (if same is set more than once, it's just overwrites previous update)
- **ListMemBuf< CommandEntry, MAX_MOTIONS, MC_ID > cmdlist**
the list where MotionCommands are stored, remember, we're in a shared memory region with different base addresses - no pointers!
- **MC_ID cur_cmd**
MC_ID of the *MotionCommand* currently being updated by *getOutputs()*, or NULL if not in *getOutputs*. This is used by the *setOutput()*'s to tell which *MotionCommand* is calling.
- **MutexLock< MAX_ACCESS > MMlock**
The main lock for the class.
- **cmdstatelist_t cmdstates [NumOutputs]**
requested positions by each of the MC's for each of the outputs
- **float cmdSums [NumOutputs]**
Holds the final values for the outputs of the last frame generated.
- **OutputCmd cmds [NumOutputs]**
Holds the weighted values and total weight for the outputs of the last frame.
- **accID_t numAcc**
The number of accessors who have registered with *InitAccess()*.
- **OSubject * subjs [MAX_ACCESS]**
The OSubject for each process (accessor) on which it should be broadcast when a command is added.
- **unsigned int _MMaccID = -1U**
Stores the accessor for the current process.

- [MotionManager](#) (const [MotionManager](#) &)
this shouldn't be called...
- [MotionManager](#) & operator= (const [MotionManager](#) &)
this shouldn't be called...

Public Types

- typedef [MotionManagerMsg::MC_ID](#) MC_ID
use this type when referring to the ID numbers that MotionManager hands out

Public Member Functions

- [MotionManager](#) ()
Constructor, sets all the outputs to 0.
- void [InitAccess](#) (OSubject *subj)
LOCKS MotionManager *Everyone who is planning to use the MotionManager needs to call this before they access it or suffer a horrible fate*
- void [receivedMsg](#) (const ONotifyEvent &event)
LOCKS MotionManager *This gets called by an OObject when it receives a message from one of the other OObject's MotionManagerComm Subject*

MotionCommand Safe

- void [setOutput](#) (const [MotionCommand](#) *caller, unsigned int output, const [OutputCmd](#) &cmd)
LOCKS MotionManager *Requests a value be set for the specified output, copies cmd across frames*
- void [setOutput](#) (const [MotionCommand](#) *caller, unsigned int output, const [OutputCmd](#) &cmd, unsigned int frame)
LOCKS MotionManager *Requests a value be set for the specified output in the specified frame*
- void [setOutput](#) (const [MotionCommand](#) *caller, unsigned int output, const [OutputCmd](#) cmd[NumFrames])
LOCKS MotionManager *Requests a value be set for the specified output across frames*

- void `setOutput` (const `MotionCommand` *caller, unsigned int output, const `OutputPID` &pid)
LOCKS MotionManager Requests a PID be set for the specified output, notice that this might be overruled by a higher priority motion
- void `setOutput` (const `MotionCommand` *caller, unsigned int output, const `OutputCmd` &cmd, const `OutputPID` &pid)
LOCKS MotionManager Requests a value and PID be set for the specified output
- void `setOutput` (const `MotionCommand` *caller, unsigned int output, const `OutputCmd` cmd[NumFrames], const `OutputPID` &pid)
LOCKS MotionManager Requests a value and PID be set for the specified output
- const `OutputCmd` & `getOutputCmd` (unsigned int output) const
Returns the value of the output last sent to the OS. Note that this will differ from the sensed value in state, even when staying still. There is no corresponding `getOutputPID` because this value *will* duplicate the value in state.
- void `setPriority` (`MC_ID` mcid, float p)
sets the priority level of a `MotionCommand`
- float `getPriority` (`MC_ID` mcid) const
returns priority level of a `MotionCommand`
- `MC_ID` `begin` () const
returns the `MC_ID` of the first `MotionCommand`
- `MC_ID` `next` (`MC_ID` cur) const
returns the `MC_ID` of `MotionCommand` following the one that is passed
- `MC_ID` `end` () const
returns the `MC_ID` of the one-past-the-end `MotionCommand` (like the STL)
- unsigned int `size` () const
returns the number of `MotionCommands` being managed

MotionCommand "Risky"

You can have one MC check out and modify another, but make sure the other MC doesn't call `setOutput()`

- `MotionCommand` * `checkoutMotion` (`MC_ID` mcid, bool block=true)
locks the command and possibly performs RTTI conversion; supports recursive calls

- void **checkinMotion** (**MC_ID** mcid)
marks a [MotionCommand](#) as unused
- **MotionCommand** * **peekMotion** (**MC_ID** mcid)
*allows access to a [MotionCommand](#) without checking it out **never** call a function based on this, only access member fields through it*
- unsigned int **checkoutLevel** (**MC_ID** mcid)
returns the number of times mcid has been checked out minus the times it's been checked in

MotionCommand Unsafe

- **MC_ID** **addMotion** (const **SharedObjectBase** &sm)
LOCKS MotionManager *Creates a new [MotionCommand](#), automatically sharing it between processes (there is some lag time here)*
- **MC_ID** **addMotion** (const **SharedObjectBase** &sm, float priority)
LOCKS MotionManager *allows a quick way to set a priority level of a new [MotionCommand](#)*
- **MC_ID** **addMotion** (const **SharedObjectBase** &sm, bool autoprunes)
LOCKS MotionManager *allows a quick way to set the autoprunes flag*
- **MC_ID** **addMotion** (const **SharedObjectBase** &sm, float priority, bool autoprunes)
LOCKS MotionManager *Call one of these to add a [MotionCommand](#) to the MotionManager, using the [SharedObject](#) class*
- void **removeMotion** (**MC_ID** mcid)
LOCKS MotionManager *removes the specified [MotionCommand](#)*
- void **lock** ()
*gets an exclusive lock on MotionManager - functions marked **LOCKS MotionManager** will cause (and require) this to happen automatically*
- bool **trylock** ()
tries to get a lock without blocking
- void **release** ()
releases a lock on the motion manager

- void [getOutputs](#) (float outputs[NumFrames][NumOutputs])

LOCKS MotionManager called by MotionObject to fill in the output values for the next ::NumFrames frames (only MotoObj should call this...)

- void [updateWorldState](#) ()

call this when you want MotionManager to set the [WorldState](#) to reflect what things should be for unsensed outputs (LEDs, ears) (only MotoObj should be calling this...)

- bool [updatePIDs](#) (OPrimitiveID primIDs[NumOutputs])

call this when you want MotionManager to update modified PID values, returns true if changes made (only MotoObj should be calling this...)

Static Public Attributes

- const unsigned int [MAX_ACCESS](#) = 2

This is the number of processes which will be accessing the MotionManager.

- const unsigned int [MAX_MOTIONS](#) = 64

This is the maximum number of Motions which can be managed, can probably be increased reasonably without trouble.

- const [MC_ID](#) [invalid_MC_ID](#) = ([MC_ID](#))-1

for errors and undefined stuff

7.103.2 Member Typedef Documentation

7.103.2.1 typedef unsigned short [MotionManager::accID_t](#) [protected]

type to use to refer to accessors of MotionManager (or its locks)

Definition at line 171 of file MotionManager.h.

7.103.2.2 typedef [ListMemBuf](#)<[OutputState](#),[MAX_MOTIONS](#)> [MotionManager::cmdstatelist_t](#) [protected]

shorthand for a list of OutputState's

Definition at line 210 of file MotionManager.h.

7.103.2.3 typedef [MotionManagerMsg::MC_ID](#) MotionManager::MC_ID

use this type when referring to the ID numbers that MotionManager hands out
Definition at line 65 of file MotionManager.h.

7.103.3 Constructor & Destructor Documentation

7.103.3.1 MotionManager::MotionManager ()

Constructor, sets all the outputs to 0.

Definition at line 30 of file MotionManager.cc.

References `cmdSums`, `ERS210Info::NumOutputs`, and `uint`.

7.103.3.2 MotionManager::MotionManager (const [MotionManager](#) &) [private]

this shouldn't be called...

7.103.4 Member Function Documentation

7.103.4.1 [MotionManager::MC_ID](#) MotionManager::addMotion (const [SharedObjectBase](#) & *sm*, float *priority*, bool *autoprune*)

LOCKS MotionManager Call one of these to add a [MotionCommand](#) to the MotionManager, using the [SharedObject](#) class

Definition at line 454 of file MotionManager.cc.

References `addMotion()`, `SharedObjectBase::data()`, `end()`, `func_begin()`, `func_end()`, `invalid_MC_ID`, `MC_ID`, `MotionCommand::setAutoPrune()`, and `setPriority()`.

7.103.4.2 [MotionManager::MC_ID](#) MotionManager::addMotion (const [SharedObjectBase](#) & *sm*, bool *autoprune*)

LOCKS MotionManager allows a quick way to set the autoprune flag

Definition at line 444 of file MotionManager.cc.

References `addMotion()`, `SharedObjectBase::data()`, `invalid_MC_ID`, and `MotionCommand::setAutoPrune()`.

7.103.4.3 [MotionManager::MC_ID](#) MotionManager::addMotion (const [SharedObjectBase](#) & *sm*, float *priority*)

LOCKS MotionManager allows a quick way to set a priority level of a new [MotionCommand](#)

Definition at line 436 of file MotionManager.cc.

References addMotion(), end(), func_begin(), func_end(), MC_ID, and setPriority().

7.103.4.4 [MotionManager::MC_ID](#) MotionManager::addMotion (const [SharedObjectBase](#) & *sm*)

LOCKS MotionManager Creates a new [MotionCommand](#), automatically sharing it between processes (there is some lag time here)

Definition at line 409 of file MotionManager.cc.

References _MMaccID, ASSERT, cmdlist, SharedObjectBase::data(), ListMemBuf< CommandEntry, MAX_MOTIONS, MC_ID >::end(), func_begin(), func_end(), SharedObjectBase::getRegion(), invalid_MC_ID, kStdPriority, MAX_ACCESS, MC_ID, numAcc, pop_free(), and subjs.

7.103.4.5 [MC_ID](#) MotionManager::begin () const [inline]

returns the MC_ID of the first [MotionCommand](#)

Definition at line 88 of file MotionManager.h.

References ListMemBuf< CommandEntry, MAX_MOTIONS, MC_ID >::begin(), cmdlist, MC_ID, and skip_ahead().

7.103.4.6 void MotionManager::checkinMotion ([MC_ID](#) *mcid*)

marks a [MotionCommand](#) as unused

Definition at line 532 of file MotionManager.cc.

References cmdlist.

7.103.4.7 unsigned int MotionManager::checkoutLevel ([MC_ID](#) *mcid*) [inline]

returns the number of times *mcid* has been checked out minus the times it's been checked in

Definition at line 99 of file MotionManager.h.

References cmdlist.

7.103.4.8 [MotionCommand](#) * MotionManager::checkoutMotion ([MC_ID](#) *mcid*, bool *block* = true)

locks the command and possibly performs RTTI conversion; supports recursive calls

Definition at line 501 of file MotionManager.cc.

References `_MMaccID`, `accID_t`, `cmdlist`, and `MAX_MOTIONS`.

7.103.4.9 [MC_ID](#) MotionManager::end () const [inline]

returns the `MC_ID` of the one-past-the-end [MotionCommand](#) (like the STL)

Definition at line 90 of file MotionManager.h.

References `cmdlist`, `ListMemBuf< CommandEntry`, `MAX_MOTIONS`, `MC_ID`
`>::end()`, and `MC_ID`.

7.103.4.10 void MotionManager::func_begin () [inline, protected]

called at the begining of many functions to lock MotionManager

Definition at line 173 of file MotionManager.h.

References `_MMaccID`, `MutexLock< MAX_ACCESS >::lock()`, and `MMlock`.

7.103.4.11 template<class T> T MotionManager::func_end (T *val*) [inline, protected]

same as [func_end\(\)](#), except passes return value through

Definition at line 175 of file MotionManager.h.

References `func_end()`.

7.103.4.12 void MotionManager::func_end () [inline, protected]

called at the end of a function which called [func.begin\(\)](#) to release it

Definition at line 174 of file MotionManager.h.

References `MMlock`, and `MutexLock< MAX_ACCESS >::release()`.

7.103.4.13 `const OutputCmd& MotionManager::getOutputCmd (unsigned int output) const` [inline]

Returns the value of the output last sent to the OS. Note that this will differ from the sensed value in state, even when staying still. There is no corresponding getOutputPID because this value *will* duplicate the value in state.

Definition at line 82 of file MotionManager.h.

References `cmds`.

7.103.4.14 `void MotionManager::getOutputs (float outputs[NumFrames][NumOutputs])`

LOCKS MotionManager called by MotionObject to fill in the output values for the next `::NumFrames` frames (only MotoObj should call this...)

What's worse? A plethora of functions which are only called, and only useful at one place, or a big massive function which doesn't pollute the namespace? This is the latter, for better or worse.

Definition at line 199 of file MotionManager.cc.

References `ASSERT`, `ListMemBuf< OutputState, MAX_MOTIONS >::begin()`, `begin()`, `checkinMotion()`, `checkoutMotion()`, `ListMemBuf< OutputState, MAX_MOTIONS >::clear()`, `cmdlist`, `cmds`, `cmdstates`, `cmdSums`, `cur_cmd`, `ListMemBuf< OutputState, MAX_MOTIONS >::end()`, `end()`, `func_begin()`, `func_end()`, `invalid_MC_ID`, `MC_ID`, `ListMemBuf< OutputState, MAX_MOTIONS >::next()`, `next()`, `ERS210Info::NumFrames`, `ERS210Info::NumLEDs`, `ERS210Info::NumOutputs`, `ERS210Info::NumPIDJoints`, `ERS210Info::PIDJointOffset`, `removeMotion()`, `setPID()`, `MotionCommand::shouldPrune()`, `state`, `uint`, `MotionCommand::updateOutputs()`, `OutputCmd::value`, and `OutputCmd::weight`.

7.103.4.15 `float MotionManager::getPriority (MC_ID mcid) const` [inline]

returns priority level of a `MotionCommand`

Definition at line 84 of file MotionManager.h.

References `cmdlist`.

7.103.4.16 `void MotionManager::InitAccess (OSubject * subj)`

LOCKS MotionManager Everyone who is planning to use the MotionManager needs to call this before they access it or suffer a horrible fate

Definition at line 43 of file MotionManager.cc.

References `_MMaccID`, `cmdlist`, `MutexLock< MAX_ACCESS >::lock()`, `MAX_ACCESS`, `MMlock`, `numAcc`, `MutexLock< MAX_ACCESS >::release()`, `ListMemBuf< CommandEntry, MAX_MOTIONS, MC_ID >::size()`, and `subjs`.

7.103.4.17 `void MotionManager::lock () [inline]`

gets an exclusive lock on MotionManager - functions marked **LOCKS MotionManager** will cause (and require) this to happen automatically

Definition at line 114 of file `MotionManager.h`.

References `_MMaccID`, `MutexLock< MAX_ACCESS >::lock()`, and `MMlock`.

7.103.4.18 `MC_ID MotionManager::next (MC_ID cur) const [inline]`

returns the MC_ID of [MotionCommand](#) following the one that is passed

Definition at line 89 of file `MotionManager.h`.

References `cmdlist`, `MC_ID`, `ListMemBuf< CommandEntry, MAX_MOTIONS, MC_ID >::next()`, and `skip_ahead()`.

7.103.4.19 `MotionManager& MotionManager::operator= (const MotionManager &) [private]`

this shouldn't be called...

7.103.4.20 `MotionCommand* MotionManager::peekMotion (MC_ID mcid) [inline]`

allows access to a [MotionCommand](#) without checking it out **never** call a function based on this, only access member fields through it

Definition at line 98 of file `MotionManager.h`.

References `_MMaccID`, and `cmdlist`.

7.103.4.21 `MC_ID MotionManager::pop_free () [inline, protected]`

pulls an entry from `cmdlist`'s free section and returns its index

Definition at line 205 of file `MotionManager.h`.

References `cmdlist`, `MC_ID`, and `ListMemBuf< CommandEntry, MAX_MOTIONS, MC_ID >::new_front()`.

7.103.4.22 `void MotionManager::push_free (MC_ID a) [inline, protected]`

puts an entry back into cmdlist's free section

Definition at line 206 of file MotionManager.h.

References cmdlist, and ListMemBuf< CommandEntry, MAX_MOTIONS, MC_ID >::erase().

7.103.4.23 `void MotionManager::receivedMsg (const ONotifyEvent & event)`

LOCKS MotionManager This gets called by an [OObject](#) when it receives a message from one of the other OObject's MotionManagerComm Subject

Definition at line 469 of file MotionManager.cc.

References _MMaccID, EventBase::activateETID, MotionManagerMsg::addMotion, cmdlist, MotionManagerMsg::deleteMotion, erouter, func_begin(), func_end(), MotionManagerMsg::mc_id, MC_ID, EventBase::motmanEGID, EventRouter::postEvent(), and MotionManagerMsg::type.

7.103.4.24 `void MotionManager::release () [inline]`

releases a lock on the motion manager

Definition at line 116 of file MotionManager.h.

References MMlock, and MutexLock< MAX_ACCESS >::release().

7.103.4.25 `void MotionManager::removeMotion (MC_ID mcid)`

LOCKS MotionManager removes the specified [MotionCommand](#)

Test

do i do this right, or does it leak memory?

Definition at line 538 of file MotionManager.cc.

References _MMaccID, checkinMotion(), checkoutMotion(), cmdlist, EventBase::deactivateETID, erouter, func_begin(), func_end(), invalid_MC_ID, EventBase::motmanEGID, EventRouter::postEvent(), push_free(), MotionManagerMsg::setDelete(), and subjs.

7.103.4.26 `void MotionManager::setOutput (const MotionCommand * caller, unsigned int output, const OutputCmd cmd[NumFrames], const OutputPID & pid)`

LOCKS MotionManager Requests a value and PID be set for the specified output

Definition at line 172 of file MotionManager.cc.

References `ListMemBuf< OutputState, MAX_MOTIONS >::begin()`, `cmdstates`, `cmdSums`, `cur_cmd`, `ListMemBuf< OutputState, MAX_MOTIONS >::end()`, `func_`, `begin()`, `func_end()`, `MotionManagerMsg::getID()`, `getPriority()`, `invalid_MC_ID`, `kBackgroundPriority`, `ERS210Info::NumFrames`, `ERS210Info::NumOutputs`, and `ListMemBuf< OutputState, MAX_MOTIONS >::push_front()`.

7.103.4.27 `void MotionManager::setOutput (const MotionCommand * caller, unsigned int output, const OutputCmd & cmd, const OutputPID & pid)`

LOCKS MotionManager Requests a value and PID be set for the specified output

Definition at line 148 of file MotionManager.cc.

References `ListMemBuf< OutputState, MAX_MOTIONS >::begin()`, `cmdstates`, `cmdSums`, `cur_cmd`, `ListMemBuf< OutputState, MAX_MOTIONS >::end()`, `func_`, `begin()`, `func_end()`, `MotionManagerMsg::getID()`, `getPriority()`, `invalid_MC_ID`, `kBackgroundPriority`, `ERS210Info::NumFrames`, `ERS210Info::NumOutputs`, `ListMemBuf< OutputState, MAX_MOTIONS >::push_front()`, `OutputCmd::value`, and `OutputCmd::weight`.

7.103.4.28 `void MotionManager::setOutput (const MotionCommand * caller, unsigned int output, const OutputPID & pid)`

LOCKS MotionManager Requests a PID be set for the specified output, notice that this might be overruled by a higher priority motion

[Todo](#)

should be able to set background pid

Definition at line 128 of file MotionManager.cc.

References `ListMemBuf< OutputState, MAX_MOTIONS >::begin()`, `cmdstates`, `cur_`, `cmd`, `ListMemBuf< OutputState, MAX_MOTIONS >::end()`, `func_begin()`, `func_end()`, `MotionManagerMsg::getID()`, `getPriority()`, `invalid_MC_ID`, `kBackgroundPriority`, `ERS210Info::NumOutputs`, and `ListMemBuf< OutputState, MAX_MOTIONS >::push_front()`.

7.103.4.29 `void MotionManager::setOutput (const MotionCommand * caller, unsigned int output, const OutputCmd cmd[NumFrames])`

LOCKS MotionManager Requests a value be set for the specified output across frames

Definition at line 106 of file MotionManager.cc.

References `ListMemBuf< OutputState, MAX_MOTIONS >::begin()`, `cmdstates`, `cmdSums`, `cur_cmd`, `ListMemBuf< OutputState, MAX_MOTIONS >::end()`, `func_-begin()`, `func_end()`, `MotionManagerMsg::getID()`, `getPriority()`, `invalid_MC_ID`, `k-BackgroundPriority`, `ERS210Info::NumFrames`, `ERS210Info::NumOutputs`, and `ListMemBuf< OutputState, MAX_MOTIONS >::push_front()`.

7.103.4.30 `void MotionManager::setOutput (const MotionCommand * caller, unsigned int output, const OutputCmd & cmd, unsigned int frame)`

LOCKS MotionManager Requests a value be set for the specified output in the specified frame

Definition at line 85 of file MotionManager.cc.

References `ListMemBuf< OutputState, MAX_MOTIONS >::begin()`, `cmdstates`, `cmdSums`, `cur_cmd`, `ListMemBuf< OutputState, MAX_MOTIONS >::end()`, `func_-begin()`, `func_end()`, `MotionManagerMsg::getID()`, `getPriority()`, `invalid_MC_ID`, `k-BackgroundPriority`, `ERS210Info::NumOutputs`, `ListMemBuf< OutputState, MAX_MOTIONS >::push_front()`, `OutputCmd::value`, and `OutputCmd::weight`.

7.103.4.31 `void MotionManager::setOutput (const MotionCommand * caller, unsigned int output, const OutputCmd & cmd)`

LOCKS MotionManager Requests a value be set for the specified output, copies cmd across frames

Definition at line 63 of file MotionManager.cc.

References `ListMemBuf< OutputState, MAX_MOTIONS >::begin()`, `cmdstates`, `cmdSums`, `cur_cmd`, `ListMemBuf< OutputState, MAX_MOTIONS >::end()`, `func_-begin()`, `func_end()`, `MotionManagerMsg::getID()`, `getPriority()`, `invalid_MC_ID`, `k-BackgroundPriority`, `ERS210Info::NumFrames`, `ERS210Info::NumOutputs`, `ListMemBuf< OutputState, MAX_MOTIONS >::push_front()`, `OutputCmd::value`, and `OutputCmd::weight`.

7.103.4.32 `void MotionManager::setPID (unsigned int joint, const float pids[3])
[protected]`

LOCKS MotionManager you can call this to set the PID values directly (instead of

using a motion command) Be careful though

Note that we don't actually set the PIDs in the system here, we just queue them up. PID changes seem to be an expensive operation, so may only want to clear the queue at some reduced rate (although that's not actually currently implemented, so right now there's no real benefit until that's done)

Definition at line 563 of file MotionManager.cc.

References `ListMemBuf< PIDUpdate, NumPIDJoints >::begin()`, `ListMemBuf< PIDUpdate, NumPIDJoints >::end()`, `ListMemBuf< PIDUpdate, NumPIDJoints >::erase()`, `func_begin()`, `func_end()`, `ListMemBuf< PIDUpdate, NumPIDJoints >::next()`, `pidchanges`, `WorldState::pids`, `ListMemBuf< PIDUpdate, NumPIDJoints >::push_back()`, `state`, and `uint`.

7.103.4.33 `void MotionManager::setPriority (MC_ID mcid, float p)` [inline]

sets the priority level of a [MotionCommand](#)

Definition at line 83 of file MotionManager.h.

References `cmdlist`.

7.103.4.34 `unsigned int MotionManager::size () const` [inline]

returns the number of MotionCommands being managed

Definition at line 91 of file MotionManager.h.

References `cmdlist`, and `ListMemBuf< CommandEntry, MAX_MOTIONS, MC_ID >::size()`.

7.103.4.35 `MotionManager::MC_ID MotionManager::skip_ahead (MC_ID mcid) const` [protected]

during iteration, skips over motioncommands which are still in transit from on [OObject](#) to another

Definition at line 596 of file MotionManager.cc.

References `_MMaccID`, `cmdlist`, `ListMemBuf< CommandEntry, MAX_MOTIONS, MC_ID >::end()`, and `ListMemBuf< CommandEntry, MAX_MOTIONS, MC_ID >::next()`.

7.103.4.36 `bool MotionManager::trylock ()` [inline]

tries to get a lock without blocking

Definition at line 115 of file MotionManager.h.

References `_MMaccID`, `MMlock`, and `MutexLock< MAX_ACCESS >::try_lock()`.

7.103.4.37 **bool MotionManager::updatePIDs (OPrimitiveID *primIDs*[NumOutputs])**

call this when you want MotionManager to update modified PID values, returns true if changes made (only MotoObj should be calling this...)

Definition at line 386 of file MotionManager.cc.

References `ListMemBuf< PIDUpdate, NumPIDJoints >::empty()`, `ListMemBuf< PIDUpdate, NumPIDJoints >::front()`, `pidchanges`, `WorldState::pids`, `ListMemBuf< PIDUpdate, NumPIDJoints >::pop_front()`, `state`, and `uint`.

7.103.4.38 **void MotionManager::updateWorldState ()**

call this when you want MotionManager to set the [WorldState](#) to reflect what things should be for unsensed outputs (LEDs, ears) (only MotoObj should be calling this...)

Definition at line 366 of file MotionManager.cc.

References `ERS210Info::BinJointOffset`, `cmdSums`, `WorldState::ERS210Mask`, `WorldState::ERS220Mask`, `ERS210Info::LEDOffset`, `ERS210Info::NumBinJoints`, `ERS210Info::NumLEDs`, `ERS210Info::NumPIDJoints`, `WorldState::outputs`, `WorldState::robotDesign`, `state`, and `uint`.

7.103.5 Member Data Documentation

7.103.5.1 **unsigned int MotionManager::_MMaccID = -1U** [static, protected]

Stores the accessor for the current process.

Definition at line 12 of file MotionManager.cc.

7.103.5.2 **ListMemBuf<CommandEntry,MAX_MOTIONS,MC_ID> MotionManager::cmdlist** [protected]

the list where MotionCommands are stored, remember, we're in a shared memory region with different base addresses - no pointers!

Definition at line 201 of file MotionManager.h.

7.103.5.3 `OutputCmd MotionManager::cmds[NumOutputs]` [protected]

Holds the weighted values and total weight for the outputs of the last frame.

Definition at line 213 of file MotionManager.h.

7.103.5.4 `cmdstatelist_t MotionManager::cmdstates[NumOutputs]`
[protected]

requested positions by each of the MC's for each of the outputs

Definition at line 211 of file MotionManager.h.

7.103.5.5 `float MotionManager::cmdSums[NumOutputs]` [protected]

Holds the final values for the outputs of the last frame generated.

Definition at line 212 of file MotionManager.h.

7.103.5.6 `MC_ID MotionManager::cur_cmd` [protected]

MC_ID of the `MotionCommand` currently being updated by `getOutputs()`, or NULL if not in `getOutputs`. This is used by the `setOutput()`'s to tell which `MotionCommand` is calling.

Definition at line 202 of file MotionManager.h.

7.103.5.7 `const MC_ID MotionManager::invalid_MC_ID = (MC_ID)-1`
[static]

for errors and undefined stuff

Definition at line 66 of file MotionManager.h.

7.103.5.8 `const float MotionManager::kBackgroundPriority = 0` [static]

will only be expressed if *nothing* else is using that joint

Definition at line 15 of file MotionManager.cc.

7.103.5.9 `const float MotionManager::kEmergencyPriority = 100` [static]

for really important stuff, such as the emergency stop

Definition at line 19 of file MotionManager.cc.

7.103.5.10 `const float MotionManager::kHighPriority = 50 [static]`

for stuff that should over ride standard stuff

Definition at line 18 of file MotionManager.cc.

7.103.5.11 `const float MotionManager::kIgnoredPriority = -1 [static]`

won't be expressed, handy if you want to temporarily pause something

Definition at line 14 of file MotionManager.cc.

7.103.5.12 `const float MotionManager::kLowPriority = 5 [static]`

for stuff that's not background but lower than standard

Definition at line 16 of file MotionManager.cc.

7.103.5.13 `const float MotionManager::kStdPriority = 10 [static]`

for every-day commands

Definition at line 17 of file MotionManager.cc.

7.103.5.14 `const unsigned int MotionManager::MAX_ACCESS = 2
[static]`

This is the number of processes which will be accessing the MotionManager.

Probably just MainObject and MotionObject Isn't really a hard maximum, but should be actual expected, need to know when they're all connected

Definition at line 61 of file MotionManager.h.

7.103.5.15 `const unsigned int MotionManager::MAX_MOTIONS = 64
[static]`

This is the maximum number of Motions which can be managed, can probably be increased reasonably without trouble.

Definition at line 63 of file MotionManager.h.

7.103.5.16 [MutexLock<MAX_ACCESS>](#) [MotionManager::MMlock](#)
[protected]

The main lock for the class.

Definition at line 208 of file MotionManager.h.

7.103.5.17 [accID.t](#) [MotionManager::numAcc](#) [protected]

The number of accessors who have registered with [InitAccess\(\)](#).

Definition at line 216 of file MotionManager.h.

7.103.5.18 [ListMemBuf<PIDUpdate,NumPIDJoints>](#)
[MotionManager::pidchanges](#) [protected]

stores PID updates, up to one per joint (if same is set more than once, it's just overwrites previous update)

Definition at line 168 of file MotionManager.h.

7.103.5.19 [OSubject*](#) [MotionManager::subs](#)[[MAX_ACCESS](#)]
[protected]

The OSubject for each process (accessor) on which it should be broadcast when a command is added.

Definition at line 217 of file MotionManager.h.

The documentation for this class was generated from the following files:

- [MotionManager.h](#)
- [MotionManager.cc](#)

7.104 MotionManager::CommandEntry Struct Reference

```
#include <MotionManager.h>
```

7.104.1 Detailed Description

All the information we need to maintain about a [MotionCommand](#).

Definition at line 180 of file MotionManager.h.

Public Member Functions

- [CommandEntry](#) ()
Constructor, sets everything to basics.

Public Attributes

- [MotionCommand](#) * [baseaddr](#) [[MAX_ACCESS](#)]
for each accessor, the base address of the motion command
- [RCRegion](#) * [rcr](#) [[MAX_ACCESS](#)]
for each accessor the shared memory region that holds the motion command
- [accID_t](#) [lastAccessor](#)
the ID of the last accessor to touch the command (which implies if it wants to touch this again, we don't have to convert again)
- [MutexLock](#)< [MAX_ACCESS](#) > [lock](#)
a lock to maintain mutual exclusion
- float [priority](#)
MotionCommand's priority level.

Private Member Functions

- [CommandEntry](#) (const [CommandEntry](#) &)
this shouldn't be called...

- [CommandEntry](#) & [operator=](#) (const [CommandEntry](#) &)

this shouldn't be called...

7.104.2 Constructor & Destructor Documentation

7.104.2.1 MotionManager::CommandEntry::CommandEntry () [inline]

Constructor, sets everything to basics.

Definition at line 182 of file MotionManager.h.

References [baseaddr](#)s, [lastAccessor](#), [priority](#), and [rcr](#).

7.104.2.2 MotionManager::CommandEntry::CommandEntry (const [CommandEntry](#) &) [private]

this shouldn't be called...

7.104.3 Member Function Documentation

7.104.3.1 [CommandEntry](#) & MotionManager::CommandEntry::operator= (const [CommandEntry](#) &) [private]

this shouldn't be called...

7.104.4 Member Data Documentation

7.104.4.1 [MotionCommand*](#) [MotionManager::CommandEntry::baseaddr](#)[[MAX_ACCESS](#)]

for each accessor, the base address of the motion command

Definition at line 190 of file MotionManager.h.

7.104.4.2 [accID_t](#) [MotionManager::CommandEntry::lastAccessor](#)

the ID of the last accessor to touch the command (which implies if it wants to touch this again, we don't have to convert again)

Definition at line 194 of file MotionManager.h.

7.104.4.3 [MutexLock<MAX_ACCESS> MotionManager::CommandEntry::lock](#)

a lock to maintain mutual exclusion

Definition at line 195 of file MotionManager.h.

7.104.4.4 [float MotionManager::CommandEntry::priority](#)

MotionCommand's priority level.

Definition at line 196 of file MotionManager.h.

7.104.4.5 [RCRegion* MotionManager::CommandEntry::rcr\[MAX_ACCESS\]](#)

for each accessor the shared memory region that holds the motion command

Definition at line 192 of file MotionManager.h.

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

- [MotionManager.h](#)

7.105 MotionManager::OutputState Class Reference

```
#include <MotionManager.h>
```

7.105.1 Detailed Description

holds the full requested value of an output

Definition at line 128 of file MotionManager.h.

Public Member Functions

Constructors

Constructor

- [OutputState](#) ()
- [OutputState](#) (unsigned int out, float pri, [MC_ID](#) mc, const [OutputCmd](#) cmds[NumFrames])
- [OutputState](#) (unsigned int out, float pri, [MC_ID](#) mc, const [OutputCmd](#) &cmd)
- [OutputState](#) (unsigned int out, float pri, [MC_ID](#) mc, const [OutputCmd](#) &cmd, unsigned int frame)
- [OutputState](#) (unsigned int out, float pri, [MC_ID](#) mc, const [OutputPID](#) &p)
- [OutputState](#) (unsigned int out, float pri, [MC_ID](#) mc, const [OutputCmd](#) cmds[NumFrames], const [OutputPID](#) &p)
- [OutputState](#) (unsigned int out, float pri, [MC_ID](#) mc, const [OutputCmd](#) &cmd, const [OutputPID](#) &p)

Public Attributes

- float [priority](#)
priority level
- [MC_ID](#) [mcid](#)
MC_ID of requester.
- [OutputCmd](#) [frames](#) [NumFrames]
values of output planned ahead
- [OutputPID](#) [pid](#)
pid of output

7.105.2 Constructor & Destructor Documentation

7.105.2.1 MotionManager::OutputState::OutputState ()

Definition at line 608 of file MotionManager.cc.

7.105.2.2 MotionManager::OutputState::OutputState (unsigned int *out*, float *pri*, **MC_ID** *mc*, const **OutputCmd** *cmds*[NumFrames])

Definition at line 611 of file MotionManager.cc.

References ERS210Info::DefaultPIDs, frames, and ERS210Info::NumFrames.

7.105.2.3 MotionManager::OutputState::OutputState (unsigned int *out*, float *pri*, **MC_ID** *mc*, const **OutputCmd** & *cmd*)

Definition at line 617 of file MotionManager.cc.

References ERS210Info::DefaultPIDs, frames, and ERS210Info::NumFrames.

7.105.2.4 MotionManager::OutputState::OutputState (unsigned int *out*, float *pri*, **MC_ID** *mc*, const **OutputCmd** & *cmd*, unsigned int *frame*)

Definition at line 623 of file MotionManager.cc.

References ERS210Info::DefaultPIDs, and frames.

7.105.2.5 MotionManager::OutputState::OutputState (unsigned int *out*, float *pri*, **MC_ID** *mc*, const **OutputPID** & *p*)

Definition at line 628 of file MotionManager.cc.

7.105.2.6 MotionManager::OutputState::OutputState (unsigned int *out*, float *pri*, **MC_ID** *mc*, const **OutputCmd** *cmds*[NumFrames], const **OutputPID** & *p*)

Definition at line 631 of file MotionManager.cc.

References frames, and ERS210Info::NumFrames.

7.105.2.7 MotionManager::OutputState::OutputState (unsigned int *out*, float *pri*, **MC_ID** *mc*, const **OutputCmd** & *cmd*, const **OutputPID** & *p*)

Definition at line 637 of file MotionManager.cc.

References frames, and ERS210Info::NumFrames.

7.105.3 Member Data Documentation

7.105.3.1 [OutputCmd MotionManager::OutputState::frames\[NumFrames\]](#)

values of output planned ahead

Definition at line 142 of file MotionManager.h.

7.105.3.2 [MC_ID MotionManager::OutputState::mcid](#)

MC_ID of requester.

Definition at line 141 of file MotionManager.h.

7.105.3.3 [OutputPID MotionManager::OutputState::pid](#)

pid of output

Definition at line 143 of file MotionManager.h.

7.105.3.4 [float MotionManager::OutputState::priority](#)

priority level

Definition at line 140 of file MotionManager.h.

The documentation for this class was generated from the following files:

- [MotionManager.h](#)
- [MotionManager.cc](#)

7.106 MotionManager::PIDUpdate Struct Reference

```
#include <MotionManager.h>
```

7.106.1 Detailed Description

used to request pids for a given joint

Definition at line 157 of file MotionManager.h.

Public Member Functions

- [PIDUpdate](#) ()
constructor
- [PIDUpdate](#) (unsigned int j, const float p[3])
constructor

Public Attributes

- unsigned int [joint](#)
the joint ID (see [RobotInfo.h](#) for offset values)
- float [pids](#) [3]
the PID values to use (see [::Pid](#))

7.106.2 Constructor & Destructor Documentation

7.106.2.1 MotionManager::PIDUpdate::PIDUpdate () [inline]

constructor

Definition at line 159 of file MotionManager.h.

References [joint](#).

7.106.2.2 MotionManager::PIDUpdate::PIDUpdate (unsigned int j, const float p[3]) [inline]

constructor

Definition at line 161 of file MotionManager.h.

References [joint](#), and [pids](#).

7.106.3 Member Data Documentation

7.106.3.1 unsigned int [MotionManager::PIDUpdate::joint](#)

the joint ID (see [RobotInfo.h](#) for offset values)

Definition at line 165 of file MotionManager.h.

7.106.3.2 float [MotionManager::PIDUpdate::pids](#)[3]

the PID values to use (see [::Pid](#))

Definition at line 166 of file MotionManager.h.

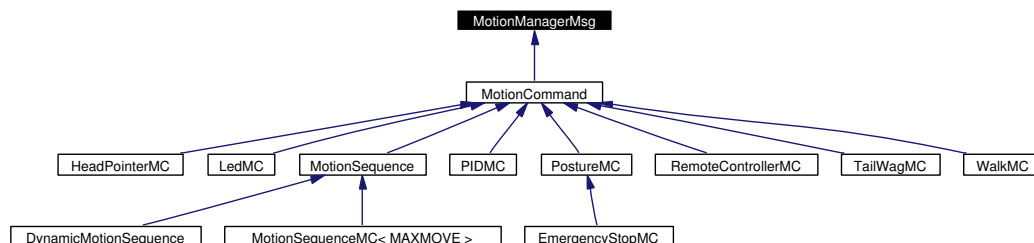
The documentation for this struct was generated from the following file:

- [MotionManager.h](#)

7.107 MotionManagerMsg Struct Reference

```
#include <MotionManagerMsg.h>
```

Inheritance diagram for MotionManagerMsg:



7.107.1 Detailed Description

A small header that precedes data sent by [MotionManager](#) between processes.

Definition at line 6 of file MotionManagerMsg.h.

Public Types

- typedef unsigned short [MC_ID](#)
the type to use when referring to [MotionCommand](#) ID's

Public Member Functions

- [MotionManagerMsg](#) ()
constructor
- virtual [~MotionManagerMsg](#) ()
virtual destructor
- [MC_ID](#) getID () const
Accessor for the id number, set by [MotionManager::addMotion\(\)](#).

Private Types

- enum [MsgType](#) { [addMotion](#), [deleteMotion](#), [unknown](#) }
Denotes what type of message this is.

Private Member Functions

- void [setAdd](#) ([MC_ID](#) id)
Sets up the header as an add motion message.
- void [setDelete](#) ([MC_ID](#) id)
Sets up the header as an erase motion message.

Private Attributes

- enum [MotionManagerMsg::MsgType](#) type
Denotes what type of message this is.
- [MC_ID](#) [mc_id](#)
The id of the [MotionCommand](#) this is in reference to.

Friends

- class [MotionManager](#)

7.107.2 Member Typedef Documentation

7.107.2.1 typedef unsigned short [MotionManagerMsg::MC_ID](#)

the type to use when referring to [MotionCommand](#) ID's

Definition at line 8 of file [MotionManagerMsg.h](#).

7.107.3 Member Enumeration Documentation

7.107.3.1 enum [MotionManagerMsg::MsgType](#) [private]

Denotes what type of message this is.

Enumeration values:**addMotion****deleteMotion****unknown**

Definition at line 24 of file MotionManagerMsg.h.

7.107.4 Constructor & Destructor Documentation

7.107.4.1 MotionManagerMsg::MotionManagerMsg () [inline]

constructor

Definition at line 11 of file MotionManagerMsg.h.

References MC_ID, mc_id, type, and unknown.

7.107.4.2 virtual MotionManagerMsg::~MotionManagerMsg () [inline, virtual]

virtual destructor

doesn't do anything, but don't remove it, otherwise this would no longer be a virtual base class

Definition at line 15 of file MotionManagerMsg.h.

7.107.5 Member Function Documentation

7.107.5.1 MC_ID MotionManagerMsg::getID () const [inline]

Accessor for the id number, set by [MotionManager::addMotion\(\)](#).

Definition at line 18 of file MotionManagerMsg.h.

References mc_id, and MC_ID.

7.107.5.2 void MotionManagerMsg::setAdd (MC_ID id) [inline, private]

Sets up the header as an add motion message.

Definition at line 30 of file MotionManagerMsg.h.

References addMotion, mc_id, and type.

7.107.5.3 `void MotionManagerMsg::setDelete (MC_ID id)` [`inline`,
 `private`]

Sets up the header as an erase motion message.

Definition at line 36 of file MotionManagerMsg.h.

References `deleteMotion`, `mc_id`, and `type`.

7.107.6 Friends And Related Function Documentation

7.107.6.1 `friend class MotionManager` [`friend`]

Definition at line 21 of file MotionManagerMsg.h.

7.107.7 Member Data Documentation

7.107.7.1 `MC_ID MotionManagerMsg::mc_id` [`private`]

The id of the [MotionCommand](#) this is in reference to.

Definition at line 27 of file MotionManagerMsg.h.

7.107.7.2 `enum MotionManagerMsg::MsgType MotionManagerMsg::type`
 [`private`]

Denotes what type of message this is.

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

- [MotionManagerMsg.h](#)

7.108 MotionRequest Struct Reference

```
#include <WorldModel2.h>
```

7.108.1 Detailed Description

Structure for containing motion requests.

Definition at line 67 of file WorldModel2.h.

Public Types

- enum { [LOOK_AT](#), [LOOK_DOWN_AT](#), [GO_TO](#) }

Is this motion request from the DM, the HM, or the GM?

Public Attributes

- enum MotionRequest:: { ... } [type](#)

Is this motion request from the DM, the HM, or the GM?

7.108.2 Member Enumeration Documentation

7.108.2.1 anonymous enum

Is this motion request from the DM, the HM, or the GM?

Enumeration values:

LOOK_AT

LOOK_DOWN_AT

GO_TO

Definition at line 69 of file WorldModel2.h.

7.108.3 Member Data Documentation

7.108.3.1 double [MotionRequest::altitude](#)

height relative to ? TSS_TODO

Definition at line 76 of file WorldModel2.h.

7.108.3.2 struct { ... } [MotionRequest::azalt](#)

Used for LOOK_AT, the requests made by the DM.

7.108.3.3 double [MotionRequest::azimuth](#)

angle from straight ahead (bearing)

Definition at line 75 of file WorldModel2.h.

7.108.3.4 enum { ... } [MotionRequest::type](#)

Is this motion request from the DM, the HM, or the GM?

7.108.3.5 double [MotionRequest::x](#)

x

Definition at line 81 of file WorldModel2.h.

7.108.3.6 struct { ... } [MotionRequest::xy](#)

Used for LOOK_DOWN_AT and GO_TO, the requests made by the HM and the GM.

7.108.3.7 double [MotionRequest::y](#)

y

Definition at line 82 of file WorldModel2.h.

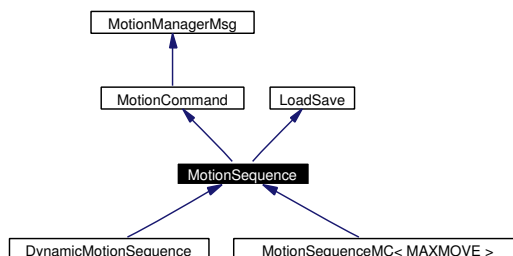
The documentation for this struct was generated from the following file:

- [WorldModel2.h](#)

7.109 MotionSequence Class Reference

```
#include <MotionSequenceMC.h>
```

Inheritance diagram for MotionSequence:



7.109.1 Detailed Description

A handy little (or not so little) class for switching between a sequence of postures.

Outputs are handled independently. It's easy to add keyframes which modify all of the outputs, but since each output is tracked individually, OutputCmd's with 0 weight can be used to not affect other motions. For instance, pan the head left to right while moving the right leg up and down several times, you won't have to specify the position of the head in its motion at each of the leg motion keyframes.

Be aware that the 0 time frame will be replaced on a call to [play\(\)](#) with the current body posture. However, this only applies to outputs which have a non-zero weighted frame defined at some point. The weights, of the 0 time frame will remain unchanged. These weights are initially set to 0, so that it's possible to 'fade in' the first frame of the motion sequence from wherever the body happens to be (or already doing)

To fade out at the end, set a frame with 0 weight for everything. Otherwise it will simply die suddenly. (Currently, will still jerk between priority level shifts - can only fade within a priority level)

Currently, MotionSequence's are intended mainly for building, not editing. It's easy to add keyframes, but hard/impossible to delete them.

The MotionSequence base class is an abstract class so that you can create memory efficient motion sequences and simply refer to them by the common base class instead of having to worry about the actual size allocated in the template, [MotionSequenceMC](#).

See also:

[MotionSequence::SizeSmall](#), [MotionSequence::SizeMedium](#), [MotionSequence::SizeLarge](#), [MotionSequence::SizeXLarge](#),

The file format used is as follows: ('<' and '>' are not meant literally)

```
*      First line: #MSq
* Zero or more of: delay <time-delta>           (moves playhead forward, in milliseconds)
*                  or: settime <time>           (sets play time to specified value, in ms)
*                  or: <outputname> <value> [<weight>] (sets the specified output to the value - assumes 1 f
*                  or: load <filename>           (file is a posture, sets position)
*                  or: overlay <filename>        (file can be a posture or another motion sequence)
*                  or: degrees                   (following <value>s will be interpreted as degrees [d
*                  or: radians                   (following <value>s will be interpreted as radians)
*      Last line: #END
*
```

After loading a motion sequence, the playtime is left at the end. This is to make it easy to append/overlay motion sequences

Lines beginning with '#' are ignored. Output names are defined in [RobotInfo.h](#), RobotInfo::outputNames.

Definition at line 62 of file MotionSequenceMC.h.

Public Member Functions

- [MotionSequence](#) ()
constructor, will start playing immediately
- virtual [~MotionSequence](#) ()
destructor

Inherited from MotionCommand

- virtual int [updateOutputs](#) ()
is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)
- virtual int [isDirty](#) ()
not used by [MotionManager](#) at the moment, but could be used to reduce recomputation, and you may find it useful
- virtual int [isAlive](#) ()
used to prune "dead" motions from the [MotionManager](#)

LoadSave related

- virtual unsigned int [getBinSize](#) () const

inherited, returns the size used to save the sequence

- virtual unsigned int [LoadBuffer](#) (const char buf[], unsigned int len)
inherited, doesn't clear before loading - call clear yourself if you want to reset, otherwise it will overlay. Leaves playtime at end of load.
- virtual unsigned int [SaveBuffer](#) (char buf[], unsigned int len) const
inherited, saves the motion sequence - will save a flat file - doesn't remember references to other files which were loaded
- void [setSaveDegrees](#) ()
will store angles as degrees on future saves
- bool [isSaveDegrees](#) () const
returns true if will store angles as degrees on future saves
- void [setSaveRadians](#) ()
will store angles as radians on future saves
- bool [isSaveRadians](#) () const
returns true if will store angles as degrees on future saves

Sequence Construction

- virtual void [clear](#) ()=0
empties out the sequence (constant time operation - faster than a series of pops)
- void [setPlayTime](#) (unsigned int x)
set the time for both playback and editing (in milliseconds)
- void [setOutputCmd](#) (unsigned int i, const [OutputCmd](#) &cmd)
will insert a keyframe for the given output, or change an existing one
- const [OutputCmd](#) & [getOutputCmd](#) (unsigned int i)
gets the value of output i at the playhead
- void [setPose](#) (const [PostureEngine](#) &pose)
calls setOutputCmd on each of the OutputCmds in pose
- void [overlayPose](#) (const [PostureEngine](#) &pose)
calls setOutputCmd on non-zero weighted OutputCmds in pose
- void [compress](#) ()
compresses the sequence by eliminating sequences of moves which are identical
- virtual unsigned int [getMaxFrames](#) () const=0

returns the maximum number of key frames (Move's) which can be stored, determined by the instantiating MotionSequenceMC's template parameter

- virtual unsigned int [getUsedFrames](#) () const=0
returns the number of used key frames (Move's) which have been stored by the instantiation MotionSequence subclass
- void [makeSafe](#) (const float vels[NumOutputs], float margin)
*will insert time into the motion where needed to keep the joint velocities at or below the speeds given in vels * margin*

Playback Control

- bool [isPlaying](#) ()
returns true if currently playing
- void [play](#) ()
restarts playback from beginning
- void [pause](#) ()
pauses playback until another call to [play\(\)](#) or [resume\(\)](#)
- void [resume](#) ()
begins playback from the current playtime
- unsigned int [getPlayTime](#) () const
returns the current position of the playback (in milliseconds), see [setPlayTime\(\)](#)
- unsigned int [getEndTime](#) () const
returns the length of the motion sequence (in milliseconds)
- void [setPlaySpeed](#) (float x)
sets the playback speed (e.g. 1=regular, 0.5=half speed, -1=backwards)
- float [getPlaySpeed](#) () const
returns the playback speed

Static Public Attributes

Template Sizes

To avoid code bloat if there are a large number of different sized MotionSequences, use these sizes where possible.

- const unsigned int [SizeTiny](#) = NumOutputs*2

Tiny, but enough to handle a transition into a full-body pose.

- const unsigned int [SizeSmall](#) = NumOutputs*3
Small, but still big enough to handle most of the included MS's (2 full-body frames ~ around 1KB).
- const unsigned int [SizeMedium](#) = NumOutputs*6
Medium (5 full body frames ~ est 4KB).
- const unsigned int [SizeLarge](#) = NumOutputs*11
Large (10 full body frames ~ est 8KB).
- const unsigned int [SizeXLarge](#) = NumOutputs*26
eXtra Large (25 full body frames ~ est 16KB)

Protected Types

- typedef unsigned short [Move_idx_t](#)
type for indexes to move structures in #moves

Protected Member Functions

- virtual [Move](#) & [getKeyFrame](#) ([Move_idx_t](#) x)=0
returns the [Move](#) struct corresponding to x in the subclass's actual data structure
- virtual const [Move](#) & [getKeyFrame](#) ([Move_idx_t](#) x) const=0
returns the [Move](#) struct corresponding to x in the subclass's actual data structure
- virtual [Move_idx_t](#) [newKeyFrame](#) ()=0
causes subclass to create a new [Move](#) structure, returns its index
- virtual void [eraseKeyFrame](#) ([Move_idx_t](#) x)=0
causes subclass to mark the corresponding [Move](#) structure as free
- void [calcOutput](#) ([OutputCmd](#) &ans, unsigned int t, const [Move](#) &prev, const [Move](#) &next) const
Does the actual calculation of position information. Perhaps replace with a Bezier or spline or something?
- virtual void [setRange](#) (unsigned int t, [Move_idx_t](#) &prev, [Move_idx_t](#) &next) const=0

Sets prev and next to the appropriate values for the given time and output index.

- unsigned int [setNextFrameTime](#) ([Move_idx_t](#) p[NumOutputs], [Move_idx_t](#) n[NumOutputs]) const
sets playtime to next time for which any output has a keyframe, -1 if none exists

Static Protected Member Functions

- bool [ChkAdvance](#) (int res, const char **buf, unsigned int *len, const char *msg)
used by [LoadBuffer\(\)](#)/[SaveBuffer\(\)](#), checks to see if the amount read/written (res) is nonzero, increments buf, decrements len, or displays msg if is zero
- bool [ChkAdvance](#) (int res, const char **buf, unsigned int *len, const char *msg, int arg1)
used by [LoadBuffer\(\)](#)/[SaveBuffer\(\)](#), checks to see if the amount read/written (res) is nonzero, increments buf, decrements len, or displays msg with arg1 if is zero
- unsigned int [readWord](#) (const char buf[], const char *const buflen, char word[], const unsigned int wordlen)
reads a line from a file, parsing it into variables, returns ending position
- unsigned int [getOutputIndex](#) (const char name[], unsigned int i)
returns the index for the output named in the string or NumOutputs if not found, begins search through RobotInfo::outputName's at index i

Protected Attributes

- [Move_idx_t](#) starts [NumOutputs]
the beginning frame for each output animation
- [Move_idx_t](#) prevs [NumOutputs]
the previous frame (the starttime for this frame will always be less than or equal to playtime)
- [Move_idx_t](#) nexts [NumOutputs]
the upcoming frame (the starttime for this frame will always be greater than playtime)
- [OutputCmd](#) curs [NumOutputs]
merely a cache of current values (if computed, see [curstamps](#))

- unsigned int `curstamps` [NumOutputs]
timestamp of corresponding value in `curs`
- unsigned int `playtime`
the current time of playback, 0 is start of sequence
- unsigned int `lasttime`
the time of the last update
- unsigned int `endtime`
max of #moves's `Move::starttime`'s
- float `playspeed`
multiplies the difference between current time and starttime, negative will cause play backwards
- bool `playing`
true if playing, false if paused
- float `loadSaveMode`
1 to use radians, $M_PI/180$ for degrees during a save

Static Protected Attributes

- `Move_idx_t invalid_move = -1U`
used to mark the ends of the `Move` linked lists

7.109.2 Member Typedef Documentation

7.109.2.1 typedef unsigned short `MotionSequence::Move_idx_t` [protected]

type for indexes to move structures in #moves

Definition at line 120 of file MotionSequenceMC.h.

7.109.3 Constructor & Destructor Documentation

7.109.3.1 `MotionSequence::MotionSequence ()` [inline]

constructor, will start playing immediately

Definition at line 65 of file MotionSequenceMC.h.

References `endtime`, `lasttime`, `loadSaveMode`, `playing`, `playspeed`, and `playtime`.

7.109.3.2 `virtual MotionSequence::~~MotionSequence () [inline, virtual]`

destructor

Definition at line 67 of file MotionSequenceMC.h.

7.109.4 Member Function Documentation

7.109.4.1 `void MotionSequence::calcOutput (OutputCmd & ans, unsigned int t, const Move & prev, const Move & next) const [inline, protected]`

Does the actual calculation of position information. Perhaps replace with a Bezier or spline or something?

Definition at line 153 of file MotionSequenceMC.h.

References `MotionSequence::Move::cmd`, `OutputCmd::set()`, and `MotionSequence::Move::starttime`.

7.109.4.2 `bool MotionSequence::ChkAdvance (int res, const char ** buf, unsigned int * len, const char * msg, int arg1) [static, protected]`

used by `LoadBuffer()/SaveBuffer()`, checks to see if the amount read/written (*res*) is nonzero, increments *buf*, decrements *len*, or displays *msg* with *arg1* if *is* zero

Definition at line 361 of file MotionSequenceMC.cc.

7.109.4.3 `bool MotionSequence::ChkAdvance (int res, const char ** buf, unsigned int * len, const char * msg) [static, protected]`

used by `LoadBuffer()/SaveBuffer()`, checks to see if the amount read/written (*res*) is nonzero, increments *buf*, decrements *len*, or displays *msg* if *is* zero

Definition at line 350 of file MotionSequenceMC.cc.

7.109.4.4 `virtual void MotionSequence::clear () [pure virtual]`

empties out the sequence (constant time operation - faster than a series of pops)

Implemented in [DynamicMotionSequence](#), and [MotionSequenceMC< MAXMOVE >](#).

7.109.4.5 void MotionSequence::compress ()

compresses the sequence by eliminating sequences of moves which are identical

Definition at line 272 of file MotionSequenceMC.cc.

References [calcOutput\(\)](#), [MotionSequence::Move::cmd](#), [eraseKeyFrame\(\)](#), [getKeyFrame\(\)](#), [Move_idx_t](#), [MotionSequence::Move::next](#), [ERS210Info::NumOutputs](#), [MotionSequence::Move::prev](#), [starts](#), [MotionSequence::Move::starttime](#), and [OutputCmd::weight](#).

7.109.4.6 virtual void MotionSequence::eraseKeyFrame ([Move_idx_t](#) x) [protected, pure virtual]

causes subclass to mark the corresponding [Move](#) structure as free

Implemented in [DynamicMotionSequence](#), and [MotionSequenceMC< MAXMOVE >](#).

7.109.4.7 unsigned int MotionSequence::getBinSize () const [virtual]

inherited, returns the size used to save the sequence

Implements [LoadSave](#).

Definition at line 39 of file MotionSequenceMC.cc.

References [MotionSequence::Move::cmd](#), [getKeyFrame\(\)](#), [isSaveRadians\(\)](#), [loadSaveMode](#), [Move_idx_t](#), [MotionSequence::Move::next](#), [ERS210Info::NumOutputs](#), [ERS210Info::outputNames](#), [setNextFrameTime\(\)](#), [starts](#), [MotionSequence::Move::starttime](#), and [OutputCmd::weight](#).

7.109.4.8 unsigned int MotionSequence::getEndTime () const [inline]

returns the length of the motion sequence (in milliseconds)

Definition at line 113 of file MotionSequenceMC.h.

References [endtime](#).

7.109.4.9 `virtual const Move& MotionSequence::getKeyFrame (Move_idx_t x)`
`const [protected, pure virtual]`

returns the [Move](#) struct corresponding to *x* in the subclass's actual data structure

Implemented in [DynamicMotionSequence](#), and [MotionSequenceMC< MAXMOVE](#)
[>](#).

7.109.4.10 `virtual Move& MotionSequence::getKeyFrame (Move_idx_t x)`
`[protected, pure virtual]`

returns the [Move](#) struct corresponding to *x* in the subclass's actual data structure

Implemented in [DynamicMotionSequence](#), and [MotionSequenceMC< MAXMOVE](#)
[>](#).

7.109.4.11 `virtual unsigned int MotionSequence::getMaxFrames () const`
`[pure virtual]`

returns the maximum number of key frames (Move's) which can be stored, determined by the instantiating MotionSequenceMC's template parameter

Implemented in [DynamicMotionSequence](#), and [MotionSequenceMC< MAXMOVE](#)
[>](#).

7.109.4.12 `const OutputCmd & MotionSequence::getOutputCmd (unsigned int i)`

gets the value of output *i* at the playhead

Definition at line 28 of file MotionSequenceMC.cc.

References [calcOutput\(\)](#), [curs](#), [curstamps](#), [getKeyFrame\(\)](#), [invalid_move](#), [nexts](#), [play-time](#), [prevs](#), and [OutputCmd::unset\(\)](#).

7.109.4.13 `unsigned int MotionSequence::getOutputIndex (const char name[], unsigned int i) [static, protected]`

returns the index for the output named in the string or NumOutputs if not found, begins search through RobotInfo::outputName's at index *i*

Definition at line 399 of file MotionSequenceMC.cc.

References [ERS210Info::NumOutputs](#), and [ERS210Info::outputNames](#).

7.109.4.14 `float MotionSequence::getPlaySpeed () const` `[inline]`

returns the playback speed

Definition at line 115 of file MotionSequenceMC.h.

References `playspeed`.

7.109.4.15 `unsigned int MotionSequence::getPlayTime () const` `[inline]`

returns the current position of the playback (in milliseconds), see [setPlayTime\(\)](#)

Definition at line 112 of file MotionSequenceMC.h.

References `playtime`.

7.109.4.16 `virtual unsigned int MotionSequence::getUsedFrames () const`
`[pure virtual]`

returns the number of used key frames (Move's) which have been stored by the instantiation MotionSequence subclass

Implemented in [DynamicMotionSequence](#), and [MotionSequenceMC< MAXMOVE >](#).

7.109.4.17 `virtual int MotionSequence::isAlive ()` `[inline, virtual]`

used to prune "dead" motions from the [MotionManager](#)

note that a motion could be "paused" or inactive and therefore not dirty, but still alive, biding its time to "strike" ;)

Returns:

zero if the motion is still processing, non-zero otherwise

Implements [MotionCommand](#).

Definition at line 81 of file MotionSequenceMC.h.

References `playspeed`.

7.109.4.18 `virtual int MotionSequence::isDirty ()` `[inline, virtual]`

not used by [MotionManager](#) at the moment, but could be used to reduce recomputation, and you may find it useful

Returns:

zero if none of the commands have changed since last `getJointCmd()`, else non-zero

Implements [MotionCommand](#).

Definition at line 80 of file `MotionSequenceMC.h`.

References `isPlaying()`.

7.109.4.19 bool MotionSequence::isPlaying () [inline]

returns true if currently playing

Definition at line 108 of file `MotionSequenceMC.h`.

References `isAlive()`, and `playing`.

7.109.4.20 bool MotionSequence::isSaveDegrees () const [inline]

returns true if will store angles as degrees on future saves

Definition at line 89 of file `MotionSequenceMC.h`.

References `loadSaveMode`.

7.109.4.21 bool MotionSequence::isSaveRadians () const [inline]

returns true if will store angles as degrees on future saves

Definition at line 91 of file `MotionSequenceMC.h`.

References `loadSaveMode`.

7.109.4.22 unsigned int MotionSequence::LoadBuffer (const char *buf* [], unsigned int *len*) [virtual]

inherited, doesn't clear before loading - call clear yourself if you want to reset, otherwise it will overlay. Leaves playtime at end of load.

Implements [LoadSave](#).

Definition at line 68 of file `MotionSequenceMC.cc`.

References `ChkAdvance()`, `getOutputIndex()`, `LoadSave::LoadFile()`, `loadSaveMode`, `ERS210Info::NumOutputs`, `overlayPose()`, `playtime`, `readWord()`, `setOutputCmd()`, `setPlayTime()`, `setPose()`, `setSaveDegrees()`, and `setSaveRadians()`.

7.109.4.23 void MotionSequence::makeSafe (const float *vels*[NumOutputs], float *margin*)

will insert time into the motion where needed to keep the joint velocities at or below the speeds given in *vels* * *margin*

Definition at line 299 of file MotionSequenceMC.cc.

References MotionSequence::Move::cmd, getKeyFrame(), Move_idx_t, MotionSequence::Move::next, ERS210Info::NumOutputs, setNextFrameTime(), starts, MotionSequence::Move::starttime, and OutputCmd::weight.

7.109.4.24 virtual [Move_idx_t](#) MotionSequence::newKeyFrame () [protected, pure virtual]

causes subclass to create a new [Move](#) structure, returns its index

Implemented in [DynamicMotionSequence](#), and [MotionSequenceMC< MAXMOVE >](#).

7.109.4.25 void MotionSequence::overlayPose (const [PostureEngine](#) & *pose*)

calls setOutputCmd on non-zero weighted OutputCmds in *pose*

Definition at line 266 of file MotionSequenceMC.cc.

References PostureEngine::getOutputCmd(), ERS210Info::NumOutputs, setOutputCmd(), and OutputCmd::weight.

7.109.4.26 void MotionSequence::pause () [inline]

pauses playback until another call to [play\(\)](#) or [resume\(\)](#)

Definition at line 110 of file MotionSequenceMC.h.

References playing.

7.109.4.27 void MotionSequence::play ()

restarts playback from beginning

Definition at line 327 of file MotionSequenceMC.cc.

References endtime, playspeed, resume(), and setPlayTime().

7.109.4.28 `unsigned int MotionSequence::readWord (const char buf[], const char *const buflen, char word[], const unsigned int wordlen)`
`[static, protected]`

reads a line from a file, parsing it into variables, returns ending position

Definition at line 383 of file MotionSequenceMC.cc.

7.109.4.29 `void MotionSequence::resume ()`

begins playback from the current playtime

Definition at line 335 of file MotionSequenceMC.cc.

References `MotionSequence::Move::cmd`, `get_time()`, `getKeyFrame()`, `lasttime`, `Move_idx_t`, `MotionSequence::Move::next`, `ERS210Info::NumOutputs`, `WorldState::outputs`, `playing`, `starts`, `state`, `OutputCmd::value`, and `OutputCmd::weight`.

7.109.4.30 `unsigned int MotionSequence::SaveBuffer (char buf[], unsigned int len) const` `[virtual]`

inherited, saves the motion sequence - will save a flat file - doesn't remember references to other files which were loaded

Implements [LoadSave](#).

Definition at line 171 of file MotionSequenceMC.cc.

References `ChkAdvance()`, `MotionSequence::Move::cmd`, `getKeyFrame()`, `isSaveRadians()`, `loadSaveMode`, `Move_idx_t`, `MotionSequence::Move::next`, `ERS210Info::NumOutputs`, `ERS210Info::outputNames`, `setNextFrameTime()`, `starts`, `MotionSequence::Move::starttime`, and `OutputCmd::weight`.

7.109.4.31 `unsigned int MotionSequence::setNextFrameTime (Move_idx_t p[NumOutputs], Move_idx_t n[NumOutputs]) const`
`[protected]`

sets playtime to next time for which any output has a keyframe, -1 if none exists

Definition at line 372 of file MotionSequenceMC.cc.

References `getKeyFrame()`, `invalid_move`, `ERS210Info::NumOutputs`, `setRange()`, and `MotionSequence::Move::starttime`.

7.109.4.32 void MotionSequence::setOutputCmd (unsigned int *i*, const [OutputCmd](#) & *cmd*)

will insert a keyframe for the given output, or change an existing one

Definition at line 237 of file MotionSequenceMC.cc.

References [MotionSequence::Move::cmd](#), [endtime](#), [getKeyFrame\(\)](#), [invalid_move](#), [Move_idx_t](#), [newKeyFrame\(\)](#), [MotionSequence::Move::next](#), [nexts](#), [playtime](#), [MotionSequence::Move::prev](#), [prevs](#), and [MotionSequence::Move::starttime](#).

7.109.4.33 void MotionSequence::setPlaySpeed (float *x*) [inline]

sets the playback speed (e.g. 1=regular, 0.5=half speed, -1=**backwards**)

Definition at line 114 of file MotionSequenceMC.h.

References [playspeed](#).

7.109.4.34 void MotionSequence::setPlayTime (unsigned int *x*)

set the time for both playback and editing (in milliseconds)

Definition at line 231 of file MotionSequenceMC.cc.

References [nexts](#), [ERS210Info::NumOutputs](#), [playtime](#), [prevs](#), and [setRange\(\)](#).

7.109.4.35 void MotionSequence::setPose (const [PostureEngine](#) & *pose*)

calls setOutputCmd on each of the OutputCmds in *pose*

Definition at line 261 of file MotionSequenceMC.cc.

References [PostureEngine::getOutputCmd\(\)](#), [ERS210Info::NumOutputs](#), and [setOutputCmd\(\)](#).

7.109.4.36 virtual void MotionSequence::setRange (unsigned int *t*, [Move_idx_t](#) & *prev*, [Move_idx_t](#) & *next*) const [protected, pure virtual]

Sets prev and next to the appropriate values for the given time and output index.

Implemented in [DynamicMotionSequence](#), and [MotionSequenceMC< MAXMOVE >](#).

7.109.4.37 void MotionSequence::setSaveDegrees () [inline]

will store angles as degrees on future saves

Definition at line 88 of file MotionSequenceMC.h.

References loadSaveMode.

7.109.4.38 void MotionSequence::setSaveRadians () [inline]

will store angles as radians on future saves

Definition at line 90 of file MotionSequenceMC.h.

References loadSaveMode.

7.109.4.39 int MotionSequence::updateOutputs () [virtual]

is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)

Returns:

zero if no changes were made, non-zero otherwise

See also:

[RobotInfo::NumFrames](#)

[RobotInfo::FrameTime](#)

Implements [MotionCommand](#).

Reimplemented in [DynamicMotionSequence](#), and [MotionSequenceMC< MAX-MOVE >](#).

Definition at line 11 of file MotionSequenceMC.cc.

References [get_time\(\)](#), [isPlaying\(\)](#), [lasttime](#), [play\(\)](#), [playspeed](#), [playtime](#), and [setPlayTime\(\)](#).

7.109.5 Member Data Documentation**7.109.5.1 OutputCmd MotionSequence::curs[NumOutputs] [protected]**

merely a cache of current values (if computed, see [curstamps](#))

Definition at line 137 of file MotionSequenceMC.h.

7.109.5.2 `unsigned int MotionSequence::curstamps[NumOutputs]` [protected]

timestamp of corresponding value in `curs`

Definition at line 138 of file MotionSequenceMC.h.

7.109.5.3 `unsigned int MotionSequence::endtime` [protected]

max of #moves's `Move::starttime`'s

Definition at line 141 of file MotionSequenceMC.h.

7.109.5.4 `MotionSequence::Move_idx_t MotionSequence::invalid_move = -1U` [static, protected]

used to mark the ends of the `Move` linked lists

Definition at line 9 of file MotionSequenceMC.cc.

7.109.5.5 `unsigned int MotionSequence::lasttime` [protected]

the time of the last update

Definition at line 140 of file MotionSequenceMC.h.

7.109.5.6 `float MotionSequence::loadSaveMode` [protected]

1 to use radians, `M_PI/180` for degrees during a save

Definition at line 145 of file MotionSequenceMC.h.

7.109.5.7 `Move_idx_t MotionSequence::nexts[NumOutputs]` [protected]

the upcoming frame (the starttime for this frame will always be greater than playtime)

Definition at line 136 of file MotionSequenceMC.h.

7.109.5.8 `bool MotionSequence::playing` [protected]

true if playing, false if paused

Definition at line 143 of file MotionSequenceMC.h.

7.109.5.9 float **MotionSequence::playspeed** [protected]

multiplies the difference between current time and starttime, negative will cause play backwards

Definition at line 142 of file MotionSequenceMC.h.

7.109.5.10 unsigned int **MotionSequence::playtime** [protected]

the current time of playback, 0 is start of sequence

Definition at line 139 of file MotionSequenceMC.h.

7.109.5.11 **Move_idx_t** **MotionSequence::prevs**[NumOutputs] [protected]

the previous frame (the starttime for this frame will always be less than or equal to playtime)

Definition at line 135 of file MotionSequenceMC.h.

7.109.5.12 const unsigned int **MotionSequence::SizeLarge** = NumOutputs*11
[static]

Large (10 full body frames ~ est 8KB).

Definition at line 74 of file MotionSequenceMC.h.

7.109.5.13 const unsigned int **MotionSequence::SizeMedium** = NumOutputs*6
[static]

Medium (5 full body frames ~ est 4KB).

Definition at line 73 of file MotionSequenceMC.h.

7.109.5.14 const unsigned int **MotionSequence::SizeSmall** = NumOutputs*3
[static]

Small, but still big enough to handle most of the included MS's (2 full-body frames ~ around 1KB).

Definition at line 72 of file MotionSequenceMC.h.

7.109.5.15 `const unsigned int MotionSequence::SizeTiny = NumOutputs*2`
[static]

Tiny, but enough to handle a transition into a full-body pose.

Definition at line 71 of file MotionSequenceMC.h.

7.109.5.16 `const unsigned int MotionSequence::SizeXLarge = NumOutputs*26`
[static]

eXtra Large (25 full body frames ~ est 16KB)

Definition at line 75 of file MotionSequenceMC.h.

7.109.5.17 `Move_idx_t MotionSequence::starts[NumOutputs]` [protected]

the beginning frame for each output animation

Definition at line 134 of file MotionSequenceMC.h.

The documentation for this class was generated from the following files:

- [MotionSequenceMC.h](#)
- [MotionSequenceMC.cc](#)

7.110 MotionSequence::Move Struct Reference

```
#include <MotionSequenceMC.h>
```

7.110.1 Detailed Description

This struct holds all the information needed about a frame for a particular output.

Definition at line 124 of file MotionSequenceMC.h.

Public Member Functions

- [Move \(\)](#)
constructor

Public Attributes

- [OutputCmd cmd](#)
the actual command to use
- [Move_idx_t next](#)
the next frame
- [Move_idx_t prev](#)
the previous frame
- unsigned int [starttime](#)
the time (relative to first frame) this frame should be expressed at

7.110.2 Constructor & Destructor Documentation

7.110.2.1 MotionSequence::Move::Move () [inline]

constructor

Definition at line 126 of file MotionSequenceMC.h.

References [cmd](#), [next](#), [prev](#), and [starttime](#).

7.110.3 Member Data Documentation

7.110.3.1 [OutputCmd](#) [MotionSequence::Move::cmd](#)

the actual command to use

Definition at line 127 of file MotionSequenceMC.h.

7.110.3.2 [Move_idx_t](#) [MotionSequence::Move::next](#)

the next frame

Definition at line 128 of file MotionSequenceMC.h.

7.110.3.3 [Move_idx_t](#) [MotionSequence::Move::prev](#)

the previous frame

Definition at line 129 of file MotionSequenceMC.h.

7.110.3.4 [unsigned int](#) [MotionSequence::Move::starttime](#)

the time (relative to first frame) this frame should be expressed at

Definition at line 130 of file MotionSequenceMC.h.

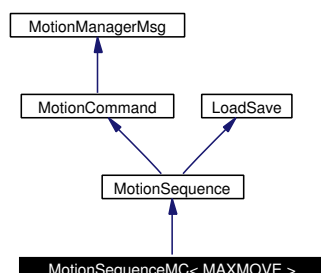
The documentation for this struct was generated from the following file:

- [MotionSequenceMC.h](#)

7.111 MotionSequenceMC< MAXMOVE > Class Template Reference

```
#include <MotionSequenceMC.h>
```

Inheritance diagram for MotionSequenceMC< MAXMOVE >:



7.111.1 Detailed Description

```
template<unsigned int MAXMOVE> class MotionSequenceMC< MAXMOVE
>
```

Instantiates MotionSequences - when you want to make a new [MotionSequence](#), make one of these.

Allows a (compile-time) variable amount of data storage through its template parameter. See [MotionSequence](#) for documentation on its members

See also:

[MotionSequence](#)
[MotionSequence::SizeSmall](#), [MotionSequence::SizeMedium](#), [MotionSequence::SizeLarge](#), [MotionSequence::SizeXLarge](#),

Definition at line 184 of file MotionSequenceMC.h.

Public Member Functions

- [MotionSequenceMC](#) ()
constructor
- [MotionSequenceMC](#) (const char *filename)
constructor, loads from a file and then resets the playtime to beginning and begins to play

- virtual [~MotionSequenceMC](#) ()
destructor
- virtual int [updateOutputs](#) ()
is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)
- virtual void [clear](#) ()
empties out the sequence (constant time operation - faster than a series of pops)
- virtual unsigned int [getMaxFrames](#) () const
returns the maximum number of key frames (Move's) which can be stored, determined by the instantiating [MotionSequenceMC](#)'s template parameter
- virtual unsigned int [getUsedFrames](#) () const
returns the number of used key frames (Move's) which have been stored by the instantiation [MotionSequence](#) subclass

Protected Types

- typedef [ListMemBuf](#)< Move, MAXMOVE, [Move_idx_t](#) > [list_t](#)
shorthand for the [ListMemBuf](#) that stores all of the movement frames

Protected Member Functions

- virtual Move & [getKeyFrame](#) ([Move_idx_t](#) x)
returns the Move struct corresponding to x in the subclass's actual data structure
- virtual const Move & [getKeyFrame](#) ([Move_idx_t](#) x) const
returns the Move struct corresponding to x in the subclass's actual data structure
- virtual [Move_idx_t](#) [newKeyFrame](#) ()
causes subclass to create a new Move structure, returns its index
- virtual void [eraseKeyFrame](#) ([Move_idx_t](#) x)
causes subclass to mark the corresponding Move structure as free
- void [setRange](#) (unsigned int t, [Move_idx_t](#) &prev, [Move_idx_t](#) &next) const
Sets prev and next to the appropriate values for the given time and output index.

Protected Attributes

- [list_t](#) moves

stores all of the movement keyframes

7.111.2 Member Typedef Documentation

7.111.2.1 `template<unsigned int MAXMOVE> typedef ListMemBuf<Move,MAXMOVE,Move_idx_t> MotionSequenceMC<MAXMOVE >::list_t [protected]`

shorthand for the [ListMemBuf](#) that stores all of the movement frames

Definition at line 263 of file MotionSequenceMC.h.

7.111.3 Constructor & Destructor Documentation

7.111.3.1 `template<unsigned int MAXMOVE> MotionSequenceMC<MAXMOVE >::MotionSequenceMC () [inline]`

constructor

Definition at line 187 of file MotionSequenceMC.h.

References [MotionSequenceMC< MAXMOVE >::clear\(\)](#), and [MotionSequenceMC< MAXMOVE >::moves](#).

7.111.3.2 `template<unsigned int MAXMOVE> MotionSequenceMC<MAXMOVE >::MotionSequenceMC (const char *filename) [inline, explicit]`

constructor, loads from a file and then resets the playtime to beginning and begins to play

Definition at line 189 of file MotionSequenceMC.h.

References [MotionSequenceMC< MAXMOVE >::clear\(\)](#), [LoadSave::LoadFile\(\)](#), [MotionSequenceMC< MAXMOVE >::moves](#), and [MotionSequence::setPlayTime\(\)](#).

7.111.3.3 `template<unsigned int MAXMOVE> virtual MotionSequenceMC<MAXMOVE >::~~MotionSequenceMC () [inline, virtual]`

destructor

Definition at line 191 of file MotionSequenceMC.h.

7.111.4 Member Function Documentation

7.111.4.1 `template<unsigned int MAXMOVE> virtual void
MotionSequenceMC< MAXMOVE >::clear () [inline,
virtual]`

empties out the sequence (constant time operation - faster than a series of pops)

Implements [MotionSequence](#).

Definition at line 247 of file MotionSequenceMC.h.

References [ListMemBuf< Move, MAXMOVE, Move_idx_t >::back\(\)](#), [ListMemBuf< Move, MAXMOVE, Move_idx_t >::clear\(\)](#), [MotionSequence::invalid_move](#), [MotionSequenceMC< MAXMOVE >::moves](#), [ListMemBuf< Move, MAXMOVE, Move_idx_t >::new_back\(\)](#), [MotionSequence::nexts](#), [ERS210Info::NumOutputs](#), [MotionSequence::prevs](#), [MotionSequence::setPlayTime\(\)](#), and [MotionSequence::starts](#).

7.111.4.2 `template<unsigned int MAXMOVE> virtual void
MotionSequenceMC< MAXMOVE >::eraseKeyFrame (Move_idx_t
x) [inline, protected, virtual]`

causes subclass to mark the corresponding Move structure as free

Implements [MotionSequence](#).

Definition at line 271 of file MotionSequenceMC.h.

References [ListMemBuf< Move, MAXMOVE, Move_idx_t >::erase\(\)](#), and [MotionSequenceMC< MAXMOVE >::moves](#).

7.111.4.3 `template<unsigned int MAXMOVE> virtual const Move&
MotionSequenceMC< MAXMOVE >::getKeyFrame (Move_idx_t x)
const [inline, protected, virtual]`

returns the Move struct corresponding to *x* in the subclass's actual data structure

Implements [MotionSequence](#).

Definition at line 269 of file MotionSequenceMC.h.

References [MotionSequenceMC< MAXMOVE >::moves](#).

7.111.4.4 `template<unsigned int MAXMOVE> virtual Move&
MotionSequenceMC< MAXMOVE >::getKeyFrame (Move_idx_t x)
[inline, protected, virtual]`

returns the Move struct corresponding to *x* in the subclass's actual data structure

Implements [MotionSequence](#).

Definition at line 268 of file MotionSequenceMC.h.

References MotionSequenceMC< MAXMOVE >::moves.

7.111.4.5 `template<unsigned int MAXMOVE> virtual unsigned int
MotionSequenceMC< MAXMOVE >::getMaxFrames () const
[inline, virtual]`

returns the maximum number of key frames (Move's) which can be stored, determined by the instantiating MotionSequenceMC's template parameter

Implements [MotionSequence](#).

Definition at line 258 of file MotionSequenceMC.h.

References ListMemBuf< Move, MAXMOVE, Move_idx_t >::getMaxCapacity(), and MotionSequenceMC< MAXMOVE >::moves.

7.111.4.6 `template<unsigned int MAXMOVE> virtual unsigned int
MotionSequenceMC< MAXMOVE >::getUsedFrames () const
[inline, virtual]`

returns the number of used key frames (Move's) which have been stored by the instantiation [MotionSequence](#) subclass

Implements [MotionSequence](#).

Definition at line 259 of file MotionSequenceMC.h.

References MotionSequenceMC< MAXMOVE >::moves, and ListMemBuf< Move, MAXMOVE, Move_idx_t >::size().

7.111.4.7 `template<unsigned int MAXMOVE> virtual Move_idx_t
MotionSequenceMC< MAXMOVE >::newKeyFrame () [inline,
protected, virtual]`

causes subclass to create a new Move structure, returns its index

Implements [MotionSequence](#).

Definition at line 270 of file MotionSequenceMC.h.

References MotionSequence::Move_idx_t, MotionSequenceMC< MAXMOVE >::moves, and ListMemBuf< Move, MAXMOVE, Move_idx_t >::new_front().

7.111.4.8 `template<unsigned int MAXMOVE> void MotionSequenceMC<
MAXMOVE >::setRange (unsigned int t, Move_idx_t & prev,
Move_idx_t & next) const` [inline, protected, virtual]

Sets prev and next to the appropriate values for the given time and output index.

Implements [MotionSequence](#).

Definition at line 272 of file MotionSequenceMC.h.

References [MotionSequence::invalid_move](#), and [MotionSequenceMC](#)< MAXMOVE >::moves.

7.111.4.9 `template<unsigned int MAXMOVE> virtual int
MotionSequenceMC< MAXMOVE >::updateOutputs ()`
[inline, virtual]

is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)

Returns:

zero if no changes were made, non-zero otherwise

See also:

[RobotInfo::NumFrames](#)

[RobotInfo::FrameTime](#)

Reimplemented from [MotionSequence](#).

Definition at line 222 of file MotionSequenceMC.h.

References [MotionSequence::calcOutput\(\)](#), [ERS210Info::FrameTime](#), [MotionSequence::getOutputCmd\(\)](#), [MotionSequence::invalid_move](#), [MotionSequence::isPlaying\(\)](#), [motman](#), [MotionSequence::Move_idx_t](#), [MotionSequenceMC](#)< MAXMOVE >::moves, [MotionSequence::nexts](#), [ERS210Info::NumFrames](#), [ERS210Info::NumOutputs](#), [MotionSequence::playtime](#), [MotionSequence::prevs](#), [MotionManager::setOutput\(\)](#), [MotionSequenceMC](#)< MAXMOVE >::setRange(), [OutputCmd::unset\(\)](#), and [MotionSequence::updateOutputs\(\)](#).

7.111.5 Member Data Documentation

7.111.5.1 `template<unsigned int MAXMOVE> list_t MotionSequenceMC<
MAXMOVE >::moves` [protected]

stores all of the movement keyframes

Definition at line 266 of file MotionSequenceMC.h.

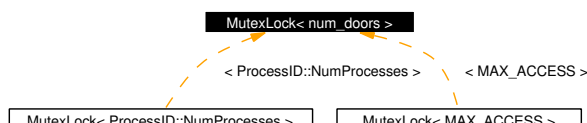
The documentation for this class was generated from the following file:

- [MotionSequenceMC.h](#)

7.112 MutexLock< num_doors > Class Template Reference

```
#include <MutexLock.h>
```

Inheritance diagram for MutexLock< num_doors >:



7.112.1 Detailed Description

template<unsigned int num_doors> class MutexLock< num_doors >

A software only mutual exclusion lock.

Use this to prevent more than one process from accessing a data structure at the same time (which often leads to unpredictable and unexpected results)

The template parameter specifies the maximum number of different processes which need to be protected. This needs to be allocated ahead of time, as there doesn't seem to be a way to dynamically scale as needed without risking possible errors if two processes are both trying to set up at the same time. Also, by using a template parameter, all data structures are contained within the class's memory allocation, so no pointers are involved.

Locks in this class can be recursive. If you lock 5 times and then [unlock\(\)](#) once, the lock is released. If you want to have releases match locks, call [release\(\)](#) instead of [unlock\(\)](#).

Note that there is no check that the process doing the unlocking is the one that actually has the lock. Be careful about this.

Warning:

Doing mutual exclusion in software is tricky business, be careful about any modifications you make!

Implements a first-come-first-served Mutex as laid out on page 11 of:

"A First Come First Served Mutual Exclusion Algorithm with Small Communication Variables"

Edward A. Lycklama, Vassos Hadzilacos - Aug. 1991

Definition at line 36 of file MutexLock.h.

Public Member Functions

- [MutexLock](#) ()
constructor, just calls the [init\(\)](#) function.
- void [lock](#) (int id)
blocks (by busy looping on [do_try_lock\(\)](#)) until a lock is achieved
- bool [try_lock](#) (int id)
attempts to get a lock, returns true if it succeeds
- void [release](#) ()
releases one recursive lock-level from whoever has the current lock
- void [unlock](#) ()
completely unlocks, regardless of how many times a recursive lock has been obtained
- unsigned int [get_lock_level](#) () const
returns the lockcount
- int [owner](#) ()
returns the current owner's id
- void [forget](#) (int id)
*allows you to reset one of the possible owners, so another process can take its place.
This is not tested*

Static Public Attributes

- const unsigned int [NO_OWNER](#) = -1U
marks as unlocked

Protected Member Functions

- bool [do_try_lock](#) (unsigned int index, bool block)
Does the work of trying to get a lock.
- unsigned int [lookup](#) (int id)
returns the internal index mapping to the id number supplied by the process

- void `init` ()

Doesn't do anything if you have the `MUTEX_LOCK_ET_USE_SPINCOUNT` undef'ed. Used to do a `memset`, but that was causing problems....

- void `spin` ()

If you find a way to sleep for a few microseconds instead of busy waiting, put it here.

Protected Attributes

- `door_t doors` [num_doors]

holds all the doors

- unsigned int `doors_used`

counts the number of doors used

- unsigned int `owner_index`

holds the door index of the current lock owner

- unsigned int `lockcount`

the depth of the lock, 0 when unlocked

7.112.2 Constructor & Destructor Documentation

7.112.2.1 `template<unsigned int num_doors> MutexLock< num_doors >::MutexLock () [inline]`

constructor, just calls the `init()` function.

Definition at line 41 of file `MutexLock.h`.

7.112.3 Member Function Documentation

7.112.3.1 `template<unsigned int num_doors> bool MutexLock< num_doors >::do_try_lock (unsigned int index, bool block) [protected]`

Does the work of trying to get a lock.

Pass `true` for *block* if you want it to use FCFS blocking instead of just returning right away if another process has the lock

Definition at line 166 of file `MutexLock.h`.

References MutexLock< num_doors >::doors, MutexLock< num_doors >::door_t::FCFS_in_use, mutexdebugout, MutexLock< num_doors >::NO_OWNER, MutexLock< num_doors >::owner_index, MutexLock< num_doors >::spin(), and MutexLock< num_doors >::door_t::turn.

7.112.3.2 `template<unsigned int num_doors> void MutexLock< num_doors >::forget (int id)`

allows you to reset one of the possible owners, so another process can take its place. This is not tested

Definition at line 245 of file MutexLock.h.

References MutexLock< num_doors >::do_try_lock(), MutexLock< num_doors >::doors, MutexLock< num_doors >::doors_used, MutexLock< num_doors >::door_t::id, MutexLock< num_doors >::lookup(), MutexLock< num_doors >::NO_OWNER, and MutexLock< num_doors >::release().

7.112.3.3 `template<unsigned int num_doors> unsigned int MutexLock< num_doors >::get_lock_level () const [inline]`

returns the lockcount

Definition at line 67 of file MutexLock.h.

7.112.3.4 `template<unsigned int num_doors> void MutexLock< num_doors >::init () [inline, protected]`

Doesn't do anything if you have the MUTEX_LOCK_ET_USE_SPINCOUNT undef'ed. Used to do a memset, but that was causing problems....

Definition at line 94 of file MutexLock.h.

7.112.3.5 `template<unsigned int num_doors> void MutexLock< num_doors >::lock (int id)`

blocks (by busy looping on [do_try_lock\(\)](#)) until a lock is achieved

You should pass some process-specific ID number as the input - just make sure no other process will be using the same value.

This is not a recursive lock - repeated locks still only require one release to undo them.

Todo

- I'd like to not use a loop here

Definition at line 120 of file MutexLock.h.

References MutexLock< num_doors >::do_try_lock(), MutexLock< num_doors >::lockcount, MutexLock< num_doors >::lookup(), MutexLock< num_doors >::owner(), and MutexLock< num_doors >::spin().

7.112.3.6 **template<unsigned int num_doors> unsigned int** **MutexLock< num_doors >::lookup (int id)** [protected]

returns the internal index mapping to the id number supplied by the process

Definition at line 226 of file MutexLock.h.

References MutexLock< num_doors >::doors, MutexLock< num_doors >::doors_-, used, MutexLock< num_doors >::door_t::id, and MutexLock< num_doors >::NO_OWNER.

7.112.3.7 **template<unsigned int num_doors> int** **MutexLock< num_doors >::owner ()** [inline]

returns the current owner's id

Definition at line 70 of file MutexLock.h.

7.112.3.8 **template<unsigned int num_doors> void** **MutexLock< num_doors >::release ()**

releases one recursive lock-level from whoever has the current lock

Definition at line 146 of file MutexLock.h.

References MutexLock< num_doors >::door_t::BL_in_use, MutexLock< num_doors >::door_t::BL_ready, MutexLock< num_doors >::doors, MutexLock< num_doors >::lockcount, MutexLock< num_doors >::NO_OWNER, and MutexLock< num_doors >::owner_index.

7.112.3.9 **template<unsigned int num_doors> void** **MutexLock< num_doors >::spin ()** [inline, protected]

If you find a way to sleep for a few microseconds instead of busy waiting, put it here.

Definition at line 96 of file MutexLock.h.

7.112.3.10 `template<unsigned int num_doors> bool MutexLock< num_doors >::try_lock (int id)`

attempts to get a lock, returns true if it succeeds

You should pass some process-specific ID number as the input - just make sure no other process will be using the same value.

This is not a recursive lock - repeated locks still only require one release to undo them.

Definition at line 130 of file MutexLock.h.

References `MutexLock< num_doors >::do_try_lock()`, `MutexLock< num_doors >::lockcount`, `MutexLock< num_doors >::lookup()`, and `MutexLock< num_doors >::owner()`.

7.112.3.11 `template<unsigned int num_doors> void MutexLock< num_doors >::unlock () [inline]`

completely unlocks, regardless of how many times a recursive lock has been obtained

Definition at line 64 of file MutexLock.h.

7.112.4 Member Data Documentation

7.112.4.1 `template<unsigned int num_doors> door_t MutexLock< num_doors >::doors[num_doors] [protected]`

holds all the doors

Definition at line 111 of file MutexLock.h.

7.112.4.2 `template<unsigned int num_doors> unsigned int MutexLock< num_doors >::doors_used [protected]`

counts the number of doors used

Definition at line 112 of file MutexLock.h.

7.112.4.3 `template<unsigned int num_doors> unsigned int MutexLock< num_doors >::lockcount [protected]`

the depth of the lock, 0 when unlocked

Definition at line 114 of file MutexLock.h.

7.112.4.4 `template<unsigned int num_doors> const unsigned int MutexLock<
num_doors >::NO_OWNER = -1U [static]`

marks as unlocked

Definition at line 38 of file MutexLock.h.

7.112.4.5 `template<unsigned int num_doors> unsigned int MutexLock<
num_doors >::owner_index [protected]`

holds the door index of the current lock owner

Definition at line 113 of file MutexLock.h.

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

- [MutexLock.h](#)

7.113 MutexLock< num_doors >::door_t Struct Reference

```
#include <MutexLock.h>
```

7.113.1 Detailed Description

template<unsigned int num_doors> struct MutexLock< num_doors >::door_t

Holds per process shared info, one of these per process.

Definition at line 100 of file MutexLock.h.

Public Member Functions

- [door_t](#) ()
constructor

Public Attributes

- int [id](#)
process ID this doorway is assigned to
- volatile bool [FCFS_in_use](#)
In FCFS doorway, corresponds to 'c.i'.
- volatile bool [BL_ready](#)
Signals past FCFS doorway, ready for BL doorway, corresponds to 'v.i'.
- volatile bool [BL_in_use](#)
Burns-Lamport doorway, corresponds to 'x.i'.
- volatile unsigned char [turn](#)
clock pulse, initial value doesn't matter
- unsigned char [next_turn_bit](#)
selects which bit of turn will be flipped next

7.113.2 Constructor & Destructor Documentation

7.113.2.1 `template<unsigned int num_doors> MutexLock< num_doors >::door_t::door_t () [inline]`

constructor

Definition at line 101 of file MutexLock.h.

References `MutexLock< num_doors >::door_t::BL_in_use`, `MutexLock< num_doors >::door_t::BL_ready`, `MutexLock< num_doors >::door_t::FCFS_in_use`, `MutexLock< num_doors >::door_t::id`, `MutexLock< num_doors >::door_t::next_turn_bit`, and `MutexLock< num_doors >::door_t::turn`.

7.113.3 Member Data Documentation

7.113.3.1 `template<unsigned int num_doors> volatile bool MutexLock< num_doors >::door_t::BL_in_use`

Burns-Lamport doorway, corresponds to 'x_i'.

Definition at line 106 of file MutexLock.h.

7.113.3.2 `template<unsigned int num_doors> volatile bool MutexLock< num_doors >::door_t::BL_ready`

Signals past FCFS doorway, ready for BL doorway, corresponds to 'v_i'.

Definition at line 105 of file MutexLock.h.

7.113.3.3 `template<unsigned int num_doors> volatile bool MutexLock< num_doors >::door_t::FCFS_in_use`

In FCFS doorway, corresponds to 'c_i'.

Definition at line 104 of file MutexLock.h.

7.113.3.4 `template<unsigned int num_doors> int MutexLock< num_doors >::door_t::id`

process ID this doorway is assigned to

Definition at line 103 of file MutexLock.h.

7.113.3.5 `template<unsigned int num_doors> unsigned char MutexLock< num_doors >::door_t::next_turn_bit`

selects which bit of turn will be flipped next

Definition at line 108 of file MutexLock.h.

7.113.3.6 `template<unsigned int num_doors> volatile unsigned char MutexLock< num_doors >::door_t::turn`

clock pulse, initial value doesn't matter

Definition at line 107 of file MutexLock.h.

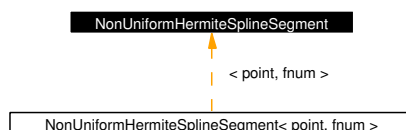
The documentation for this struct was generated from the following file:

- [MutexLock.h](#)

7.114 NonUniformHermiteSplineSegment Class Reference

```
#include <Spline.h>
```

Inheritance diagram for NonUniformHermiteSplineSegment:



Public Member Functions

- [NonUniformHermiteSplineSegment](#) ()
- void [create](#) (point x0, point x1, point dx0, point dx1, fnum t1)
- void [create](#) (point *pts, double t1, int num, int i)
- point [eval](#) (fnum u)
- point [eval_deriv](#) (fnum u)

Public Attributes

- point [a](#)
- point [b](#)
- point [c](#)
- point [d](#)
- fnum [t](#)

7.114.1 Constructor & Destructor Documentation

7.114.1.1 NonUniformHermiteSplineSegment::NonUniformHermiteSplineSegment () [inline]

Definition at line 48 of file Spline.h.

7.114.2 Member Function Documentation

7.114.2.1 void NonUniformHermiteSplineSegment::create (point * *pts*, double *tI*, int *num*, int *i*)

7.114.2.2 void NonUniformHermiteSplineSegment::create (point *x0*, point *xI*, point *dx0*, point *dxI*, fnum *tI*)

7.114.2.3 point NonUniformHermiteSplineSegment::eval (fnum *u*)

7.114.2.4 point NonUniformHermiteSplineSegment::eval_deriv (fnum *u*)

7.114.3 Member Data Documentation

7.114.3.1 point [NonUniformHermiteSplineSegment::a](#)

Definition at line 49 of file Spline.h.

7.114.3.2 point [NonUniformHermiteSplineSegment::b](#)

Definition at line 49 of file Spline.h.

7.114.3.3 point [NonUniformHermiteSplineSegment::c](#)

Definition at line 49 of file Spline.h.

7.114.3.4 point [NonUniformHermiteSplineSegment::d](#)

Definition at line 49 of file Spline.h.

7.114.3.5 fnum [NonUniformHermiteSplineSegment::t](#)

Definition at line 50 of file Spline.h.

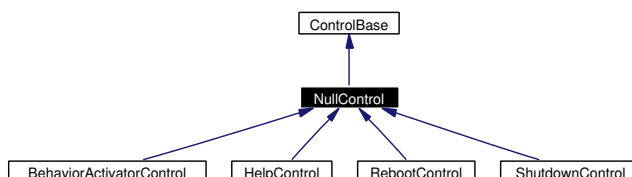
The documentation for this class was generated from the following file:

- [Spline.h](#)

7.115 NullControl Class Reference

```
#include <NullControl.h>
```

Inheritance diagram for NullControl:



7.115.1 Detailed Description

when activated, this will return immediately (handy for fake items in a menu)

Definition at line 8 of file NullControl.h.

Public Member Functions

- [NullControl](#) ()
Constructor.
- [NullControl](#) (const std::string &n)
Constructor.
- [NullControl](#) (const std::string &n, const std::string &d)
Constructor.
- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#), [Socket](#) *)
returns NULL
- virtual [ControlBase](#) * [doSelect](#) ()
returns NULL
- virtual [ControlBase](#) * [doNextItem](#) ()
returns NULL
- virtual [ControlBase](#) * [doPrevItem](#) ()
returns NULL

- virtual [ControlBase](#) * [doReadStdIn](#) (const std::string &=std::string())
returns NULL
- virtual [ControlBase](#) * [takeInput](#) (const std::string &)
returns NULL

7.115.2 Constructor & Destructor Documentation

7.115.2.1 [NullControl::NullControl](#) () [inline]

Constructor.

Definition at line 12 of file [NullControl.h](#).

7.115.2.2 [NullControl::NullControl](#) (const std::string & *n*) [inline]

Constructor.

Definition at line 14 of file [NullControl.h](#).

7.115.2.3 [NullControl::NullControl](#) (const std::string & *n*, const std::string & *d*) [inline]

Constructor.

Definition at line 16 of file [NullControl.h](#).

7.115.3 Member Function Documentation

7.115.3.1 virtual [ControlBase](#)* [NullControl::activate](#) ([MotionManager::MC.ID](#), [Socket](#) *) [inline, virtual]

returns NULL

Reimplemented from [ControlBase](#).

Reimplemented in [BehaviorActivatorControl](#), [HelpControl](#), [RebootControl](#), and [ShutdownControl](#).

Definition at line 20 of file [NullControl.h](#).

7.115.3.2 `virtual ControlBase* NullControl::doNextItem ()` [inline, virtual]

returns NULL

Reimplemented from [ControlBase](#).

Definition at line 23 of file NullControl.h.

7.115.3.3 `virtual ControlBase* NullControl::doPrevItem ()` [inline, virtual]

returns NULL

Reimplemented from [ControlBase](#).

Definition at line 24 of file NullControl.h.

7.115.3.4 `virtual ControlBase* NullControl::doReadStdIn (const std::string &= std::string())` [inline, virtual]

returns NULL

Reimplemented from [ControlBase](#).

Definition at line 25 of file NullControl.h.

7.115.3.5 `virtual ControlBase* NullControl::doSelect ()` [inline, virtual]

returns NULL

Reimplemented from [ControlBase](#).

Reimplemented in [RebootControl](#), and [ShutdownControl](#).

Definition at line 22 of file NullControl.h.

7.115.3.6 `virtual ControlBase* NullControl::takeInput (const std::string &)` [inline, virtual]

returns NULL

Reimplemented from [ControlBase](#).

Definition at line 26 of file NullControl.h.

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

- [NullControl.h](#)

7.116 VisionInterface::ObjectInfo Struct Reference

```
#include <VisionInterface.h>
```

Public Attributes

- [VObject marker](#) [12]
- [VObject rball](#)
- [VObject pball](#)
- [VObject hand](#)
- [VObject thing](#)

7.116.1 Member Data Documentation

7.116.1.1 [VObject VisionInterface::ObjectInfo::hand](#)

Definition at line 87 of file VisionInterface.h.

7.116.1.2 [VObject VisionInterface::ObjectInfo::marker](#)[12]

Definition at line 84 of file VisionInterface.h.

7.116.1.3 [VObject VisionInterface::ObjectInfo::pball](#)

Definition at line 86 of file VisionInterface.h.

7.116.1.4 [VObject VisionInterface::ObjectInfo::rball](#)

Definition at line 85 of file VisionInterface.h.

7.116.1.5 [VObject VisionInterface::ObjectInfo::thing](#)

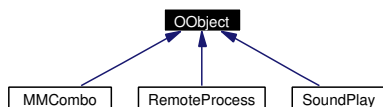
Definition at line 88 of file VisionInterface.h.

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

- [VisionInterface.h](#)

7.117 OObject Class Reference

Inheritance diagram for OObject:

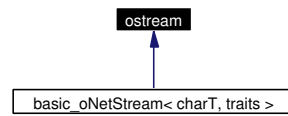


The documentation for this class was generated from the following file:

- [RemoteProcess.h](#)

7.118 ostream Class Reference

Inheritance diagram for ostream:



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

- [ionetstream.h](#)

7.119 OutputCmd Class Reference

```
#include <OutputCmd.h>
```

7.119.1 Detailed Description

This object holds all the information needed to control a single output.

Definition at line 6 of file OutputCmd.h.

Public Member Functions

- [OutputCmd](#) ()
Constructor.
- [OutputCmd](#) (float v)
Constructor.
- [OutputCmd](#) (float v, float w)
Constructor.
- [OutputCmd](#) (const [OutputCmd](#) &a, const [OutputCmd](#) &b, float w)
Constructor, see set(a,b,w).
- void [set](#) (float v, float w=1)
sets the value to v and weight to w
- void [set](#) (const [OutputCmd](#) &a, const [OutputCmd](#) &b, float w)
sets the value to a weighted average of a and b (higher w, more a)
- void [unset](#) ()
sets value and weight to 0 (same as assigning ::unusedJoint)
- bool [operator==](#) (const [OutputCmd](#) &c) const
tests for equality of weight and value
- bool [operator!=](#) (const [OutputCmd](#) &c) const
tests for inequality of weight and value

Public Attributes

- float [value](#)

value of the output

- float [weight](#)

weight to be used in averaging, 0 to "fall through"

7.119.2 Constructor & Destructor Documentation

7.119.2.1 `OutputCmd::OutputCmd ()` [inline]

Constructor.

Definition at line 9 of file `OutputCmd.h`.

References [value](#), and [weight](#).

7.119.2.2 `OutputCmd::OutputCmd (float v)` [inline]

Constructor.

Definition at line 10 of file `OutputCmd.h`.

References [value](#), and [weight](#).

7.119.2.3 `OutputCmd::OutputCmd (float v, float w)` [inline]

Constructor.

Definition at line 11 of file `OutputCmd.h`.

References [value](#), and [weight](#).

7.119.2.4 `OutputCmd::OutputCmd (const OutputCmd & a, const OutputCmd & b, float w)` [inline]

Constructor, see `set(a,b,w)`.

Definition at line 12 of file `OutputCmd.h`.

References [value](#), and [weight](#).

7.119.3 Member Function Documentation

7.119.3.1 `bool OutputCmd::operator!= (const OutputCmd & c) const`
[inline]

tests for inequality of weight and value

Definition at line 18 of file OutputCmd.h.

References value, and weight.

7.119.3.2 `bool OutputCmd::operator== (const OutputCmd & c) const`
[inline]

tests for equality of weight and value

Definition at line 17 of file OutputCmd.h.

References value, and weight.

7.119.3.3 `void OutputCmd::set (const OutputCmd & a, const OutputCmd & b, float w) [inline]`

sets the value to a weighted average of *a* and *b* (higher *w*, more *a*)

Definition at line 15 of file OutputCmd.h.

References value, and weight.

7.119.3.4 `void OutputCmd::set (float v, float w = 1) [inline]`

sets the value to *v* and weight to *w*

Definition at line 14 of file OutputCmd.h.

References value, and weight.

7.119.3.5 `void OutputCmd::unset () [inline]`

sets value and weight to 0 (same as assigning ::unusedJoint)

Definition at line 16 of file OutputCmd.h.

References value, and weight.

7.119.4 Member Data Documentation

7.119.4.1 float [OutputCmd::value](#)

value of the output

Definition at line 20 of file OutputCmd.h.

7.119.4.2 float [OutputCmd::weight](#)

weight to be used in averaging, 0 to "fall through"

Definition at line 21 of file OutputCmd.h.

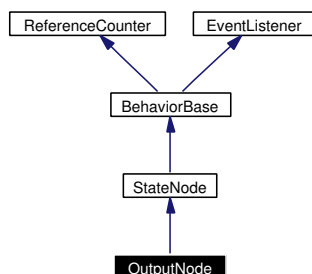
The documentation for this class was generated from the following file:

- [OutputCmd.h](#)

7.120 OutputNode Class Reference

```
#include <OutputNode.h>
```

Inheritance diagram for OutputNode:



7.120.1 Detailed Description

A very simple [StateNode](#) that outputs its name to a given ostream upon activation, handy for debugging.

Definition at line 10 of file OutputNode.h.

Public Member Functions

- [OutputNode](#) ()
constructor, uses cout for output
- [OutputNode](#) (const char *nm, [StateNode](#) *par, ostream &output)
constructor, sets name and ostream to use for output
- [OutputNode](#) (const char *nm, [StateNode](#) *par, ostream &output, [StateNode](#) *nextstate)
constructor, sets name and another state which will immediately be transitioned to upon activation
- virtual void [DoStart](#) ()
outputs this state's name, will transition to [next](#) if non-NULL

Protected Attributes

- [StateNode](#) * [next](#)

the state to transition to upon entering, can be NULL

- `ostream` & `out`

the stream to use for output - if not specified (default constructor) cout will be used

Private Member Functions

- `OutputNode` (const `OutputNode` &node)

don't call this

- `OutputNode operator=` (const `OutputNode` &node)

don't call this

7.120.2 Constructor & Destructor Documentation

7.120.2.1 `OutputNode::OutputNode ()` [inline]

constructor, uses cout for output

Definition at line 13 of file `OutputNode.h`.

References `next`, and `out`.

7.120.2.2 `OutputNode::OutputNode (const char * nm, StateNode * par, ostream & output)` [inline]

constructor, sets name and ostream to use for output

Definition at line 15 of file `OutputNode.h`.

References `next`, and `out`.

7.120.2.3 `OutputNode::OutputNode (const char * nm, StateNode * par, ostream & output, StateNode * nextstate)` [inline]

constructor, sets name and another state which will immediately be transitioned to upon activation

Definition at line 17 of file `OutputNode.h`.

References `next`, and `out`.

7.120.2.4 OutputNode::OutputNode (const [OutputNode](#) & node) [private]

don't call this

7.120.3 Member Function Documentation

7.120.3.1 virtual void OutputNode::DoStart () [inline, virtual]

outputs this state's name, will transition to [next](#) if non-NULL

if [next](#) is NULL, the state will simply stay active until some other transition causes it to leave

Reimplemented from [StateNode](#).

Definition at line 21 of file OutputNode.h.

References [StateNode::DoStart\(\)](#), [StateNode::DoStop\(\)](#), [StateNode::name](#), [next](#), and [out](#).

7.120.3.2 [OutputNode](#) OutputNode::operator= (const [OutputNode](#) & node) [private]

don't call this

7.120.4 Member Data Documentation

7.120.4.1 [StateNode*](#) [OutputNode::next](#) [protected]

the state to transition to upon entering, can be NULL

Definition at line 30 of file OutputNode.h.

7.120.4.2 [ostream&](#) [OutputNode::out](#) [protected]

the stream to use for output - if not specified (default constructor) cout will be used

Definition at line 31 of file OutputNode.h.

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

- [OutputNode.h](#)

7.121 OutputPID Class Reference

```
#include <OutputPID.h>
```

7.121.1 Detailed Description

This object holds all the information needed to control a single output.

Definition at line 6 of file OutputPID.h.

Public Member Functions

- [OutputPID](#) ()
Constructor.
- [OutputPID](#) (const float p[3])
Constructor.
- [OutputPID](#) (const float p[3], float w)
Constructor.
- [OutputPID](#) (const [OutputPID](#) &a, const [OutputPID](#) &b, float w)
Constructor, see set(a,b,w).
- void [set](#) (const float p[3], float w=1)
sets the value to v and weight to w
- void [set](#) (const [OutputPID](#) &a, const [OutputPID](#) &b, float w)
sets the value to a weighted average of a and b (higher w, more a)
- void [unset](#) ()
sets value and weight to 0 (same as assigning ::unusedJoint)

Public Attributes

- float [pid](#) [3]
pid value of the output
- float [weight](#)
weight to be used in averaging, 0 to "fall through"

Protected Member Functions

- void [set_pid](#) (const float p[3])

7.121.2 Constructor & Destructor Documentation

7.121.2.1 OutputPID::OutputPID () [inline]

Constructor.

Definition at line 8 of file OutputPID.h.

References [pid](#), and [weight](#).

7.121.2.2 OutputPID::OutputPID (const float p[3]) [inline]

Constructor.

Definition at line 9 of file OutputPID.h.

References [set_pid\(\)](#), and [weight](#).

7.121.2.3 OutputPID::OutputPID (const float p[3], float w) [inline]

Constructor.

Definition at line 10 of file OutputPID.h.

References [set_pid\(\)](#), and [weight](#).

7.121.2.4 OutputPID::OutputPID (const [OutputPID](#) & a, const [OutputPID](#) & b, float w) [inline]

Constructor, see [set\(a,b,w\)](#).

Definition at line 11 of file OutputPID.h.

References [set\(\)](#), and [weight](#).

7.121.3 Member Function Documentation

7.121.3.1 void OutputPID::set (const [OutputPID](#) & a, const [OutputPID](#) & b, float w) [inline]

sets the value to a weighted average of *a* and *b* (higher *w*, more *a*)

Definition at line 16 of file OutputPID.h.

References pid, and weight.

7.121.3.2 void OutputPID::set (const float *p*[3], float *w* = 1) [inline]

sets the value to *v* and weight to *w*

Definition at line 13 of file OutputPID.h.

References set_pid(), and weight.

7.121.3.3 void OutputPID::set_pid (const float *p*[3]) [inline, protected]

< handy utility function

Definition at line 28 of file OutputPID.h.

References pid.

7.121.3.4 void OutputPID::unset () [inline]

sets value and weight to 0 (same as assigning ::unusedJoint)

Definition at line 22 of file OutputPID.h.

References weight.

7.121.4 Member Data Documentation

7.121.4.1 float [OutputPID::pid](#)[3]

pid value of the output

Definition at line 24 of file OutputPID.h.

7.121.4.2 float [OutputPID::weight](#)

weight to be used in averaging, 0 to "fall through"

Definition at line 25 of file OutputPID.h.

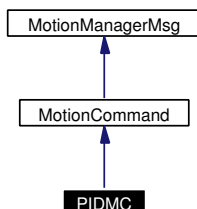
The documentation for this class was generated from the following file:

- [OutputPID.h](#)

7.122 PIDMC Class Reference

```
#include <PIDMC.h>
```

Inheritance diagram for PIDMC:



7.122.1 Detailed Description

A nice little [MotionCommand](#) for manually manipulating the PID values.

This can be a one shot deal if you set the autoprune flag as you're adding it (which is the default)

Definition at line 11 of file PIDMC.h.

Public Member Functions

- [PIDMC](#) ()
Constructor, uses default PIDs and 0 weight for all.
- [PIDMC](#) (float powerlevel, float w=1)
Constructor, sets general power level of all.
- [PIDMC](#) (unsigned int low, unsigned int high, float powerlevel, float w=1)
Constructor, sets general power level of a range of joints, uses default and 0 weight for others.
- virtual [~PIDMC](#) ()
Destructor.
- void [setDefault](#)s (float weight=1)
Sets the PIDs to the defaults specified in [RobotInfo](#).
- void [setJointPowerLevel](#) (unsigned int i, float p, float w=1)

Sets the PIDs to a percentage of default for a given joint, and sets weight.

- void [setAllPowerLevel](#) (float p, float w=1)
Sets the PIDs to a percentage of default for all joints.
- void [setRangePowerLevel](#) (unsigned int low, unsigned int high, float p, float w=1)
Sets a range of joints' PIDs to a given power level and weight.
- void [setPID](#) (unsigned int i, const [OutputPID](#) &pid)
Use this to set the PID value and weight.
- [OutputPID](#) & [getPID](#) (unsigned int i)
Use this if you want to double check the PID you set.
- const [OutputPID](#) & [getPID](#) (unsigned int i) const
Use this if you want to double check the PID you set.
- virtual int [updateOutputs](#) ()
is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)
- virtual int [isDirty](#) ()
not used by [MotionManager](#) at the moment, but could be used to reduce recomputation, and you may find it useful
- virtual int [isAlive](#) ()
used to prune "dead" motions from the [MotionManager](#)

Static Protected Member Functions

- bool [isPID](#) (unsigned int i)
returns true if the output i is a PID joint

Protected Attributes

- bool [dirty](#)
true if there are changes that have not been picked up by Motion

- [OutputPID PIDs](#) [NumPIDJoints]
the PIDs being requested

7.122.2 Constructor & Destructor Documentation

7.122.2.1 PIDMC::PIDMC () [inline]

Constructor, uses default PIDs and 0 weight for all.

Definition at line 14 of file PIDMC.h.

References `dirty`, and `setDefault()`.

7.122.2.2 PIDMC::PIDMC (float *powerlevel*, float *w* = 1) [inline]

Constructor, sets general power level of all.

Definition at line 19 of file PIDMC.h.

References `dirty`, and `setAllPowerLevel()`.

7.122.2.3 PIDMC::PIDMC (unsigned int *low*, unsigned int *high*, float *powerlevel*, float *w* = 1) [inline]

Constructor, sets general power level of a range of joints, uses default and 0 weight for others.

Definition at line 23 of file PIDMC.h.

References `dirty`, `ERS210Info::NumPIDJoints`, `ERS210Info::PIDJointOffset`, and `setRangePowerLevel()`.

7.122.2.4 virtual PIDMC::~PIDMC () [inline, virtual]

Destructor.

Definition at line 29 of file PIDMC.h.

7.122.3 Member Function Documentation

7.122.3.1 const [OutputPID](#)& PIDMC::getPID (unsigned int *i*) const [inline]

Use this if you want to double check the PID you set.

Definition at line 92 of file PIDMC.h.

References ERS210Info::PIDJointOffset, and PIDs.

7.122.3.2 **OutputPID& PIDMC::getPID (unsigned int *i*)** [inline]

Use this if you want to double check the PID you set.

Definition at line 87 of file PIDMC.h.

References ERS210Info::PIDJointOffset, and PIDs.

7.122.3.3 **virtual int PIDMC::isAlive ()** [inline, virtual]

used to prune "dead" motions from the [MotionManager](#)

note that a motion could be "paused" or inactive and therefore not dirty, but still alive, biding its time to "strike" ;)

Returns:

zero if the motion is still processing, non-zero otherwise

Implements [MotionCommand](#).

Definition at line 39 of file PIDMC.h.

References dirty.

7.122.3.4 **virtual int PIDMC::isDirty ()** [inline, virtual]

not used by [MotionManager](#) at the moment, but could be used to reduce recomputation, and you may find it useful

Returns:

zero if none of the commands have changed since last getJointCmd(), else non-zero

Implements [MotionCommand](#).

Definition at line 38 of file PIDMC.h.

References dirty.

7.122.3.5 **bool PIDMC::isPID (unsigned int *i*)** [inline, static, protected]

returns true if the output *i* is a PID joint

Definition at line 98 of file PIDMC.h.

References ERS210Info::NumPIDJoints, and ERS210Info::PIDJointOffset.

7.122.3.6 void PIDMC::setAllPowerLevel (float *p*, float *w* = 1) [inline]

Sets the PIDs to a percentage of default for all joints.

Definition at line 57 of file PIDMC.h.

References ERS210Info::DefaultPIDs, dirty, ERS210Info::NumPIDJoints, OutputPID::pid, PIDs, and OutputPID::weight.

7.122.3.7 void PIDMC::setDefaults (float *weight* = 1) [inline]

Sets the PIDs to the defaults specified in [RobotInfo](#).

Definition at line 43 of file PIDMC.h.

References setAllPowerLevel().

7.122.3.8 void PIDMC::setJointPowerLevel (unsigned int *i*, float *p*, float *w* = 1) [inline]

Sets the PIDs to a percentage of default for a given joint, and sets weight.

Definition at line 48 of file PIDMC.h.

References ERS210Info::DefaultPIDs, dirty, OutputPID::pid, ERS210Info::PIDJointOffset, PIDs, and OutputPID::weight.

7.122.3.9 void PIDMC::setPID (unsigned int *i*, const [OutputPID](#) & *pid*) [inline]

Use this to set the PID value and weight.

Definition at line 80 of file PIDMC.h.

References dirty, ERS210Info::PIDJointOffset, and PIDs.

7.122.3.10 void PIDMC::setRangePowerLevel (unsigned int *low*, unsigned int *high*, float *p*, float *w* = 1) [inline]

Sets a range of joints' PIDs to a given power level and weight.

Definition at line 67 of file PIDMC.h.

References `ERS210Info::DefaultPIDs`, `dirty`, `OutputPID::pid`, `ERS210Info::PIDJointOffset`, `PIDs`, and `OutputPID::weight`.

7.122.3.11 `virtual int PIDMC::updateOutputs ()` `[inline, virtual]`

is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)

Returns:

zero if no changes were made, non-zero otherwise

See also:

`RobotInfo::NumFrames`
`RobotInfo::FrameTime`

Implements [MotionCommand](#).

Definition at line 33 of file `PIDMC.h`.

References `dirty`, `motman`, `ERS210Info::NumPIDJoints`, `ERS210Info::PIDJointOffset`, `PIDs`, and `MotionManager::setOutput()`.

7.122.4 Member Data Documentation

7.122.4.1 `bool PIDMC::dirty` `[protected]`

true if there are changes that have not been picked up by Motion

Definition at line 102 of file `PIDMC.h`.

7.122.4.2 `OutputPID PIDMC::PIDs[NumPIDJoints]` `[protected]`

the PIDs being requested

Definition at line 103 of file `PIDMC.h`.

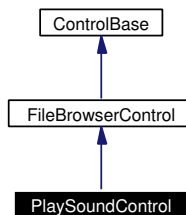
The documentation for this class was generated from the following file:

- [PIDMC.h](#)

7.123 PlaySoundControl Class Reference

```
#include <PlaySoundControl.h>
```

Inheritance diagram for PlaySoundControl:



7.123.1 Detailed Description

Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...).

Definition at line 9 of file PlaySoundControl.h.

Public Member Functions

- [PlaySoundControl](#) (const std::string &n)
Constructor.
- virtual [~PlaySoundControl](#) ()
Destructor.

Protected Member Functions

- virtual [ControlBase](#) * [selectedFile](#) (const std::string &f)
does the actual loading of the [MotionSequence](#)

7.123.2 Constructor & Destructor Documentation

7.123.2.1 PlaySoundControl::PlaySoundControl (const std::string & n) [inline]

Constructor.

Definition at line 12 of file PlaySoundControl.h.

References `config`, `FileBrowserControl::root`, and `FileBrowserControl::setFilter()`.

7.123.2.2 `virtual PlaySoundControl::~~PlaySoundControl () [inline, virtual]`

Destructor.

Definition at line 18 of file PlaySoundControl.h.

7.123.3 Member Function Documentation

7.123.3.1 `virtual ControlBase* PlaySoundControl::selectedFile (const std::string &f) [inline, protected, virtual]`

does the actual loading of the [MotionSequence](#)

Reimplemented from [FileBrowserControl](#).

Definition at line 22 of file PlaySoundControl.h.

References `SoundManager::PlayFile()`, `sndman`, and `SoundManager::StopPlay()`.

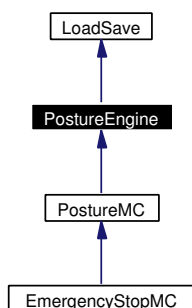
The documentation for this class was generated from the following file:

- [PlaySoundControl.h](#)

7.124 PostureEngine Class Reference

```
#include <PostureEngine.h>
```

Inheritance diagram for PostureEngine:



7.124.1 Detailed Description

A class for storing a set of positions and weights for all the outputs.

Handy for any class which wants to deal with setting joints and postures without writing a custom class

See also:

[PostureMC](#)

Definition at line 12 of file PostureEngine.h.

Public Member Functions

- [PostureEngine](#) ()
constructor
- [PostureEngine](#) (const char *filename)
constructor, loads a position from a file - not necessarily quick!
- virtual [~PostureEngine](#) ()
destructor
- virtual void [takeSnapshot](#) ()
sets the internal [cmds](#) to the current state of the outputs

- virtual void `clear` ()
sets all joints to unused
- virtual `PostureEngine` & `setOverlay` (const `PostureEngine` &pe)
*sets joints of this to all joints of pe which are not equal to unused (layers pe over this)
stores into this*
- virtual `PostureEngine` `createOverlay` (const `PostureEngine` &pe) const
*sets joints of this to all joints of pe which are not equal to unused (layers pe over this)
returns new PostureEngine*
- virtual `PostureEngine` & `setUnderlay` (const `PostureEngine` &pe)
sets joints of this which are equal to unused to pe, (layers this over pe) stores into this
- virtual `PostureEngine` `createUnderlay` (const `PostureEngine` &pe) const
*sets joints of this which are equal to unused to pe, (layers this over pe) returns new
PostureEngine*
- virtual `PostureEngine` & `setAverage` (const `PostureEngine` &pe, float w=0.5)
*computes a weighted average of this vs. pe, w being the weight towards pe (so w==1
just copies pe)*
- virtual `PostureEngine` `createAverage` (const `PostureEngine` &pe, float w=0.5)
const
*computes a weighted average of this vs. pe, w being the weight towards pe (so w==1
just copies pe)*
- virtual `PostureEngine` & `setCombine` (const `PostureEngine` &pe)
*computes a weighted average of this vs. pe, using the weight values of the joints,
storing the total weight in the result's weight value*
- virtual `PostureEngine` `createCombine` (const `PostureEngine` &pe) const
*computes a weighted average of this vs. pe, using the weight values of the joints,
storing the total weight in the result's weight value*
- float `diff` (const `PostureEngine` &pe) const
*returns the sum squared error between this and pe's output values, but only between
outputs which are both not unused*
- float `avgdiff` (const `PostureEngine` &pe) const
*returns the average sum squared error between this and pe's output values for outputs
which are both not unused*
- float `maxdiff` (const `PostureEngine` &pe) const

returns the max sum squared error between this and pe's output values for outputs which are both not unused

Output Accessors

NOT VIRTUAL! You should be able to call this to set outputs without checking out, just a peekMotion(). Theoretically.

- **PostureEngine** & **setOutputCmd** (unsigned int i, const **OutputCmd** &c)
*sets output i to **OutputCmd** c, returns *this so you can chain them*
- **OutputCmd** & **getOutputCmd** (unsigned int i)
returns output i, returns a reference so you can also set "through" this call.
- const **OutputCmd** & **getOutputCmd** (unsigned int i) const
returns output i

LoadSave

*Uses **LoadSave** interface so you can load/save to files, uses a human-readable storage format*

- virtual unsigned int **getBinSize** () const
calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.
- virtual unsigned int **LoadBuffer** (const char buf[], unsigned int len)
Load from a saved buffer.
- virtual unsigned int **SaveBuffer** (char buf[], unsigned int len) const
Save to a given buffer.

Static Protected Member Functions

- bool **ChkAdvance** (int res, const char **buf, unsigned int *len, const char *msg)
*used by **LoadBuffer**()/**SaveBuffer**(), checks to see if the amount read/written (res) is nonzero, increments buf, decrements len, or displays msg if is zero*
- bool **ChkAdvance** (int res, const char **buf, unsigned int *len, const char *msg, int arg1)
*used by **LoadBuffer**()/**SaveBuffer**(), checks to see if the amount read/written (res) is nonzero, increments buf, decrements len, or displays msg with arg1 if is zero*

Protected Attributes

- [OutputCmd cmds](#) [NumOutputs]

the table of outputs' values and weights, can be accessed through [setOutputCmd\(\)](#) and [getOutputCmd\(\)](#)

7.124.2 Constructor & Destructor Documentation

7.124.2.1 PostureEngine::PostureEngine () [inline]

constructor

Definition at line 15 of file PostureEngine.h.

7.124.2.2 PostureEngine::PostureEngine (const char *filename) [inline]

constructor, loads a position from a file - not necessarily quick!

Todo

might want to make a library of common positions so they don't have to be loaded repeatedly from memstick

Definition at line 18 of file PostureEngine.h.

References [LoadSave::LoadFile\(\)](#).

7.124.2.3 PostureEngine::~PostureEngine () [virtual]

destructor

Definition at line 5 of file PostureEngine.cc.

7.124.3 Member Function Documentation

7.124.3.1 float PostureEngine::avgdiff (const [PostureEngine](#) &pe) const

returns the average sum squared error between this and pe's output values for outputs which are both not unused

Todo

create a version which looks at weights? This doesn't use them.

Definition at line 95 of file PostureEngine.cc.

References `cmds`, `ERS210Info::NumOutputs`, `OutputCmd::value`, and `OutputCmd::weight`.

7.124.3.2 `bool PostureEngine::ChkAdvance (int res, const char ** buf, unsigned int * len, const char * msg, int arg1)` `[static, protected]`

used by [LoadBuffer\(\)](#)/[SaveBuffer\(\)](#), checks to see if the amount read/written (*res*) is nonzero, increments *buf*, decrements *len*, or displays *msg* with *arg1* if *is* zero

Definition at line 216 of file PostureEngine.cc.

7.124.3.3 `bool PostureEngine::ChkAdvance (int res, const char ** buf, unsigned int * len, const char * msg)` `[static, protected]`

used by [LoadBuffer\(\)](#)/[SaveBuffer\(\)](#), checks to see if the amount read/written (*res*) is nonzero, increments *buf*, decrements *len*, or displays *msg* if *is* zero

Definition at line 205 of file PostureEngine.cc.

7.124.3.4 `void PostureEngine::clear ()` `[virtual]`

sets all joints to unused

Reimplemented in [PostureMC](#).

Definition at line 12 of file PostureEngine.cc.

References `cmds`, `ERS210Info::NumOutputs`, and `OutputCmd::unset()`.

7.124.3.5 `PostureEngine PostureEngine::createAverage (const PostureEngine & pe, float w = 0.5) const` `[virtual]`

computes a weighted average of this vs. *pe*, *w* being the weight towards *pe* (so *w*==1 just copies *pe*)

joints being averaged with weight<=0 have their weights averaged, but not their values (so an output can crossfade properly)

Parameters:

pe the other PostureEngine

w amount to weight towards *pe*

- if *w* < .001, nothing is done
- if *w* > .999, a straight copy of *pe* occurs (sets joints to unused properly at end of fade)

- .001 and .999 is used instead of 0 and 1 to allow for slight addition errors in a loop (if using repeated additions of a delta value instead of repeated divisions)

Returns:

a new posture containing the results

Definition at line 69 of file PostureEngine.cc.

References `setAverage()`.

7.124.3.6 **PostureEngine** **PostureEngine::createCombine** (const **PostureEngine** & *pe*) const [virtual]

computes a weighted average of this vs. *pe*, using the weight values of the joints, storing the total weight in the result's weight value

Definition at line 80 of file PostureEngine.cc.

References `setCombine()`.

7.124.3.7 **PostureEngine** **PostureEngine::createOverlay** (const **PostureEngine** & *pe*) const [virtual]

sets joints of this to all joints of *pe* which are not equal to unused (layers *pe* over this) returns new PostureEngine

Definition at line 23 of file PostureEngine.cc.

References `setOverlay()`.

7.124.3.8 **PostureEngine** **PostureEngine::createUnderlay** (const **PostureEngine** & *pe*) const [virtual]

sets joints of this which are equal to unused to *pe*, (layers this over *pe*) returns new PostureEngine

Definition at line 33 of file PostureEngine.cc.

References `setUnderlay()`.

7.124.3.9 float **PostureEngine::diff** (const **PostureEngine** & *pe*) const

returns the sum squared error between this and *pe*'s output values, but only between outputs which are both not unused

Todo

create a version which looks at weights? This doesn't use them.

Definition at line 85 of file PostureEngine.cc.

References `cmds`, `ERS210Info::NumOutputs`, `OutputCmd::value`, and `OutputCmd::weight`.

7.124.3.10 unsigned int PostureEngine::getBinSize () const [virtual]

calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.

Returns:

number of bytes read/written, 0 if error (or empty)

Implements [LoadSave](#).

Definition at line 119 of file PostureEngine.cc.

References `cmds`, `ERS210Info::NumOutputs`, `ERS210Info::outputNames`, and `OutputCmd::weight`.

7.124.3.11 const OutputCmd& PostureEngine::getOutputCmd (unsigned int i)
const [inline]

returns output *i*

Definition at line 64 of file PostureEngine.h.

References `cmds`.

7.124.3.12 OutputCmd& PostureEngine::getOutputCmd (unsigned int i)
[inline]

returns output *i*, returns a reference so you can also set "through" this call.

Definition at line 63 of file PostureEngine.h.

References `cmds`.

7.124.3.13 unsigned int PostureEngine::LoadBuffer (const char buf[], unsigned int len) [virtual]

Load from a saved buffer.

Parameters:

buf pointer to the memory where you should begin loading
len length of *buf* available (this isn't all yours, might be more stuff saved after yours)

Returns:

the number of bytes actually used

Implements [LoadSave](#).

Reimplemented in [PostureMC](#).

Definition at line 128 of file PostureEngine.cc.

References [ChkAdvance\(\)](#), [clear\(\)](#), [cmds](#), [ERS210Info::NumOutputs](#), [ERS210Info::outputNameLen](#), [ERS210Info::outputNames](#), and [OutputCmd::set\(\)](#).

7.124.3.14 float PostureEngine::maxdiff (const [PostureEngine](#) &pe) const

returns the max sum squared error between this and pe's output values for outputs which are both not unused

Todo

create a version which looks at weights? This doesn't use them.

Definition at line 107 of file PostureEngine.cc.

References [cmds](#), [ERS210Info::NumOutputs](#), [OutputCmd::value](#), and [OutputCmd::weight](#).

7.124.3.15 unsigned int PostureEngine::SaveBuffer (char *buf*[], unsigned int *len*) const [virtual]

Save to a given buffer.

Parameters:

buf pointer to the memory where you should begin writing
len length of *buf* available. (this isn't all yours, constrain yourself to what you returned in [getBinSize\(\)](#))

Returns:

the number of bytes actually used

Implements [LoadSave](#).

Definition at line 179 of file PostureEngine.cc.

References [ChkAdvance\(\)](#), [cmds](#), [ERS210Info::NumOutputs](#), [ERS210Info::outputNames](#), and [OutputCmd::weight](#).

7.124.3.16 [PostureEngine](#) & [PostureEngine::setAverage](#) (const [PostureEngine](#) & *pe*, float *w* = 0.5) [[virtual](#)]

computes a weighted average of this vs. *pe*, *w* being the weight towards *pe* (so *w*==1 just copies *pe*)

joints being averaged with `::unusedJoint` have their weights averaged, but not their values (so an output can crossfade properly)

Parameters:

pe the other [PostureEngine](#)

w amount to weight towards *pe*

- if *w* < .001, nothing is done
- if *w* > .999, a straight copy of *pe* occurs (sets joints to unused properly at end of fade)
- .001 and .999 is used instead of 0 and 1 to allow for slight addition errors in a loop (if using repeated additions of a delta value instead of repeated divisions)

Returns:

`*this`, stores results into this

Reimplemented in [PostureMC](#).

Definition at line 45 of file [PostureEngine.cc](#).

References [cmds](#), [ERS210Info::NumOutputs](#), [OutputCmd::set\(\)](#), [OutputCmd::value](#), and [OutputCmd::weight](#).

7.124.3.17 [PostureEngine](#) & [PostureEngine::setCombine](#) (const [PostureEngine](#) & *pe*) [[virtual](#)]

computes a weighted average of this vs. *pe*, using the weight values of the joints, storing the total weight in the result's weight value

Reimplemented in [PostureMC](#).

Definition at line 73 of file [PostureEngine.cc](#).

References [cmds](#), [ERS210Info::NumOutputs](#), [OutputCmd::value](#), and [OutputCmd::weight](#).

7.124.3.18 [PostureEngine](#) & [PostureEngine::setOutputCmd](#) (unsigned int *i*, const [OutputCmd](#) & *c*) [[inline](#)]

sets output *i* to [OutputCmd](#) *c*, returns `*this` so you can chain them

Reimplemented in [PostureMC](#).

Definition at line 62 of file PostureEngine.h.

References [cmds](#).

7.124.3.19 [PostureEngine](#) & [PostureEngine::setOverlay](#) (const [PostureEngine](#) & *pe*) [virtual]

sets joints of this to all joints of *pe* which are not equal to unused (layers *pe* over this) stores into this

Reimplemented in [PostureMC](#).

Definition at line 17 of file PostureEngine.cc.

References [cmds](#), [ERS210Info::NumOutputs](#), and [OutputCmd::weight](#).

7.124.3.20 [PostureEngine](#) & [PostureEngine::setUnderlay](#) (const [PostureEngine](#) & *pe*) [virtual]

sets joints of this which are equal to unused to *pe*, (layers this over *pe*) stores into this

Reimplemented in [PostureMC](#).

Definition at line 27 of file PostureEngine.cc.

References [cmds](#), [ERS210Info::NumOutputs](#), and [OutputCmd::weight](#).

7.124.3.21 [void PostureEngine::takeSnapshot](#) () [virtual]

sets the internal [cmds](#) to the current state of the outputs

Reimplemented in [EmergencyStopMC](#), and [PostureMC](#).

Definition at line 7 of file PostureEngine.cc.

References [cmds](#), [ERS210Info::NumOutputs](#), [WorldState::outputs](#), [OutputCmd::set\(\)](#), and [state](#).

7.124.4 Member Data Documentation

7.124.4.1 [OutputCmd PostureEngine::cmds](#)[[NumOutputs](#)] [protected]

the table of outputs' values and weights, can be accessed through [setOutputCmd\(\)](#) and [getOutputCmd\(\)](#)

Definition at line 81 of file PostureEngine.h.

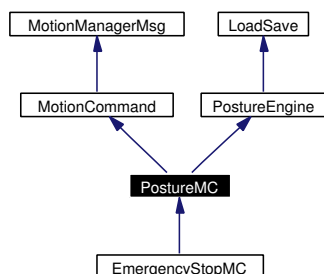
The documentation for this class was generated from the following files:

- [PostureEngine.h](#)
- [PostureEngine.cc](#)

7.125 PostureMC Class Reference

```
#include <PostureMC.h>
```

Inheritance diagram for PostureMC:



7.125.1 Detailed Description

a [MotionCommand](#) shell for [PostureEngine](#)

Note:

PostureMC does not autoprune by default - this is contrary to the default for other MotionCommands

Definition at line 11 of file PostureMC.h.

Public Member Functions

- [PostureMC](#) ()
constructor
- [PostureMC](#) (const char *filename)
constructor, loads from filename
- virtual [~PostureMC](#) ()
destructor

New Stuff

- [PostureMC](#) & [setDirty](#) (bool d=true)

call this if you call [PostureEngine::setOutputCmd\(\)](#), that doesn't know about dirty flags

- bool [isDirty](#) () const
if you want to check the dirty flag
- virtual [PostureMC](#) & [setTolerance](#) (float t)
*sets [tolerance](#), returns *this*
- virtual float [getTolerance](#) ()
returns [tolerance](#)

MotionCommand Stuff

- virtual int [updateOutputs](#) ()
is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)
- virtual int [isDirty](#) ()
not used by [MotionManager](#) at the moment, but could be used to reduce recomputation, and you may find it useful
- virtual int [isAlive](#) ()
returns non-zero (true) if [PostureEngine::maxdiff\(\)](#) between this and the current position is over [tolerance](#)

PostureEngine Stuff

Had to override stuff to manage a dirty flag

- virtual void [takeSnapshot](#) ()
sets the internal [cmds](#) to the current state of the outputs
- virtual void [clear](#) ()
sets all joints to unused
- virtual [PostureMC](#) & [setOverlay](#) (const [PostureEngine](#) &pe)
sets joints of this to all joints of pe which are not equal to unused (layers pe over this) stores into this
- virtual [PostureMC](#) & [setUnderlay](#) (const [PostureEngine](#) &pe)
sets joints of this which are equal to unused to pe, (layers this over pe) stores into this
- virtual [PostureMC](#) & [setAverage](#) (const [PostureEngine](#) &pe, float w=0.5)

computes a weighted average of this vs. pe, w being the weight towards pe (so w==1 just copies pe)

- virtual **PostureMC** & **setCombine** (const **PostureEngine** &pe)
computes a weighted average of this vs. pe, using the weight values of the joints, storing the total weight in the result's weight value
- **PostureMC** & **setOutputCmd** (unsigned int i, const **OutputCmd** &c)
*sets output i to **OutputCmd** c, returns *this so you can chain them*
- virtual unsigned int **LoadBuffer** (const char buf[], unsigned int len)
Load from a saved buffer.

Protected Attributes

- bool **dirty**
*true if changes have been made since last **updateOutputs()***
- float **tolerance**
*when autopruning, if the **maxdiff()** of this posture and the robot's current position is below this value, **isAlive()** will be false, defaults to 0.01 (5.7 degree error)*

7.125.2 Constructor & Destructor Documentation

7.125.2.1 **PostureMC::PostureMC ()** [inline]

constructor

Definition at line 14 of file PostureMC.h.

References **dirty**, **MotionCommand::setAutoPrune()**, and **tolerance**.

7.125.2.2 **PostureMC::PostureMC (const char **filename*)** [inline]

constructor, loads from *filename*

Definition at line 16 of file PostureMC.h.

References **dirty**, **MotionCommand::setAutoPrune()**, and **tolerance**.

7.125.2.3 **virtual PostureMC::~~PostureMC ()** [inline, virtual]

destructor

Definition at line 18 of file PostureMC.h.

7.125.3 Member Function Documentation

7.125.3.1 `virtual void PostureMC::clear ()` [inline, virtual]

sets all joints to unused

Reimplemented from [PostureEngine](#).

Definition at line 59 of file PostureMC.h.

References [PostureEngine::clear\(\)](#), and [dirty](#).

7.125.3.2 `virtual float PostureMC::getTolerance ()` [inline, virtual]

returns [tolerance](#)

Definition at line 30 of file PostureMC.h.

References [tolerance](#).

7.125.3.3 `virtual int PostureMC::isAlive ()` [inline, virtual]

returns non-zero (true) if [PostureEngine::maxdiff\(\)](#) between this and the current position is over [tolerance](#)

This is handy so you can set to have the robot go to a position and then automatically remove the [MotionCommand](#) when it gets there - but beware fighting Postures which average out and neither succeeds

Implements [MotionCommand](#).

Definition at line 47 of file PostureMC.h.

References [dirty](#), [MotionCommand::getAutoPrune\(\)](#), [PostureEngine::maxdiff\(\)](#), [PostureEngine::takeSnapshot\(\)](#), and [tolerance](#).

7.125.3.4 `virtual int PostureMC::isDirty ()` [inline, virtual]

not used by [MotionManager](#) at the moment, but could be used to reduce recomputation, and you may find it useful

Returns:

zero if none of the commands have changed since last [getJointCmd\(\)](#), else non-zero

Implements [MotionCommand](#).

Definition at line 41 of file PostureMC.h.

References [dirty](#).

7.125.3.5 **bool PostureMC::isDirty () const** [inline]

if you want to check the dirty flag

Definition at line 28 of file PostureMC.h.

References [dirty](#).

7.125.3.6 **virtual unsigned int PostureMC::LoadBuffer (const char *buf*[], unsigned int *len*)** [inline, virtual]

Load from a saved buffer.

Parameters:

buf pointer to the memory where you should begin loading

len length of *buf* available (this isn't all yours, might be more stuff saved after yours)

Returns:

the number of bytes actually used

Reimplemented from [PostureEngine](#).

Definition at line 65 of file PostureMC.h.

References [dirty](#), and [PostureEngine::LoadBuffer\(\)](#).

7.125.3.7 **virtual [PostureMC](#)& PostureMC::setAverage (const [PostureEngine](#) & *pe*, float *w* = 0.5)** [inline, virtual]

computes a weighted average of this vs. *pe*, *w* being the weight towards *pe* (so *w*==1 just copies *pe*)

joints being averaged with [::unusedJoint](#) have their weights averaged, but not their values (so an output can crossfade properly)

Parameters:

pe the other [PostureEngine](#)

w amount to weight towards *pe*

- if *w* < .001, nothing is done

- if $w > .999$, a straight copy of pe occurs (sets joints to unused properly at end of fade)
- .001 and .999 is used instead of 0 and 1 to allow for slight addition errors in a loop (if using repeated additions of a delta value instead of repeated divisions)

Returns:

`*this`, stores results into this

Reimplemented from [PostureEngine](#).

Definition at line 62 of file PostureMC.h.

References `dirty`, and `PostureEngine::setAverage()`.

7.125.3.8 **virtual PostureMC& PostureMC::setCombine (const PostureEngine &pe) [inline, virtual]**

computes a weighted average of this vs. pe , using the weight values of the joints, storing the total weight in the result's weight value

Reimplemented from [PostureEngine](#).

Definition at line 63 of file PostureMC.h.

References `dirty`, and `PostureEngine::setCombine()`.

7.125.3.9 **PostureMC& PostureMC::setDirty (bool d = true) [inline]**

call this if you call [PostureEngine::setOutputCmd\(\)](#), that doesn't know about dirty flags

Definition at line 27 of file PostureMC.h.

References `dirty`.

7.125.3.10 **PostureMC& PostureMC::setOutputCmd (unsigned int i, const OutputCmd &c) [inline]**

sets output i to [OutputCmd](#) c , returns `*this` so you can chain them

Reimplemented from [PostureEngine](#).

Definition at line 64 of file PostureMC.h.

References `dirty`, and `PostureEngine::setOutputCmd()`.

7.125.3.11 `virtual PostureMC& PostureMC::setOverlay (const PostureEngine & pe)` [inline, virtual]

sets joints of this to all joints of *pe* which are not equal to unused (layers *pe* over this) stores into this

Reimplemented from [PostureEngine](#).

Definition at line 60 of file PostureMC.h.

References [dirty](#), and [PostureEngine::setOverlay\(\)](#).

7.125.3.12 `virtual PostureMC& PostureMC::setTolerance (float t)` [inline, virtual]

sets [tolerance](#), returns *this

Definition at line 29 of file PostureMC.h.

References [tolerance](#).

7.125.3.13 `virtual PostureMC& PostureMC::setUnderlay (const PostureEngine & pe)` [inline, virtual]

sets joints of this which are equal to unused to *pe*, (layers this over *pe*) stores into this

Reimplemented from [PostureEngine](#).

Definition at line 61 of file PostureMC.h.

References [dirty](#), and [PostureEngine::setUnderlay\(\)](#).

7.125.3.14 `virtual void PostureMC::takeSnapshot ()` [inline, virtual]

sets the internal [cmds](#) to the current state of the outputs

Reimplemented from [PostureEngine](#).

Reimplemented in [EmergencyStopMC](#).

Definition at line 58 of file PostureMC.h.

References [dirty](#), and [PostureEngine::takeSnapshot\(\)](#).

7.125.3.15 `virtual int PostureMC::updateOutputs ()` [inline, virtual]

is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)

Returns:

zero if no changes were made, non-zero otherwise

See also:

RobotInfo::NumFrames
RobotInfo::FrameTime

Implements [MotionCommand](#).

Reimplemented in [EmergencyStopMC](#).

Definition at line 35 of file PostureMC.h.

References [PostureEngine::cmds](#), [dirty](#), [motman](#), [ERS210Info::NumOutputs](#), [MotionManager::setOutput\(\)](#), and [OutputCmd::weight](#).

7.125.4 Member Data Documentation

7.125.4.1 bool [PostureMC::dirty](#) [protected]

true if changes have been made since last [updateOutputs\(\)](#)

Definition at line 69 of file PostureMC.h.

7.125.4.2 float [PostureMC::tolerance](#) [protected]

when autopruning, if the [maxdiff\(\)](#) of this posture and the robot's current position is below this value, [isAlive\(\)](#) will be false, defaults to 0.01 (5.7 degree error)

Definition at line 70 of file PostureMC.h.

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

- [PostureMC.h](#)

7.126 ProcessID Class Reference

```
#include <ProcessID.h>
```

7.126.1 Detailed Description

this is a class instead of a namespace so i can limit write access of the ID value to the OObjects

Definition at line 6 of file ProcessID.h.

Public Types

- enum [ProcessID_t](#) { [MainProcess](#), [MotionProcess](#), [SoundProcess](#), [NumProcesses](#) }

Holds ID number for each process.

Static Public Member Functions

- [ProcessID_t](#) [getID](#) ()

returns process's ID number, or if within a virtual function on a shared object, the process which created it (annoying)

Static Private Member Functions

- void [setID](#) ([ProcessID_t](#) id)

sets the ID during init

Static Private Attributes

- [ProcessID_t](#) [ID](#) = [ProcessID::NumProcesses](#)

holds ID number

Friends

- class [MMCombo](#)

so that it can set the ID during init

- class [SoundPlay](#)
so that it can set the ID during init

7.126.2 Member Enumeration Documentation

7.126.2.1 enum [ProcessID::ProcessID_t](#)

Holds ID number for each process.

Enumeration values:

- MainProcess** MainObj process.
- MotionProcess** MotoObj process.
- SoundProcess** [SoundPlay](#) process.
- NumProcesses** count of processes

Definition at line 9 of file ProcessID.h.

7.126.3 Member Function Documentation

7.126.3.1 [ProcessID_t](#) [ProcessID::getID\(\)](#) [`inline`, `static`]

returns process's ID number, or if within a virtual function on a shared object, the process which created it (annoying)

Definition at line 16 of file ProcessID.h.

References ID, and ProcessID.t.

7.126.3.2 `void` [ProcessID::setID](#) ([ProcessID_t](#) *id*) [`inline`, `static`, `private`]

sets the ID during init

Definition at line 21 of file ProcessID.h.

References ID.

7.126.4 Friends And Related Function Documentation

7.126.4.1 `friend` class [MMCombo](#) [`friend`]

so that it can set the ID during init

Definition at line 19 of file ProcessID.h.

7.126.4.2 friend class [SoundPlay](#) [[friend](#)]

so that it can set the ID during init

Definition at line 20 of file ProcessID.h.

7.126.5 Member Data Documentation

7.126.5.1 [ProcessID::ProcessID_t](#) [ProcessID::ID](#) = [ProcessID::NumProcesses](#) [[static](#), [private](#)]

holds ID number

Definition at line 3 of file ProcessID.cc.

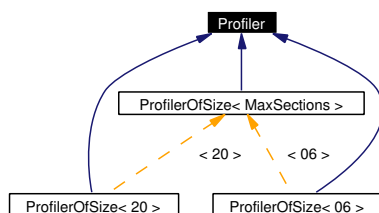
The documentation for this class was generated from the following files:

- [ProcessID.h](#)
- [ProcessID.cc](#)

7.127 Profiler Class Reference

```
#include <Profiler.h>
```

Inheritance diagram for Profiler:



7.127.1 Detailed Description

Manages a hierarchy of timers for profiling time spent in code, gives microsecond resolution.

Doesn't use any pointers so it's safe to put this in shared memory regions.

That's handy so one process can collate all the profiling information across processes to give a summary report to the user.

Example usage: (two different methods, f() or g())

```

ProfilerOfSize<2> prof; //A global manager for all the sections

void f() {
    static unsigned int id=prof.getNewID("f"); // *** YOU NEED THIS LINE for each section to profile
    Profiler::Timer timer(id,&prof);           // *** YOU NEED THIS LINE for each section to profile
    //...
    if(rand()>RAND_MAX/2)
        return; // destruction of timer occurs automatically!
    //...
} // if we didn't hit the return, timer will otherwise destruct here!

void g() {
    PROFSECTION("g",prof); // Could also use this macro instead (recommended if you're
    //...                  // not doing something fancy like conditional timers)
    f(); // will note f's time as g's child time
    //...
}
  
```

The idea is similar to that used by [MMAccessor](#). If you want to profile a section at smaller resolution than a function, you can use tricks shown in MMAccessor's documentation to limit the timer's scope.

Here were the constraints I set for myself:

- Processes can read each other's Profilers - so must be able to live in shared memory
This is so one process can generate a report on performance of the entire system at once
- Flexible memory footprint
MainObject will probably have a lot of subsections. MotionObject won't. Since [SectionInfo](#) takes some significant memory space, we don't want to force MotionObject to needlessly make a lot of them.
- Flexible usage - can have a single generic global, as well as creating multiple
- Fast - don't want to kill performance of profiled sections, or throw off reported results

Concessions made:

- Sections are not dynamically allocated
- Sections within a Profiler are mutually exclusive (otherwise curSection won't be reliable)
- Results don't try to take affects of pre-emptive multitasking into account.

Global readability is first priority since generating reports is the primary usage, thus we have to be able to handle being in shared memory space. This means no virtual functions and no pointer storage. Unfortunately, this makes the second constraint rather difficult.

Run-time dynamic allocation is right out. But the number of sections is set at compile time anyway, so it should be OK to set this at compile time, using a template parameter.

That gets us 99% of the way there, but it can be burdensome to use this since the template means there's no constant type for all profilers - you can't have a generic Profiler type if it's templated - you would have to know the size of the profiler you're referring to.

That kind of brings in the third constraint... Instead of accepting a single global, I decided to make a general base (Profiler) and then a templated subclass to hold the bulk data section. This has the nice side affect of not having to specify the bulk of the code in the header, but has the downside that accessing the info stored in the subclass from the super class is very much a hack. If you think you can get around this, good luck!

Note:

~Profiler() isn't virtual - that's on purpose.

This could be made much prettier if we didn't care about the virtual function-shared memory problems... sigh

Definition at line 80 of file Profiler.h.

Public Member Functions

- unsigned int [getNewID](#) (const char *name)
call this to get a new ID number
- float * [getBuckets](#) ()
returns the bucket boundaries
- std::string [report](#) ()
outputs profiling information
- void [reset](#) ()
resets profiling information
- [SectionInfo](#) * [getInfos](#) ()
gets the actual storage area of the SectionInfo's

Static Public Member Functions

- void [initBuckets](#) ()
called during process init (before any profiled sections)

Public Attributes

- unsigned int [curSection](#)
the current timer
- [TimeET](#) [startTime](#)
time of beginning profiling
- float [gamma](#)
gamma to use with exponential averages (1 to freeze, 0 to set to last)
- const unsigned int [maxSections](#)
so we can read the size of the infos array back again at runtime
- unsigned int [sectionsUsed](#)
the number of timer IDs which have been assigned

Static Public Attributes

- const unsigned int `MaxSectionNameLen` = 32
maximum length of names of timers, 32 chosen to make nice memory alignment
- const unsigned int `HistSize` = 32
number of slots in the histograms
- const unsigned int `HistTime` = 10*1000
the upper bound (exclusive) of the histograms, in milliseconds.
- const float `HistCurve` = 4.05
affects how linearly the buckets are distributed - 1 means linear, >1 causes higher resolution for short times

Protected Member Functions

- `Profiler` (unsigned int mx)
constructor, protected because you don't want to construct one of these - use `Profiler-OfSize<x>!`
- void `setCurrent` (`Timer` &tr)
called automatically by `Timer()` - sets the current timer
- void `finished` (`Timer` &tr)
called automatically by `~Timer()` - notes the specified timer as finished (doesn't check if the timer is actually the current one - don't screw up!)
- unsigned int `getBucket` (float t)
returns which bucket a time should go in, does a binary search over buckets (unless someone thinks a `log()` call would be faster...)

Static Protected Attributes

- float `buckets` [`HistSize`]
holds boundaries for each bucket
- unsigned int `infosOffset` = `reinterpret_cast<unsigned int>((static_cast<Profiler-OfSize<1>*>(NULL)) → infos)`
NASTY HACK - this is how we get around using virtual functions.

Friends

- class [Timer](#)

Only the Timer's should be calling [setCurrent\(\)](#) and [finished\(\)](#) upon the Timer's construction and destruction.

7.127.2 Constructor & Destructor Documentation

7.127.2.1 Profiler::Profiler (unsigned int *mx*) [protected]

constructor, protected because you don't want to construct one of these - use Profiler-OfSize<x>!

Definition at line 114 of file Profiler.cc.

7.127.3 Member Function Documentation

7.127.3.1 void Profiler::finished ([Timer](#) & *tr*) [protected]

called automatically by ~Timer() - notes the specified timer as finished (doesn't check if the timer is actually the current one - don't screw up!)

Definition at line 140 of file Profiler.cc.

References Profiler::Timer::id, Profiler::Timer::_parent, Profiler::SectionInfo::calls, curSection, Profiler::Timer::elapsed(), Profiler::SectionInfo::execExpAvg, Profiler::SectionInfo::execHist, gamma, getBucket(), getInfos(), HistSize, HistTime, Profiler::SectionInfo::totalTime, and TimeET::Value().

7.127.3.2 unsigned int Profiler::getBucket (float *t*) [inline, protected]

returns which bucket a time should go in, does a binary search over buckets (unless someone thinks a log() call would be faster...)

Definition at line 167 of file Profiler.h.

References buckets, and HistSize.

7.127.3.3 float* Profiler::getBuckets () [inline]

returns the bucket boundaries

Definition at line 138 of file Profiler.h.

References buckets.

7.127.3.4 [SectionInfo*](#) **Profiler::getInfos ()** [`inline`]

gets the actual storage area of the SectionInfo's

Definition at line 153 of file Profiler.h.

References infosOffset.

7.127.3.5 **unsigned int** **Profiler::getNewID (const char * *name*)**

call this to get a new ID number

Definition at line 57 of file Profiler.cc.

References ASSERTRETURN, getInfos(), MaxSectionNameLen, maxSections, Profiler::SectionInfo::name, and sectionsUsed.

7.127.3.6 **void** **Profiler::initBuckets ()** [`static`]

called during process init (before any profiled sections)

Definition at line 51 of file Profiler.cc.

References buckets, HistCurve, HistSize, and HistTime.

7.127.3.7 **std::string** **Profiler::report ()**

outputs profiling information

Definition at line 70 of file Profiler.cc.

References buckets, Profiler::SectionInfo::calls, getInfos(), HistSize, sectionsUsed, startTime, and TimeET::Value().

7.127.3.8 **void** **Profiler::reset ()**

resets profiling information

Definition at line 107 of file Profiler.cc.

References getInfos(), Profiler::SectionInfo::reset(), sectionsUsed, TimeET::Set(), and startTime.

7.127.3.9 **void** **Profiler::setCurrent ([Timer](#) & *tr*)** [`protected`]

called automatically by [Timer\(\)](#) - sets the current timer

Definition at line 119 of file Profiler.cc.

References Profiler::Timer::_id, Profiler::Timer::_parent, Profiler::Timer::_t, TimeET::Age(), Profiler::SectionInfo::calls, curSection, gamma, getBucket(), getInfos(), HistSize, HistTime, Profiler::SectionInfo::interExpAvg, Profiler::SectionInfo::interHist, Profiler::SectionInfo::lastTime, TimeET::Set(), Profiler::SectionInfo::totalInterval, and TimeET::Value().

7.127.4 Friends And Related Function Documentation

7.127.4.1 friend class **Timer** [friend]

Only the Timer's should be calling `setCurrent()` and `finished()` upon the Timer's construction and destruction.

Definition at line 160 of file Profiler.h.

7.127.5 Member Data Documentation

7.127.5.1 float **Profiler::buckets** [static, protected]

holds boundaries for each bucket

Definition at line 6 of file Profiler.cc.

7.127.5.2 unsigned int **Profiler::curSection**

the current timer

Definition at line 146 of file Profiler.h.

7.127.5.3 float **Profiler::gamma**

gamma to use with exponential averages (1 to freeze, 0 to set to last)

Definition at line 148 of file Profiler.h.

7.127.5.4 const float **Profiler::HistCurve** = 4.05 [static]

affects how linearly the buckets are distributed - 1 means linear, >1 causes higher resolution for short times

Definition at line 4 of file Profiler.cc.

7.127.5.5 `const unsigned int Profiler::HistSize = 32` [static]

number of slots in the histograms

Definition at line 85 of file Profiler.h.

7.127.5.6 `const unsigned int Profiler::HistTime = 10*1000` [static]

the upper bound (exclusive) of the histograms, in milliseconds.

Definition at line 87 of file Profiler.h.

7.127.5.7 `unsigned int Profiler::infosOffset = reinterpret_cast<unsigned int>((static_cast<ProfilerOfSize<1>*>(NULL)) → infos)`
[static, protected]

NASTY HACK - this is how we get around using virtual functions.

Definition at line 8 of file Profiler.cc.

7.127.5.8 `const unsigned int Profiler::MaxSectionNameLen = 32` [static]

maximum length of names of timers, 32 chosen to make nice memory alignment

Definition at line 83 of file Profiler.h.

7.127.5.9 `const unsigned int Profiler::maxSections`

so we can read the size of the infos array back again at runtime

Definition at line 149 of file Profiler.h.

7.127.5.10 `unsigned int Profiler::sectionsUsed`

the number of timer IDs which have been assigned

Definition at line 150 of file Profiler.h.

7.127.5.11 `TimeET Profiler::startTime`

time of beginning profiling

Definition at line 147 of file Profiler.h.

The documentation for this class was generated from the following files:

- [Profiler.h](#)
- [Profiler.cc](#)

7.128 Profiler::SectionInfo Struct Reference

```
#include <Profiler.h>
```

7.128.1 Detailed Description

holds all the information needed for book keeping for each timer

Definition at line 93 of file Profiler.h.

Public Member Functions

- [SectionInfo](#) ()
constructor
- void [reset](#) ()
resets profiling information

Public Attributes

- char [name](#) [[MaxSectionNameLen](#)]
the name of this timer
- [TimeET](#) [totalTime](#)
the total time spent in this section
- [TimeET](#) [lastTime](#)
time of last call, used to calculate [totalInterval](#), which gives idea of rate of calls
- [TimeET](#) [totalInterval](#)
the total time spent between calls (not time between end of one and start of next, is time between start of one and start of next)
- [TimeET](#) [childTime](#)
the total time spent in child sections
- float [execExpAvg](#)
exponential average of execution time
- float [interExpAvg](#)
exponential average of inter-call time

- unsigned int [execHist](#) [[HistSize](#)]
histogram of execution times, uses logarithmic size bins (so high res for quick functions, low res for longer functions)
- unsigned int [interHist](#) [[HistSize](#)]
histogram of inter-call time, uses logarithmic size bins (so high res for quick functions, low res for longer functions)
- unsigned int [calls](#)
number of calls to this section

7.128.2 Constructor & Destructor Documentation

7.128.2.1 Profiler::SectionInfo::SectionInfo ()

constructor

Definition at line 31 of file Profiler.cc.

References [execHist](#), [interHist](#), and [name](#).

7.128.3 Member Function Documentation

7.128.3.1 void Profiler::SectionInfo::reset ()

resets profiling information

Definition at line 39 of file Profiler.cc.

References [calls](#), [childTime](#), [execExpAvg](#), [execHist](#), [interExpAvg](#), [interHist](#), [lastTime](#), [TimeET::Set\(\)](#), [totalInterval](#), and [totalTime](#).

7.128.4 Member Data Documentation

7.128.4.1 unsigned int [Profiler::SectionInfo::calls](#)

number of calls to this section

Definition at line 105 of file Profiler.h.

7.128.4.2 [TimeET Profiler::SectionInfo::childTime](#)

the total time spent in child sections

Definition at line 100 of file Profiler.h.

7.128.4.3 float [Profiler::SectionInfo::execExpAvg](#)

exponential average of execution time

Definition at line 101 of file Profiler.h.

7.128.4.4 unsigned int [Profiler::SectionInfo::execHist](#)[HistSize]

histogram of execution times, uses logarithmic size bins (so high res for quick functions, low res for longer functions)

Definition at line 103 of file Profiler.h.

7.128.4.5 float [Profiler::SectionInfo::interExpAvg](#)

exponential average of inter-call time

Definition at line 102 of file Profiler.h.

7.128.4.6 unsigned int [Profiler::SectionInfo::interHist](#)[HistSize]

histogram of inter-call time, uses logarithmic size bins (so high res for quick functions, low res for longer functions)

Definition at line 104 of file Profiler.h.

7.128.4.7 TimeET [Profiler::SectionInfo::lastTime](#)

time of last call, used to calculate [totalInterval](#), which gives idea of rate of calls

Definition at line 98 of file Profiler.h.

7.128.4.8 char [Profiler::SectionInfo::name](#)[MaxSectionNameLen]

the name of this timer

Definition at line 96 of file Profiler.h.

7.128.4.9 TimeET [Profiler::SectionInfo::totalInterval](#)

the total time spent between calls (not time between end of one and start of next, is time between start of one and start of next)

Definition at line 99 of file Profiler.h.

7.128.4.10 [TimeET Profiler::SectionInfo::totalTime](#)

the total time spent in this section

Definition at line 97 of file Profiler.h.

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

- [Profiler.h](#)
- [Profiler.cc](#)

7.129 Profiler::Timer Class Reference

```
#include <Profiler.h>
```

7.129.1 Detailed Description

Measures the time that this class exists, reports result to a profiler.

Don't bother trying to use this as a quick timer - just use [TimeET](#) directly. But there are functions to get the elapsed time and such if you insist.

Definition at line 111 of file Profiler.h.

Public Member Functions

- [Timer](#) ()
constructor - starts timer, but you can restart it...
- [Timer](#) (unsigned int id, [Profiler](#) *prof)
constructor - starts the timer, sets current timer in prof
- [Timer](#) (const [Timer](#) &t)
copy constructor, not that you should need it, does same as default
- [Timer](#) operator= (const [Timer](#) &t)
not that you should need it, does same as default
- [~Timer](#) ()
destructor - stops the timer, reports results
- void [setID](#) (unsigned int id, [Profiler](#) *prof)
sets the ID and profiler; also starts timer
- void [start](#) ()
starts timer (or resets it)
- const [TimeET](#) & [startTime](#) ()
returns time of start
- [TimeET](#) [elapsed](#) ()
returns time since start

Protected Attributes

- [Profiler](#) * [_prof](#)
the profiler this should report to
- unsigned int [_id](#)
the id number for this code section (See example in beginning of class documentation for how these are assigned)
- unsigned int [_parent](#)
the id number of the timer this timer is under
- [TimeET](#) [_t](#)
the time this timer was created

Friends

- class [Profiler](#)
[Profiler](#) will need to read out some data that no one else should be depending on.

7.129.2 Constructor & Destructor Documentation

7.129.2.1 [Profiler::Timer::Timer](#) () [inline]

constructor - starts timer, but you can restart it...

Definition at line 115 of file [Profiler.h](#).

References [_id](#), [_parent](#), [_prof](#), and [_t](#).

7.129.2.2 [Profiler::Timer::Timer](#) (unsigned int *id*, [Profiler](#) * *prof*)

constructor - starts the timer, sets current timer in *prof*

Tells the profiler that this is now the active timer, so new timers will fit "under" this.

[Timer](#) isn't actually started here, lets [Profiler::setCurrent](#) do that.

Parameters:

prof profiler to report results to. If is NULL, does nothing.

id id number for this function. See [Profiler::getNewID\(\)](#) for what you should pass this

Definition at line 14 of file Profiler.cc.

References `_prof`, and `Profiler::setCurrent()`.

7.129.2.3 `Profiler::Timer::Timer (const Timer & t)` `[inline]`

copy constructor, not that you should need it, does same as default

Definition at line 117 of file Profiler.h.

References `_id`, `_parent`, `_prof`, and `_t`.

7.129.2.4 `Profiler::Timer::~~Timer ()`

destructor - stops the timer, reports results

Definition at line 19 of file Profiler.cc.

References `_prof`, and `Profiler::finished()`.

7.129.3 Member Function Documentation

7.129.3.1 `TimeET Profiler::Timer::elapsed ()` `[inline]`

returns time since start

Definition at line 123 of file Profiler.h.

References `_t`, and `TimeET::Age()`.

7.129.3.2 `Timer Profiler::Timer::operator= (const Timer & t)` `[inline]`

not that you should need it, does same as default

Definition at line 118 of file Profiler.h.

References `_id`, `_parent`, `_prof`, and `_t`.

7.129.3.3 `void Profiler::Timer::setID (unsigned int id, Profiler * prof)`

sets the ID and profiler, also starts timer

Definition at line 24 of file Profiler.cc.

References `_id`, `_prof`, and `Profiler::setCurrent()`.

7.129.3.4 void Profiler::Timer::start () [inline]

starts timer (or resets it)

Definition at line 121 of file Profiler.h.

References `_t`, and `TimeET::Set()`.

7.129.3.5 const TimeET& Profiler::Timer::startTime () [inline]

returns time of start

Definition at line 122 of file Profiler.h.

References `_t`.

7.129.4 Friends And Related Function Documentation

7.129.4.1 friend class Profiler [friend]

`Profiler` will need to read out some data that no one else should be depending on.

Definition at line 113 of file Profiler.h.

7.129.5 Member Data Documentation

7.129.5.1 unsigned int Profiler::Timer::id [protected]

the id number for this code section (See example in beginning of class documentation for how these are assigned)

Definition at line 126 of file Profiler.h.

7.129.5.2 unsigned int Profiler::Timer::_parent [protected]

the id number of the timer this timer is under

Definition at line 127 of file Profiler.h.

7.129.5.3 Profiler* Profiler::Timer::_prof [protected]

the profiler this should report to

Definition at line 125 of file Profiler.h.

7.129.5.4 [TimeET Profiler::Timer::t](#) [protected]

the time this timer was created

Definition at line 128 of file Profiler.h.

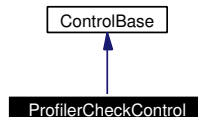
The documentation for this class was generated from the following files:

- [Profiler.h](#)
- [Profiler.cc](#)

7.130 ProfilerCheckControl Class Reference

```
#include <ProfilerCheckControl.h>
```

Inheritance diagram for ProfilerCheckControl:



7.130.1 Detailed Description

causes the [WorldState::mainProfile](#) and [WorldState::motionProfile](#) to display reports to cout

Definition at line 9 of file ProfilerCheckControl.h.

Public Member Functions

- [ProfilerCheckControl \(\)](#)
Constructor.
- [~ProfilerCheckControl \(\)](#)
Destructor.
- virtual [ControlBase *](#) [activate](#) ([MotionManager::MC_ID](#) display, [Socket *](#))
Prints a report to cout.

7.130.2 Constructor & Destructor Documentation

7.130.2.1 ProfilerCheckControl::ProfilerCheckControl () [inline]

Constructor.

Definition at line 12 of file ProfilerCheckControl.h.

References time.

7.130.2.2 `ProfilerCheckControl::~~ProfilerCheckControl()` [inline]

Destructor.

Definition at line 15 of file `ProfilerCheckControl.h`.

7.130.3 Member Function Documentation

7.130.3.1 `virtual ControlBase* ProfilerCheckControl::activate(MotionManager::MC_ID display, Socket *)` [inline, virtual]

Prints a report to cout.

Todo

- make the leds flash

Reimplemented from `ControlBase`.

Definition at line 18 of file `ProfilerCheckControl.h`.

References `MotionManager::invalid_MC_ID`, `WorldState::mainProfile`, `WorldState::motionProfile`, `Profiler::report()`, and `state`.

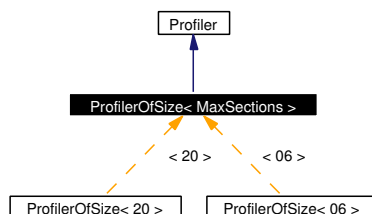
The documentation for this class was generated from the following file:

- `ProfilerCheckControl.h`

7.131 ProfilerOfSize< MaxSections > Class Template Reference

```
#include <Profiler.h>
```

Inheritance diagram for ProfilerOfSize< MaxSections >:



7.131.1 Detailed Description

```
template<unsigned int MaxSections> class ProfilerOfSize< MaxSections >
```

templated subclass allows compile-time flexibility of how much memory to use.

Definition at line 189 of file Profiler.h.

Public Member Functions

- [ProfilerOfSize](#) ()
constructor

Public Attributes

- SectionInfo [infos](#) [MaxSections]
the actual profiling information storage

7.131.2 Constructor & Destructor Documentation

```
7.131.2.1 template<unsigned int MaxSections> ProfilerOfSize< MaxSections  
>::ProfilerOfSize () [inline]
```

constructor

Definition at line 191 of file Profiler.h.

7.131.3 Member Data Documentation

7.131.3.1 `template<unsigned int MaxSections> SectionInfo ProfilerOfSize<MaxSections >::infos[MaxSections]`

the actual profiling information storage

Definition at line 192 of file Profiler.h.

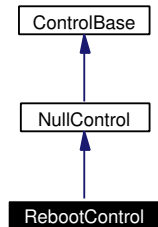
The documentation for this class was generated from the following file:

- [Profiler.h](#)

7.132 RebootControl Class Reference

```
#include <RebootControl.h>
```

Inheritance diagram for RebootControl:



7.132.1 Detailed Description

when activated, this will cause the aibo to reboot

Definition at line 8 of file RebootControl.h.

Public Member Functions

- [RebootControl](#) ()
constructor
- [RebootControl](#) (const std::string &n)
constructor
- [RebootControl](#) (const std::string &n, const std::string &d)
constructor
- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#), [Socket](#) *)
calls [doSelect\(\)](#)
- virtual [ControlBase](#) * [doSelect](#) ()
reboots

7.132.2 Constructor & Destructor Documentation

7.132.2.1 `RebootControl::RebootControl ()` [inline]

constructor

Definition at line 11 of file `RebootControl.h`.

7.132.2.2 `RebootControl::RebootControl (const std::string & n)` [inline]

constructor

Definition at line 12 of file `RebootControl.h`.

7.132.2.3 `RebootControl::RebootControl (const std::string & n, const std::string & d)` [inline]

constructor

Definition at line 13 of file `RebootControl.h`.

7.132.3 Member Function Documentation

7.132.3.1 `virtual ControlBase* RebootControl::activate (MotionManager::MC_ID, Socket *)` [inline, virtual]

calls `doSelect()`

Reimplemented from `NullControl`.

Definition at line 15 of file `RebootControl.h`.

References `doSelect()`.

7.132.3.2 `ControlBase* RebootControl::doSelect ()` [virtual]

reboots

Reimplemented from `NullControl`.

Definition at line 4 of file `RebootControl.cc`.

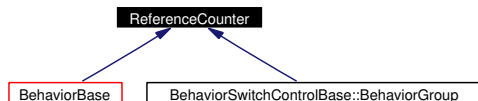
The documentation for this class was generated from the following files:

- [RebootControl.h](#)
- [RebootControl.cc](#)

7.133 ReferenceCounter Class Reference

```
#include <ReferenceCounter.h>
```

Inheritance diagram for ReferenceCounter:



7.133.1 Detailed Description

Performs simple reference counting, will delete the object when removing the last reference.

Definition at line 11 of file ReferenceCounter.h.

Public Member Functions

- [ReferenceCounter](#) ()
constructor
- virtual [~ReferenceCounter](#) ()
destructor - will std::cout a warning if still has references
- virtual void [AddReference](#) ()
adds one to references
- virtual void [RemoveReference](#) ()
subtracts one from references AND DELETES the object IF ZERO
- virtual unsigned int [GetReferences](#) ()
returns the number of references
- void [SetAutoDelete](#) (bool b)
if true, next time a [RemoveReference\(\)](#) causes references to hit 0, the object will delete itself
- bool [GetAutoDelete](#) ()
returns RC_autodelete

Protected Attributes

- unsigned int [references](#)
the current number of references
- bool [RC_autodelete](#)
prevents deletion when counter hits 0

7.133.2 Constructor & Destructor Documentation

7.133.2.1 `ReferenceCounter::ReferenceCounter ()` [`inline`]

constructor

Definition at line 14 of file `ReferenceCounter.h`.

References `RC_autodelete`, and `references`.

7.133.2.2 `virtual ReferenceCounter::~~ReferenceCounter ()` [`inline`, `virtual`]

destructor - will `std::cout` a warning if still has references

Definition at line 17 of file `ReferenceCounter.h`.

References `references`.

7.133.3 Member Function Documentation

7.133.3.1 `virtual void ReferenceCounter::AddReference ()` [`inline`, `virtual`]

adds one to `references`

Definition at line 23 of file `ReferenceCounter.h`.

References `references`.

7.133.3.2 `bool ReferenceCounter::GetAutoDelete ()` [`inline`]

returns `RC_autodelete`

Definition at line 39 of file `ReferenceCounter.h`.

References `RC_autodelete`.

7.133.3.3 **virtual unsigned int ReferenceCounter::GetReferences ()** [inline, virtual]

returns the number of references

Returns:

references

Definition at line 34 of file ReferenceCounter.h.

References references.

7.133.3.4 **virtual void ReferenceCounter::RemoveReference ()** [inline, virtual]

subtracts one from references AND DELETES the object IF ZERO

Definition at line 25 of file ReferenceCounter.h.

References RC_autodelete, and references.

7.133.3.5 **void ReferenceCounter::SetAutoDelete (bool *b*)** [inline]

if true, next time a [RemoveReference\(\)](#) causes references to hit 0, the object will delete itself

Definition at line 37 of file ReferenceCounter.h.

References RC_autodelete.

7.133.4 Member Data Documentation

7.133.4.1 **bool [ReferenceCounter::RC_autodelete](#)** [protected]

prevents deletion when counter hits 0

Definition at line 46 of file ReferenceCounter.h.

7.133.4.2 **unsigned int [ReferenceCounter::references](#)** [protected]

the current number of references

Definition at line 43 of file ReferenceCounter.h.

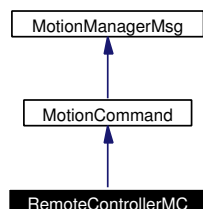
The documentation for this class was generated from the following file:

- [ReferenceCounter.h](#)

7.134 RemoteControllerMC Class Reference

```
#include <RemoteControllerMC.h>
```

Inheritance diagram for RemoteControllerMC:



7.134.1 Detailed Description

This class is used for setting all PIDJoints to a certain set of values (not the gains, just the joint positions).

Definition at line 10 of file RemoteControllerMC.h.

Public Member Functions

- [RemoteControllerMC](#) ()
constructor, defaults to active, all joints at 0
- virtual [~RemoteControllerMC](#) ()
destructor
- void [setDirty](#) ()
sets dirty flag to true

Inherited:

Updates all PIDJoint values

- virtual int [updateOutputs](#) ()
is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)
- virtual int [isDirty](#) ()
true if a change has been made since the last `updateJointCmds()` and we're active

- virtual int [isAlive](#) ()
always true

Public Attributes

- float [cmds](#) [NumPIDJoints]
current vector of positions

Protected Attributes

- bool [dirty](#)
true if a change has been made since last call to updateJointCmds()
- bool [active](#)
set by accessor functions, defaults to true

7.134.2 Constructor & Destructor Documentation

7.134.2.1 RemoteControllerMC::RemoteControllerMC () [inline]

constructor, defaults to active, all joints at 0

Definition at line 13 of file RemoteControllerMC.h.

References [active](#), [cmds](#), [dirty](#), and [ERS210Info::NumPIDJoints](#).

7.134.2.2 virtual RemoteControllerMC::~RemoteControllerMC () [inline, virtual]

destructor

Definition at line 17 of file RemoteControllerMC.h.

7.134.3 Member Function Documentation

7.134.3.1 virtual int RemoteControllerMC::isAlive () [inline, virtual]

always true

Implements [MotionCommand](#).

Definition at line 30 of file RemoteControllerMC.h.

7.134.3.2 `virtual int RemoteControllerMC::isDirty () [inline, virtual]`

true if a change has been made since the last updateJointCmds() and we're active

Implements [MotionCommand](#).

Definition at line 29 of file RemoteControllerMC.h.

References active, and dirty.

7.134.3.3 `void RemoteControllerMC::setDirty () [inline]`

sets dirty flag to true

Definition at line 33 of file RemoteControllerMC.h.

References dirty.

7.134.3.4 `virtual int RemoteControllerMC::updateOutputs () [inline, virtual]`

is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)

Returns:

zero if no changes were made, non-zero otherwise

See also:

[RobotInfo::NumFrames](#)

[RobotInfo::FrameTime](#)

Implements [MotionCommand](#).

Definition at line 21 of file RemoteControllerMC.h.

References [cmds](#), [dirty](#), [isDirty\(\)](#), [motman](#), [ERS210Info::NumPIDJoints](#), [ERS210Info::PIDJointOffset](#), and [MotionManager::setOutput\(\)](#).

7.134.4 Member Data Documentation

7.134.4.1 `bool RemoteControllerMC::active [protected]`

set by accessor functions, defaults to true

Definition at line 38 of file RemoteControllerMC.h.

7.134.4.2 float [RemoteControllerMC::cmds](#)[NumPIDJoints]

current vector of positions

Definition at line 34 of file RemoteControllerMC.h.

7.134.4.3 bool [RemoteControllerMC::dirty](#) [protected]

true if a change has been made since last call to updateJointCmds()

Definition at line 37 of file RemoteControllerMC.h.

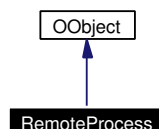
The documentation for this class was generated from the following file:

- [RemoteControllerMC.h](#)

7.135 RemoteProcess Class Reference

```
#include <RemoteProcess.h>
```

Inheritance diagram for RemoteProcess:



7.135.1 Detailed Description

Sample RemoteProcessingOPENR process.

RemoteProcess runs on Linux with OPEN_R_SDK installed (and the Remote-ProcessingOPENR patch). RemoteProcess talks to the Aibo using OPENR messages. RemoteProcessingOPENR can receive all the OPENR system sensor streams and write to all output streams available on the Aibo.

In short, use this if you're falling short of processing power on the Aibo, and the data you want processed is small in size. It'll help greatly if you're comfortable with OPENR processes and message passing.

Please read the RemoteProcessingOPENR instructions on the main Tekkotsu page for more information.

Definition at line 24 of file RemoteProcess.h.

Public Member Functions

- [RemoteProcess](#) ()
constructor
- [~RemoteProcess](#) ()
destructor
- void [start](#) ()
called when objects are connected (by DoStart). Add user code here.
- void [data_received](#) (const char *buf)
called when a message is received (by ROPENR_notify). Add user code here.

- virtual OStatus [DoInit](#) (const OSystemEvent &event)
first call (after constructor), set up memory
- virtual OStatus [DoStart](#) (const OSystemEvent &event)
second call, ask for messages
- virtual OStatus [DoStop](#) (const OSystemEvent &event)
next to last call, stop sending and receiving messages
- virtual OStatus [DoDestroy](#) (const OSystemEvent &event)
last call (before destructor), clean up memory here
- bool [RPOPENR_isReady](#) ()
indicates whether the Aibo is ready to receive more messages
- int [RPOPENR_send](#) (char *buf, int bufsize)
send message to Aibo
- void [RPOPENR_ready](#) (const OReadyEvent &event)
OPENR callback for registering when the Aibo is ready for messages.
- void [RPOPENR_notify](#) (const ONotifyEvent &event)
OPENR callback for when a message is received from the Aibo.

Public Attributes

- OSubject * [subject](#) [numOfSubject]
holds information for each of our subjects (data we provide)
- OObserver * [observer](#) [numOfObserver]
holds information for each of the sources we're observing
- bool [RPOPENR_isready](#)
set to true after [RPOPENR_ready\(\)](#) was called

7.135.2 Constructor & Destructor Documentation

7.135.2.1 RemoteProcess::RemoteProcess ()

constructor

Definition at line 5 of file RemoteProcess.cc.

7.135.2.2 RemoteProcess::~~RemoteProcess ()

destructor

Definition at line 10 of file RemoteProcess.cc.

7.135.3 Member Function Documentation

7.135.3.1 void RemoteProcess::data_received (const char * *buf*)

called when a message is received (by RPOPENR_notify). Add user code here.

Definition at line 28 of file RemoteProcess.cc.

7.135.3.2 OStatus RemoteProcess::DoDestroy (const OSystemEvent & *event*) [virtual]

last call (before destructor), clean up memory here

Definition at line 63 of file RemoteProcess.cc.

7.135.3.3 OStatus RemoteProcess::DoInit (const OSystemEvent & *event*) [virtual]

first call (after constructor), set up memory

Definition at line 37 of file RemoteProcess.cc.

7.135.3.4 OStatus RemoteProcess::DoStart (const OSystemEvent & *event*) [virtual]

second call, ask for messages

Definition at line 46 of file RemoteProcess.cc.

References start().

7.135.3.5 OStatus RemoteProcess::DoStop (const OSystemEvent & *event*) [virtual]

next to last call, stop sending and receiving messages

Definition at line 55 of file RemoteProcess.cc.

7.135.3.6 bool RemoteProcess::RPOPENR_isReady () [inline]

indicates whether the Aibo is ready to receive more messages

Definition at line 45 of file RemoteProcess.h.

References RPOPENR_isready.

7.135.3.7 void RemoteProcess::RPOPENR_notify (const ONotifyEvent & event)

OPENR callback for when a message is received from the Aibo.

Definition at line 70 of file RemoteProcess.cc.

References data_received(), and observer.

**7.135.3.8 void RemoteProcess::RPOPENR_ready (const OReadyEvent & event)
[inline]**

OPENR callback for registering when the Aibo is ready for messages.

Definition at line 48 of file RemoteProcess.h.

References RPOPENR_isready.

7.135.3.9 int RemoteProcess::RPOPENR_send (char * buf, int bufsize)

send message to Aibo

Definition at line 77 of file RemoteProcess.cc.

References RPOPENR_isready, and subject.

7.135.3.10 void RemoteProcess::start ()

called when objects are connected (by DoStart). Add user code here.

Definition at line 16 of file RemoteProcess.cc.

References RPOPENR_isReady(), and RPOPENR_send().

7.135.4 Member Data Documentation**7.135.4.1 OObserver* RemoteProcess::observer[numOfObserver]**

holds information for each of the sources we're observing

Definition at line 37 of file RemoteProcess.h.

7.135.4.2 bool [RemoteProcess::RPOPENR_isready](#)

set to true after [RPOPENR_ready\(\)](#) was called

Definition at line 44 of file RemoteProcess.h.

7.135.4.3 OSubject* [RemoteProcess::subject](#)[numOfSubject]

holds information for each of our subjects (data we provide)

Definition at line 36 of file RemoteProcess.h.

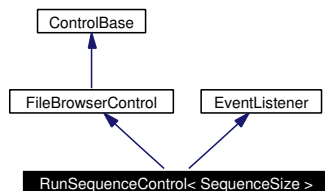
The documentation for this class was generated from the following files:

- [RemoteProcess.h](#)
- [RemoteProcess.cc](#)

7.136 RunSequenceControl< SequenceSize > Class Template Reference

```
#include <RunSequenceControl.h>
```

Inheritance diagram for RunSequenceControl< SequenceSize >:



7.136.1 Detailed Description

template<unsigned int SequenceSize> class RunSequenceControl< SequenceSize >

Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...).

The template parameter is passed to MotionSequenceMC's template parameter in order to specify the number of keyframes to reserve - larger values use more memory, but will allow you to load more complicated sequences.

Definition at line 17 of file RunSequenceControl.h.

Public Member Functions

- **RunSequenceControl** (const std::string &n, [MotionManager::MC_ID](#) estop_id)
*Constructor, sets filter to *.mot.*
- virtual void **processEvent** (const [EventBase](#) &event)
this is to help reduce the twitch at the end (estop tries to go back to its position when this is removed)

Protected Member Functions

- virtual [ControlBase](#) * **selectedFile** (const std::string &f)
does the actual loading of the [MotionSequence](#)

Protected Attributes

- [MotionManager::MC_ID](#) estopid

MC_ID of the emergency stop (so we can reset it to the end frame).

7.136.2 Constructor & Destructor Documentation

7.136.2.1 `template<unsigned int SequenceSize> RunSequenceControl< SequenceSize >::RunSequenceControl (const std::string & n, MotionManager::MC_ID estop_id) [inline]`

Constructor, sets filter to *.mot.

Definition at line 20 of file [RunSequenceControl.h](#).

References [config](#), [RunSequenceControl](#)< SequenceSize >::estopid, [FileBrowserControl::root](#), and [FileBrowserControl::setFilter\(\)](#).

7.136.3 Member Function Documentation

7.136.3.1 `template<unsigned int SequenceSize> virtual void RunSequenceControl< SequenceSize >::processEvent (const EventBase & event) [inline, virtual]`

this is to help reduce the twitch at the end (estop tries to go back to its position when this is removed)

Implements [EventListener](#).

Definition at line 27 of file [RunSequenceControl.h](#).

References [erouter](#), [RunSequenceControl](#)< SequenceSize >::estopid, and [EventRouter::removeListener\(\)](#).

7.136.3.2 `template<unsigned int SequenceSize> virtual ControlBase* RunSequenceControl< SequenceSize >::selectedFile (const std::string & f) [inline, protected, virtual]`

does the actual loading of the [MotionSequence](#)

Reimplemented from [FileBrowserControl](#).

Definition at line 36 of file [RunSequenceControl.h](#).

References EventRouter::addListener(), MotionManager::addMotion(), TimeET::Age(), EventBase::deactivateETID, erouter, RunSequenceControl< SequenceSize >::estopid, MotionManager::kEmergencyPriority, MotionManager::MC_ID, motman, and EventBase::motmanEGID.

7.136.4 Member Data Documentation

7.136.4.1 `template<unsigned int SequenceSize> MotionManager::MC_ID RunSequenceControl< SequenceSize >::estopid` [protected]

MC_ID of the emergency stop (so we can reset it to the end frame).

Definition at line 46 of file RunSequenceControl.h.

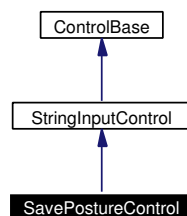
The documentation for this class was generated from the following file:

- [RunSequenceControl.h](#)

7.137 SavePostureControl Class Reference

```
#include <SavePostureControl.h>
```

Inheritance diagram for SavePostureControl:



7.137.1 Detailed Description

Upon activation, saves the current position to a file name read from user (stored in /ms/data/motion/...).

Definition at line 9 of file SavePostureControl.h.

Public Member Functions

- [SavePostureControl](#) ()
Constructor.
- [SavePostureControl](#) (const std::string &n)
Constructor.
- virtual [ControlBase](#) * [takeInput](#) (const std::string &msg)
called when the user has supplied a text string (may not have been prompted by [do-ReadStdIn\(!\)](#))

7.137.2 Constructor & Destructor Documentation

7.137.2.1 SavePostureControl::SavePostureControl () [inline]

Constructor.

Definition at line 12 of file SavePostureControl.h.

References [ControlBase::name](#).

7.137.2.2 SavePostureControl::SavePostureControl (const std::string & n) [inline]

Constructor.

Definition at line 14 of file SavePostureControl.h.

References [ControlBase::name](#).

7.137.3 Member Function Documentation

7.137.3.1 virtual [ControlBase](#)* SavePostureControl::takeInput (const std::string & msg) [inline, virtual]

called when the user has supplied a text string (may not have been prompted by [do-ReadStdIn\(\)](#)!)

Reimplemented from [StringInputControl](#).

Definition at line 16 of file SavePostureControl.h.

References [LoadSave::SaveFile\(\)](#), [StringInputControl::takeInput\(\)](#), and [Posture-Engine::takeSnapshot\(\)](#).

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

- [SavePostureControl.h](#)

7.138 SaveWalkControl Class Reference

```
#include <SaveWalkControl.h>
```

Inheritance diagram for SaveWalkControl:



7.138.1 Detailed Description

When activated, saves walk parameters to a file specified from cin.

Definition at line 10 of file SaveWalkControl.h.

Public Member Functions

- **SaveWalkControl** (const std::string &n, **MotionManager::MC_ID** w)
constructor, pass the MC_ID of the walk you want to save
- virtual **~SaveWalkControl** ()
destructor
- virtual **ControlBase * activate** (**MotionManager::MC_ID** disp_id, **Socket ***)
when called, prompts on cin for file to save into – will overwrite this file
- virtual void **deactivate** ()
does nothing

Protected Attributes

- **MotionManager::MC_ID** walk_id
MC_ID of walk to save from.

7.138.2 Constructor & Destructor Documentation

7.138.2.1 SaveWalkControl::SaveWalkControl (const std::string & *n*, [MotionManager::MC_ID](#) *w*) [inline]

constructor, pass the MC_ID of the walk you want to save

Definition at line 13 of file SaveWalkControl.h.

References [walk_id](#).

7.138.2.2 virtual SaveWalkControl::~~SaveWalkControl () [inline, virtual]

destructor

Definition at line 15 of file SaveWalkControl.h.

7.138.3 Member Function Documentation

7.138.3.1 virtual [ControlBase](#)* SaveWalkControl::activate ([MotionManager::MC_ID](#) *disp_id*, [Socket](#) *) [inline, virtual]

when called, prompts on cin for file to save into – will overwrite this file

Reimplemented from [ControlBase](#).

Definition at line 18 of file SaveWalkControl.h.

References [MotionManager::checkinMotion\(\)](#), [MotionManager::checkoutMotion\(\)](#), [ERS210Info::FaceLEDMask](#), [MotionManager::invalid_MC_ID](#), [MMAccessor< MC.t >::mc\(\)](#), [motman](#), [WalkMC::save\(\)](#), and [walk_id](#).

7.138.3.2 virtual void SaveWalkControl::deactivate () [inline, virtual]

does nothing

Reimplemented from [ControlBase](#).

Definition at line 36 of file SaveWalkControl.h.

7.138.4 Member Data Documentation

7.138.4.1 [MotionManager::MC_ID](#) SaveWalkControl::walk_id [protected]

MC_ID of walk to save from.

Definition at line 39 of file SaveWalkControl.h.

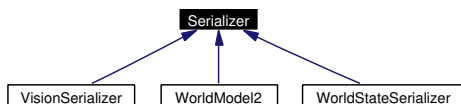
The documentation for this class was generated from the following file:

- [SaveWalkControl.h](#)

7.139 Serializer Class Reference

```
#include <Serializer.h>
```

Inheritance diagram for Serializer:



7.139.1 Detailed Description

provides a default serializer base class for simple objects

Definition at line 5 of file `Serializer.h`.

Static Protected Member Functions

- `template<class T> void encode (char **dst, T value)`
writes value to dst and advances dst
- `void hostToNetwork (char *dst, char *src, int length)`
converts to network byte order (big endian - aibo is little endian)
- `void encode (char **dst, char *src, int length)`
writes length bytes from src to dst
- `void encodeDoublesAsFloats (char **dst, double *src, int length)`
a simple form of compression - calls `encode(dst, float(src[i]))` for $i=0..length$

7.139.2 Member Function Documentation

7.139.2.1 `void Serializer::encode (char ** dst, char * src, int length)`
[inline, static, protected]

writes *length* bytes from *src* to *dst*

Definition at line 26 of file `Serializer.h`.

7.139.2.2 `template<class T> void Serializer::encode (char ** dst, T value)`
[inline, static, protected]

writes *value* to *dst* and advances *dst*

Definition at line 11 of file Serializer.h.

7.139.2.3 `void Serializer::encodeDoublesAsFloats (char ** dst, double * src, int length)` [inline, static, protected]

a simple form of compression - calls `encode(dst, float(src[i]))` for `i=0..length`

Definition at line 32 of file Serializer.h.

References `encode()`.

7.139.2.4 `void Serializer::hostToNetwork (char * dst, char * src, int length)`
[inline, static, protected]

converts to network byte order (big endian - aibo is little endian)

Definition at line 20 of file Serializer.h.

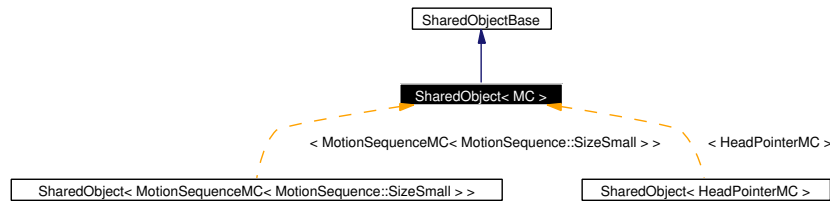
The documentation for this class was generated from the following file:

- [Serializer.h](#)

7.140 SharedObject< MC > Class Template Reference

```
#include <SharedObject.h>
```

Inheritance diagram for SharedObject< MC >:



7.140.1 Detailed Description

```
template<class MC> class SharedObject< MC >
```

This templated class allows convenient creation of any type of class wrapped in a shared memory region.

See also:

[MotionManager](#) for an example on how to use this.

Definition at line 26 of file `SharedObject.h`.

Public Member Functions

- `MC * operator → () const`
smart pointer to the underlying class
- `MC & operator * () const`
smart pointer to the underlying class
- `MC & operator[] (int i) const`
smart pointer to the underlying class

templated constructors - allows you to pass constructor arguments on to the object being created

if you really need more than 5 arguments for your class, well, you're one crazy puppy but if you really want to, just make more like shown... (yay templates!)

- [SharedObject](#) ()
Creates the class with the default constructor.
- `template<class T1> SharedObject (T1 t1)`
Creates the class, passing its constructor t1.
- `template<class T1, class T2> SharedObject (T1 t1, T2 t2)`
Creates the class, passing its constructor t1 and t2.
- `template<class T1, class T2, class T3> SharedObject (T1 t1, T2 t2, T3 t3)`
Creates the class, passing its constructor t1, t2, and t3.
- `template<class T1, class T2, class T3, class T4> SharedObject (T1 t1, T2 t2, T3 t3, T4 t4)`
Creates the class, passing its constructor t1, t2, t3 and t4.
- `template<class T1, class T2, class T3, class T4, class T5> SharedObject (T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)`
Creates the class, passing its constructor t1, t2, t3, t4 and t5 - if you need more arguments, just add them.

Protected Member Functions

- `MC * dataCasted () const`
returns a correctly typed pointer to the object's memory
- `unsigned int calcsiz ()`
Calculates the size of the memory region to be used, rounding up to the nearest page size.

7.140.2 Constructor & Destructor Documentation

7.140.2.1 `template<class MC> SharedObject< MC >::SharedObject ()` [inline]

Creates the class with the default constructor.

Definition at line 32 of file SharedObject.h.

7.140.2.2 `template<class MC> template<class T1> SharedObject< MC >::SharedObject (T1 t1)` [inline]

Creates the class, passing its constructor t1.

Definition at line 37 of file SharedObject.h.

7.140.2.3 `template<class MC> template<class T1, class T2> SharedObject< MC >::SharedObject (T1 t1, T2 t2) [inline]`

Creates the class, passing its constructor t1 and t2.

Definition at line 42 of file SharedObject.h.

7.140.2.4 `template<class MC> template<class T1, class T2, class T3> SharedObject< MC >::SharedObject (T1 t1, T2 t2, T3 t3) [inline]`

Creates the class, passing its constructor t1, t2, and t3.

Definition at line 47 of file SharedObject.h.

7.140.2.5 `template<class MC> template<class T1, class T2, class T3, class T4> SharedObject< MC >::SharedObject (T1 t1, T2 t2, T3 t3, T4 t4) [inline]`

Creates the class, passing its constructor t1, t2, t3 and t4.

Definition at line 52 of file SharedObject.h.

7.140.2.6 `template<class MC> template<class T1, class T2, class T3, class T4, class T5> SharedObject< MC >::SharedObject (T1 t1, T2 t2, T3 t3, T4 t4, T5 t5) [inline]`

Creates the class, passing its constructor t1, t2, t3, t4 and t5 - if you need more arguments, just add them.

Definition at line 57 of file SharedObject.h.

7.140.3 Member Function Documentation

7.140.3.1 `template<class MC> unsigned int SharedObject< MC >::calcsize () [inline, protected]`

Calculates the size of the memory region to be used, rounding up to the nearest page size.

Not sure this is completely necessary, but may be nice. Of course, this also means even small regions are going to be at least 4K (current page size) If memory gets tight or we get a lot of little regions floating around, this might be worth checking into

Definition at line 73 of file SharedObject.h.

7.140.3.2 `template<class MC> MC* SharedObject< MC >::dataCasted ()
const [inline, protected]`

returns a correctly typed pointer to the object's memory

Definition at line 67 of file SharedObject.h.

7.140.3.3 `template<class MC> MC& SharedObject< MC >::operator * ()
const [inline]`

smart pointer to the underlying class

Definition at line 64 of file SharedObject.h.

7.140.3.4 `template<class MC> MC* SharedObject< MC >::operator → ()
const [inline]`

smart pointer to the underlying class

Definition at line 63 of file SharedObject.h.

7.140.3.5 `]`

`template<class MC> MC& SharedObject< MC >::operator[] (int i) const
[inline]`

smart pointer to the underlying class

Definition at line 65 of file SharedObject.h.

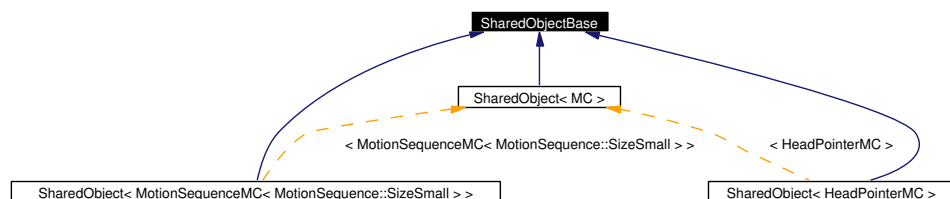
The documentation for this class was generated from the following file:

- [SharedObject.h](#)

7.141 SharedObjectBase Class Reference

```
#include <SharedObject.h>
```

Inheritance diagram for SharedObjectBase:



7.141.1 Detailed Description

It's nice to have a parent class of [SharedObject](#) (which is what you probably want to be reading) so that you can pass around the data structure without worrying about what type is inside the shared memory region.

See [MotionManager](#) for an example on how to use this.

Definition at line 9 of file [SharedObject.h](#).

Public Member Functions

- `void * data () const`
returns a pointer to the data region
- `RcRegion * getRegion () const`
returns the OPEN-R memory region, should you need it

Protected Member Functions

- `SharedObjectBase ()`
constructor, protected because you shouldn't need to create this directly, just a common interface to all templates of [SharedObject](#)

Protected Attributes

- `RcRegion * rcr`

the pointer to the shared memory region this is in charge of

Private Member Functions

- [SharedObjectBase](#) (const [SharedObjectBase](#) &)
this shouldn't be called...
- [SharedObjectBase](#) & operator= (const [SharedObjectBase](#) &)
this shouldn't be called...

7.141.2 Constructor & Destructor Documentation

7.141.2.1 [SharedObjectBase::SharedObjectBase](#) () [inline, protected]

constructor, protected because you shouldn't need to create this directly, just a common interface to all templates of [SharedObject](#)

Definition at line 15 of file [SharedObject.h](#).

References [rcr](#).

7.141.2.2 [SharedObjectBase::SharedObjectBase](#) (const [SharedObjectBase](#) &) [private]

this shouldn't be called...

7.141.3 Member Function Documentation

7.141.3.1 void* [SharedObjectBase::data](#) () const [inline]

returns a pointer to the data region

Definition at line 11 of file [SharedObject.h](#).

References [rcr](#).

7.141.3.2 [RCRegion*](#) [SharedObjectBase::getRegion](#) () const [inline]

returns the OPEN-R memory region, should you need it

Definition at line 12 of file [SharedObject.h](#).

References [rcr](#).

7.141.3.3 [SharedObjectBase](#)& SharedObjectBase::operator= (const [SharedObjectBase](#) &) [private]

this shouldn't be called...

7.141.4 Member Data Documentation

7.141.4.1 RCRegion* [SharedObjectBase::rcr](#) [protected]

the pointer to the shared memory region this is in charge of

Definition at line 16 of file SharedObject.h.

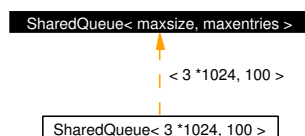
The documentation for this class was generated from the following file:

- [SharedObject.h](#)

7.142 SharedQueue< maxsize, maxentries > Class Template Reference

```
#include <SharedQueue.h>
```

Inheritance diagram for SharedQueue< maxsize, maxentries >:



7.142.1 Detailed Description

```
template<unsigned int maxsize, unsigned int maxentries> class SharedQueue<
maxsize, maxentries >
```

SharedQueue is a shared memory message buffer for interprocess communication.

This doesn't gain you much functionality over the Aperios messages - in fact, you lose some because you have to poll this for new data whereas Aperios will notify you.

However, the big advantage of this is that you don't have any lag time from the system in sending the message from one process to the other - it's instantly available. Also, you can poll for new data from within a loop, whereas you cannot check to see if you have any messages waiting from Aperios until you exit to the system.

This makes no assumptions about the data to be stored - it's all just an array of bytes

Assumptions are made about reception usage: all entries will be processed and then cleared so that the queue is emptied. You cannot pop the front, only clear the whole thing. This assumption allows much simpler/faster access. (Otherwise we need to implement a circular buffer and worry about wrap around. *shrug* Not too hard, but unnecessary complication for now.)

Definition at line 27 of file SharedQueue.h.

Public Member Functions

- [SharedQueue](#) ()
constructor
- template<class T> bool [SharedQueue](#) (const T &obj)
inserts a class into the queue

- void * [reserve](#) (unsigned int size)
reserves a number of bytes in the queue, LEAVES QUEUE LOCKED, call done when finished
- void [done](#) ()
call this when you're finished filling in a buffer you received from reserve
- bool [clear](#) (unsigned int taken)
checks to see if the number taken by the reader is the number added, and then clears
- unsigned int [size](#) () const
current number of entries
- unsigned int [size](#) (unsigned int i) const
size of entry i
- const void * [data](#) (unsigned int i) const
data of entry i

Static Public Attributes

- const unsigned int [MAX_SIZE](#) = maxsize
maximum capacity of data storage for the queue
- const unsigned int [MAX_ENTRIES](#) = maxentries
maximum number of entries that can be queued

Protected Types

- typedef [LockScope](#)< ProcessID::NumProcesses > [AutoLock](#)
for convenience in locking functions

Protected Member Functions

- unsigned int [buf_end](#) ()
the first free byte in buf

- unsigned int [round_up](#) (unsigned int sz)
sz rounded up to the nearest word (makes sz divisible by sizeof(int))

Protected Attributes

- [MutexLock](#)< ProcessID::NumProcesses > [lock](#)
provides mutual exclusion on non-const functions
- char [buf](#) [[MAX_SIZE](#)]
data storage
- unsigned int [len](#)
holds number of entries
- [entry_t](#) [entries](#) [[MAX_ENTRIES](#)]
holds entry information

7.142.2 Member Typedef Documentation

- 7.142.2.1** `template<unsigned int maxsize, unsigned int maxentries> typedef LockScope<ProcessID::NumProcesses> SharedQueue< maxsize, maxentries >::AutoLock [protected]`

for convenience in locking functions

Definition at line 71 of file SharedQueue.h.

7.142.3 Constructor & Destructor Documentation

- 7.142.3.1** `template<unsigned int maxsize, unsigned int maxentries> SharedQueue< maxsize, maxentries >::SharedQueue ()`

constructor

Definition at line 90 of file SharedQueue.h.

- 7.142.3.2** `template<unsigned int maxsize, unsigned int maxentries> template<class T> bool SharedQueue< maxsize, maxentries >::SharedQueue (const T & obj)`

inserts a class into the queue

7.142.4 Member Function Documentation

7.142.4.1 `template<unsigned int maxsize, unsigned int maxentries> unsigned int SharedQueue< maxsize, maxentries >::buf_end () [inline, protected]`

the first free byte in buf

Definition at line 63 of file SharedQueue.h.

7.142.4.2 `template<unsigned int maxsize, unsigned int maxentries> bool SharedQueue< maxsize, maxentries >::clear (unsigned int taken)`

checks to see if the number taken by the reader is the number added, and then clears

This will allow you to process entries without locking the entire queue, but still not worry about missing one (or more) if it was added while you were processing the last one.

Definition at line 124 of file SharedQueue.h.

References SharedQueue< maxsize, maxentries >::AutoLock, ProcessID::getID(), SharedQueue< maxsize, maxentries >::len, SharedQueue< maxsize, maxentries >::lock, and SharedQueue< maxsize, maxentries >::size().

7.142.4.3 `template<unsigned int maxsize, unsigned int maxentries> const void* SharedQueue< maxsize, maxentries >::data (unsigned int i) const [inline]`

data of entry *i*

Definition at line 59 of file SharedQueue.h.

7.142.4.4 `template<unsigned int maxsize, unsigned int maxentries> void SharedQueue< maxsize, maxentries >::done () [inline]`

call this when you're finished filling in a buffer you received from reserve

Definition at line 44 of file SharedQueue.h.

7.142.4.5 `template<unsigned int maxsize, unsigned int maxentries> void * SharedQueue< maxsize, maxentries >::reserve (unsigned int size)`

reserves a number of bytes in the queue, LEAVES QUEUE LOCKED, call done when finished

Definition at line 111 of file SharedQueue.h.

References SharedQueue< maxsize, maxentries >::AutoLock, SharedQueue< maxsize, maxentries >::buf, SharedQueue< maxsize, maxentries >::buf_end(), SharedQueue< maxsize, maxentries >::entries, ProcessID::getID(), SharedQueue< maxsize, maxentries >::len, SharedQueue< maxsize, maxentries >::lock, SharedQueue< maxsize, maxentries >::MAX_ENTRIES, SharedQueue< maxsize, maxentries >::MAX_SIZE, and SharedQueue< maxsize, maxentries >::round_up().

7.142.4.6 `template<unsigned int maxsize, unsigned int maxentries> unsigned int SharedQueue< maxsize, maxentries >::round_up (unsigned int sz) [inline, protected]`

sz rounded up to the nearest word (makes *sz* divisible by sizeof(int))

Definition at line 66 of file SharedQueue.h.

7.142.4.7 `template<unsigned int maxsize, unsigned int maxentries> unsigned int SharedQueue< maxsize, maxentries >::size (unsigned int i) const [inline]`

size of entry *i*

Definition at line 56 of file SharedQueue.h.

7.142.4.8 `template<unsigned int maxsize, unsigned int maxentries> unsigned int SharedQueue< maxsize, maxentries >::size () const [inline]`

current number of entries

Definition at line 53 of file SharedQueue.h.

7.142.5 Member Data Documentation

7.142.5.1 `template<unsigned int maxsize, unsigned int maxentries> char SharedQueue< maxsize, maxentries >::buf[MAX_SIZE] [protected]`

data storage

Definition at line 74 of file SharedQueue.h.

7.142.5.2 `template<unsigned int maxsize, unsigned int maxentries> entry_t
SharedQueue< maxsize, maxentries >::entries[MAX_ENTRIES]
[protected]`

holds entry information

Definition at line 86 of file SharedQueue.h.

7.142.5.3 `template<unsigned int maxsize, unsigned int maxentries> unsigned
int SharedQueue< maxsize, maxentries >::len [protected]`

holds number of entries

Definition at line 83 of file SharedQueue.h.

7.142.5.4 `template<unsigned int maxsize, unsigned int maxentries>
MutexLock<ProcessID::NumProcesses> SharedQueue< maxsize,
maxentries >::lock [protected]`

provides mutual exclusion on non-const functions

Definition at line 69 of file SharedQueue.h.

7.142.5.5 `template<unsigned int maxsize, unsigned int maxentries> const
unsigned int SharedQueue< maxsize, maxentries >::MAX_ENTRIES
= maxentries [static]`

maximum number of entries that can be queued

Definition at line 32 of file SharedQueue.h.

7.142.5.6 `template<unsigned int maxsize, unsigned int maxentries> const
unsigned int SharedQueue< maxsize, maxentries >::MAX_SIZE =
maxsize [static]`

maximum capacity of data storage for the queue

Definition at line 30 of file SharedQueue.h.

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

- [SharedQueue.h](#)

7.143 SharedQueue< maxsize, maxentries >::entry_t Struct Reference

```
#include <SharedQueue.h>
```

7.143.1 Detailed Description

template<unsigned int maxsize, unsigned int maxentries> struct SharedQueue<maxsize, maxentries >::entry_t

entry information

Definition at line 77 of file SharedQueue.h.

Public Attributes

- unsigned int [offset](#)
offset within [SharedQueue::buf](#)
- unsigned int [size](#)
size of entry

7.143.2 Member Data Documentation

7.143.2.1 **template<unsigned int maxsize, unsigned int maxentries> unsigned int [SharedQueue< maxsize, maxentries >::entry_t::offset](#)**

offset within [SharedQueue::buf](#)

Definition at line 78 of file SharedQueue.h.

7.143.2.2 **template<unsigned int maxsize, unsigned int maxentries> unsigned int [SharedQueue< maxsize, maxentries >::entry_t::size](#)**

size of entry

Definition at line 79 of file SharedQueue.h.

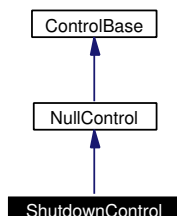
The documentation for this struct was generated from the following file:

- [SharedQueue.h](#)

7.144 ShutdownControl Class Reference

```
#include <ShutdownControl.h>
```

Inheritance diagram for ShutdownControl:



7.144.1 Detailed Description

when activated, this will cause the aibo to shut down

Definition at line 8 of file ShutdownControl.h.

Public Member Functions

- [ShutdownControl](#) ()
constructor
- [ShutdownControl](#) (const std::string &n)
constructor
- [ShutdownControl](#) (const std::string &n, const std::string &d)
constructor
- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#), [Socket](#) *)
calls [doSelect\(\)](#)
- virtual [ControlBase](#) * [doSelect](#) ()
shuts down

7.144.2 Constructor & Destructor Documentation

7.144.2.1 ShutdownControl::ShutdownControl () [inline]

constructor

Definition at line 11 of file ShutdownControl.h.

7.144.2.2 ShutdownControl::ShutdownControl (const std::string & n) [inline]

constructor

Definition at line 12 of file ShutdownControl.h.

7.144.2.3 ShutdownControl::ShutdownControl (const std::string & n, const std::string & d) [inline]

constructor

Definition at line 13 of file ShutdownControl.h.

7.144.3 Member Function Documentation

7.144.3.1 virtual [ControlBase](#)* ShutdownControl::activate ([MotionManager::MC_ID](#), [Socket](#) *) [inline, virtual]

calls [doSelect\(\)](#)

Reimplemented from [NullControl](#).

Definition at line 15 of file ShutdownControl.h.

References [doSelect\(\)](#).

7.144.3.2 [ControlBase](#) * ShutdownControl::doSelect () [virtual]

shuts down

Reimplemented from [NullControl](#).

Definition at line 4 of file ShutdownControl.cc.

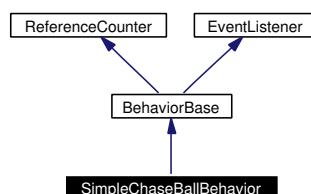
The documentation for this class was generated from the following files:

- [ShutdownControl.h](#)
- [ShutdownControl.cc](#)

7.145 SimpleChaseBallBehavior Class Reference

```
#include <SimpleChaseBallBehavior.h>
```

Inheritance diagram for SimpleChaseBallBehavior:



7.145.1 Detailed Description

A simple behavior to chase after any objects seen by the vision system.

Similar to [ChaseBallBehavior](#), but this one doesn't try to move the head, so it's a little more... simple. However, it does make sure to take into account which direction the head is pointing when it sees the object.

Definition at line 19 of file SimpleChaseBallBehavior.h.

Public Member Functions

- [SimpleChaseBallBehavior](#) ()
constructor
- virtual [~SimpleChaseBallBehavior](#) ()
destructor
- virtual void [DoStart](#) ()
adds a headpointer and a walker, and a listens for vision events
- virtual void [DoStop](#) ()
removes motion commands and stops listening
- virtual void [processEvent](#) (const [EventBase](#) &event)
sets the head to point at the object and sets the body to move where the head points
- virtual std::string [getName](#) () const
Identifies the behavior in menus and such.

Protected Attributes

- [MotionManager::MC_ID](#) `walker_id`
a *WalkMC* object

7.145.2 Constructor & Destructor Documentation

7.145.2.1 `SimpleChaseBallBehavior::SimpleChaseBallBehavior ()` [inline]

constructor

Definition at line 22 of file `SimpleChaseBallBehavior.h`.

References `walker_id`.

7.145.2.2 `virtual SimpleChaseBallBehavior::~~SimpleChaseBallBehavior ()` [inline, virtual]

destructor

Definition at line 26 of file `SimpleChaseBallBehavior.h`.

7.145.3 Member Function Documentation

7.145.3.1 `virtual void SimpleChaseBallBehavior::DoStart ()` [inline, virtual]

adds a headpointer and a walker, and a listens for vision events

Reimplemented from [BehaviorBase](#).

Definition at line 29 of file `SimpleChaseBallBehavior.h`.

References `EventRouter::addListener()`, `MotionManager::addMotion()`, `BehaviorBase::DoStart()`, `Vision::enableEvents()`, `erouter`, `motman`, `vision`, `EventBase::vision-EGID`, and `walker_id`.

7.145.3.2 `virtual void SimpleChaseBallBehavior::DoStop ()` [inline, virtual]

removes motion commands and stops listening

Reimplemented from [BehaviorBase](#).

Definition at line 40 of file `SimpleChaseBallBehavior.h`.

References BehaviorBase::DoStop(), erouter, EventRouter::forgetListener(), motman, MotionManager::removeMotion(), and walker_id.

7.145.3.3 virtual std::string SimpleChaseBallBehavior::getName () const [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 57 of file SimpleChaseBallBehavior.h.

7.145.3.4 virtual void SimpleChaseBallBehavior::processEvent (const [EventBase](#) & event) [inline, virtual]

sets the head to point at the object and sets the body to move where the head points

Reimplemented from [BehaviorBase](#).

Definition at line 47 of file SimpleChaseBallBehavior.h.

References [EventBase::getGeneratorID\(\)](#), [EventBase::getTypeID\(\)](#), [ERS210Info::HeadOffset](#), [WorldState::outputs](#), [ERS210Info::PanOffset](#), [state](#), [EventBase::statusETID](#), [EventBase::visionEGID](#), and [walker_id](#).

7.145.4 Member Data Documentation

7.145.4.1 [MotionManager::MC_ID](#) SimpleChaseBallBehavior::walker_id [protected]

a [WalkMC](#) object

Definition at line 60 of file SimpleChaseBallBehavior.h.

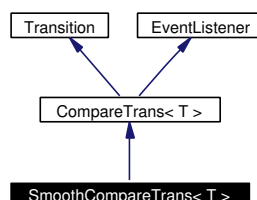
The documentation for this class was generated from the following file:

- [SimpleChaseBallBehavior.h](#)

7.146 SmoothCompareTrans< T > Class Template Reference

```
#include <SmoothCompareTrans.h>
```

Inheritance diagram for SmoothCompareTrans< T >:



7.146.1 Detailed Description

template<class T> class SmoothCompareTrans< T >

A subclass of [CompareTrans](#), which provides monitoring of exponentially weighted averages to a threshold.

Has the additional requirement that template types must supply `operator*=(float)` and `operator+=(T)` for the weighted average

The gamma parameter is how much to weight the preceeding value - 1 will cause it to never update, 0 will cause it to only look at most recent. So, the lower the value, the faster it is to switch, but more prone to noise

Definition at line 15 of file `SmoothCompareTrans.h`.

Public Member Functions

- [SmoothCompareTrans](#) ([StateNode](#) *source, [StateNode](#) *destination, const T *monitor, typename [SmoothCompareTrans< T >::Test_t](#) test, const T &value, const [EventBase](#) &poll, float gammap=0)

constructor, see class notes for information

- virtual void [processEvent](#) (const [EventBase](#) &e)

don't care about the event, just a pulse to check the values

Protected Attributes

- T [avg](#)
the current running average
- const T * [realmon](#)
pointer to the value being monitored
- float [g](#)
the gamma value controlling the exponential average, see the class documentation at the top

Private Member Functions

- [SmoothCompareTrans](#) (const [SmoothCompareTrans](#) &node)
don't call this
- [SmoothCompareTrans](#) operator= (const [SmoothCompareTrans](#) &node)
don't call this

7.146.2 Constructor & Destructor Documentation

7.146.2.1 `template<class T> SmoothCompareTrans< T
>::SmoothCompareTrans (StateNode * source, StateNode *
destination, const T * monitor, typename SmoothCompareTrans< T
>::Test_t test, const T & value, const EventBase & poll, float gammap
= 0) [inline]`

constructor, see class notes for information

Definition at line 18 of file `SmoothCompareTrans.h`.

References `SmoothCompareTrans< T >::avg`, `SmoothCompareTrans< T >::g`, and `SmoothCompareTrans< T >::realmon`.

7.146.2.2 `template<class T> SmoothCompareTrans< T
>::SmoothCompareTrans (const SmoothCompareTrans< T > &
node) [private]`

don't call this

7.146.3 Member Function Documentation

7.146.3.1 `template<class T> SmoothCompareTrans SmoothCompareTrans< T >::operator= (const SmoothCompareTrans< T > & node)`
[private]

don't call this

7.146.3.2 `template<class T> virtual void SmoothCompareTrans< T >::processEvent (const EventBase & e)` [inline, virtual]

don't care about the event, just a pulse to check the values

Reimplemented from [CompareTrans](#)< T >.

Definition at line 23 of file [SmoothCompareTrans.h](#).

References [SmoothCompareTrans](#)< T >::avg, [SmoothCompareTrans](#)< T >::g, [CompareTrans](#)< T >::processEvent(), and [SmoothCompareTrans](#)< T >::realmon.

7.146.4 Member Data Documentation

7.146.4.1 `template<class T> T SmoothCompareTrans< T >::avg`
[protected]

the current running average

Definition at line 31 of file [SmoothCompareTrans.h](#).

7.146.4.2 `template<class T> float SmoothCompareTrans< T >::g`
[protected]

the gamma value controlling the exponential average, see the class documentation at the top

Definition at line 35 of file [SmoothCompareTrans.h](#).

7.146.4.3 `template<class T> const T* SmoothCompareTrans< T >::realmon`
[protected]

pointer to the value being monitored

Definition at line 32 of file [SmoothCompareTrans.h](#).

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

- [SmoothCompareTrans.h](#)

7.147 Socket Class Reference

```
#include <Socket.h>
```

7.147.1 Detailed Description

Tekkotsu wireless Socket class.

For more information on using wireless, please read the following tutorials:

- [TekkotsuMon](#)
- [TCP/IP](#)
- [Remote Processing OPENR](#) Tekkotsu [Wireless](#) and Remote Processing OPENR provide different interfaces to comparable wireless functionality.

Definition at line 47 of file Socket.h.

Public Member Functions

- [Socket](#) (int sockn)
constructor
- [~Socket](#) ()
destructor
- byte * [getWriteBuffer](#) (int bytesreq)
- void [write](#) (int size)
writes the specified number of bytes starting at the pointer returned.
- int [read](#) ()
Blocking read.
- byte * [getReadBuffer](#) ()
getReadBuffer is used with blocking read's
- void [init](#) ()
initialize socket member variables. This is different from the constructor since sockets are reused
- int [setFlushType](#) (FlushType_t fType)
Chooses between blocking and non-blocking input, output.

- void `setTextForward` ()
causes this socket to forward output to stdout if it is not connected, call `setForward(NULL)` to unset
- void `setForward` (Socket *forsock)
causes this socket to forward output to sock if it is not connected, pass NULL to unset
- void `setVerbosity` (int verbose)
Picks a level of verbosity for filtering pprintf commands.
- int `write` (const byte *buf, int size)
- int `read` (byte *buf, int size)
Blocking read.
- int `printf` (const char *fmt,...)
- int `vprintf` (const char *fmt, va_list al)
- int `pprintf` (int vlevel, const char *fmt,...)
Similar to printf, except it takes an extra first argument.
- void `flush` ()

Public Attributes

- int `sock`
unique non-negative integer representing socket. Serves as index into socket Objects array

Private Member Functions

- `Socket` (const `Socket` &)
copy constructor, don't call
- `Socket` & `operator=` (const `Socket` &)
assignment operator, don't call

Private Attributes

- TransportType_t [trType](#)
private ALOKL_TODO
- FlushType_t [flType](#)
private ALOKL_TODO
- int [verbosity](#)
private ALOKL_TODO
- antModuleRef [endpoint](#)
private ALOKL_TODO
- ConnectionState [state](#)
private ALOKL_TODO
- int [sendBufSize](#)
private ALOKL_TODO
- int [recvBufSize](#)
private ALOKL_TODO
- int [sendSize](#)
private ALOKL_TODO
- int [recvSize](#)
private ALOKL_TODO
- int [writeSize](#)
private ALOKL_TODO
- int [readSize](#)
private ALOKL_TODO
- bool [tx](#)
private ALOKL_TODO
- bool [rx](#)
private ALOKL_TODO
- antSharedBuffer [sendBuffer](#)
private ALOKL_TODO

- antSharedBuffer [recvBuffer](#)
private ALOKL_TODO
- byte * [sendData](#)
private ALOKL_TODO
- byte * [recvData](#)
private ALOKL_TODO
- byte * [readData](#)
private ALOKL_TODO
- byte * [writeData](#)
private ALOKL_TODO
- int [server_port](#)
private ALOKL_TODO
- int(* [rcvcbckfn](#))(char *, int)
private ALOKL_TODO
- bool [textForward](#)
private ALOKL_TODO
- char * [textForwardBuf](#)
private ALOKL_TODO
- [Socket](#) * [forwardSock](#)
private ALOKL_TODO

Friends

- class [Wireless](#)

7.147.2 Constructor & Destructor Documentation

7.147.2.1 [Socket::Socket](#) (int *sockn*) [[inline](#)]

constructor

Definition at line 55 of file Socket.h.

References SocketNS::CONNECTION_CLOSED, endpoint, flType, SocketNS::FLUSH_NONBLOCKING, forwardSock, rcvcbckfn, readData, readSize, recvBuffer, recvBufSize, recvData, recvSize, rx, sendBuffer, sendBufSize, sendData, sendSize, server_port, sock, state, textForward, textForwardBuf, trType, tx, verbosity, writeData, and writeSize.

7.147.2.2 Socket::~~Socket () [inline]

destructor

Definition at line 63 of file Socket.h.

7.147.2.3 Socket::Socket (const Socket &) [private]

copy constructor, don't call

7.147.3 Member Function Documentation

7.147.3.1 void Socket::flush ()

All write commands on the socket will implicitly call this. You don't need to call it, unless you're implementing your own write

Definition at line 66 of file Socket.cc.

References Wireless::blockingSend(), SocketNS::CONNECTION_CONNECTED, flType, SocketNS::FLUSH_NONBLOCKING, forwardSock, Wireless::send(), sendData, sendSize, sock, state, tx, wireless, writeData, and writeSize.

7.147.3.2 byte * Socket::getReadBuffer ()

getReadBuffer is used with blocking read's

The [read\(void\)](#) and getReadBuffer combo eliminates one buffer copy. You don't need to use getReadBuffer with [read\(byte*, int\)](#)

Blocking read is currently broken - it will be fixed in the next release

Returns:

pointer to the buffer the previous call to blocking read wrote into or NULL if no data was read

Definition at line 44 of file Socket.cc.

7.147.3.3 `byte * Socket::getWriteBuffer (int bytesreq)`

The `getWriteBuffer-write(int)` combo eliminates one buffer copy. You don't need to use `getWriteBuffer` with `write(byte*, int)`

Returns:

pointer to the current position in the current write buffer for this socket or NULL on error

Parameters:

bytesreq maximum number of bytes the caller intends to set before the write method is called

Definition at line 11 of file `Socket.cc`.

References `SocketNS::CONNECTION_CONNECTED`, `forwardSock`, `sendBufSize`, `state`, `textForward`, `textForwardBuf`, `writeData`, and `writeSize`.

7.147.3.4 `void Socket::init ()`

initialize socket member variables. This is different from the constructor since sockets are reused

Definition at line 51 of file `Socket.cc`.

References `sendSize`, and `writeSize`.

7.147.3.5 `Socket& Socket::operator= (const Socket &) [private]`

assignment operator, don't call

7.147.3.6 `int Socket::pprintf (int vlevel, const char * fmt, ...)`

Similar to `printf`, except it takes an extra first argument.

If *vlevel* is than or equal to the current verbosity level, the string will be printed else it will be ignored

Parameters:

vlevel if (*vlevel* ≤ *verbosity*) print, else ignore

fmt same as the standard `printf`'s format string

Definition at line 91 of file `Socket.cc`.

References `printf()`, and *verbosity*.

7.147.3.7 int Socket::printf (const char * *fmt*, ...)

It's standard stuff. man 3 printf on most systems should give you more information

Definition at line 105 of file Socket.cc.

References [vprintf\(\)](#).

7.147.3.8 int Socket::read (byte * *buf*, int *size*)

Blocking read.

You might want to consider the [read\(void\)](#) and [getReadBuffer](#) combo if you call this often

Blocking read is currently broken - it will be fixed in the next release

Parameters:

buf buffer to write from

size number of bytes to write

Returns:

number of bytes actually read

Definition at line 156 of file Socket.cc.

7.147.3.9 int Socket::read ()

Blocking read.

Tries to read upto receive buffer size worth of data from this socket.

Blocking read is currently broken - it will be fixed in the next release

Returns:

number of bytes read or -1 on error

Definition at line 38 of file Socket.cc.

7.147.3.10 int Socket::setFlushType (FlushType_t *fType*)

Chooses between blocking and non-blocking input, output.

This function can only be called when a socket is disconnected, since it is a bad idea to mix blocking and non-blocking input, output. The default for a socket is non-blocking

Returns:

0 on success

Definition at line 58 of file Socket.cc.

References SocketNS::CONNECTION_CLOSED, flType, and state.

7.147.3.11 void Socket::setForward (Socket *forsock) [inline]

causes this socket to forward output to *sock* if it is not connected, pass NULL to unset

Definition at line 116 of file Socket.h.

References forwardSock, and textForward.

7.147.3.12 void Socket::setTextForward () [inline]

causes this socket to forward output to stdout if it is not connected, call setForward(NULL) to unset

Definition at line 113 of file Socket.h.

References forwardSock, and textForward.

7.147.3.13 void Socket::setVerbosity (int verbose) [inline]

Picks a level of verbosity for filtering pprintf commands.

The higher the verbosity, the more the number of messages printed. This is useful for filtering out non-important messages with very little processor cost. Default is 0.

Parameters:

verbose the higher the value of verbose, the more the output

Definition at line 125 of file Socket.h.

References verbosity.

7.147.3.14 int Socket::vprintf (const char *fmt, va_list al)

It's standard stuff. man 3 printf on most systems should give you more information

Definition at line 115 of file Socket.cc.

References SocketNS::CONNECTION_CONNECTED, flush(), forwardSock, sendBufSize, state, textForward, writeData, and writeSize.

7.147.3.15 int Socket::write (const byte * *buf*, int *size*)

You might want to consider the getWriteBuffer-write(int) combo if you call this often

Parameters:

buf buffer to write from

size number of bytes to write

Returns:

the number of bytes actually written or -1 on error

Definition at line 138 of file Socket.cc.

References SocketNS::CONNECTION_CONNECTED, forwardSock, getWriteBuffer(), state, textForward, and write().

7.147.3.16 void Socket::write (int *size*)

writes the specified number of bytes starting at the pointer returned.

in a (prior) call to getWriteBufer

Parameters:

size number of bytes to be sent from the current write buffer

Definition at line 26 of file Socket.cc.

References flush(), textForwardBuf, and writeSize.

7.147.4 Friends And Related Function Documentation**7.147.4.1 friend class [Wireless](#) [*friend*]**

Definition at line 48 of file Socket.h.

7.147.5 Member Data Documentation**7.147.5.1 antModuleRef [Socket::endpoint](#) [*private*]**

private ALOKL_TODO

Definition at line 181 of file Socket.h.

7.147.5.2 FlushType.t Socket::ffType [private]

private ALOKL_TODO

Definition at line 177 of file Socket.h.

7.147.5.3 Socket* Socket::forwardSock [private]

private ALOKL_TODO

Definition at line 195 of file Socket.h.

7.147.5.4 int(* Socket::recvbckfn)(char*, int) [private]

private ALOKL_TODO

7.147.5.5 byte* Socket::readData [private]

private ALOKL_TODO

Definition at line 189 of file Socket.h.

7.147.5.6 int Socket::readSize [private]

private ALOKL_TODO

Definition at line 184 of file Socket.h.

7.147.5.7 antSharedBuffer Socket::recvBuffer [private]

private ALOKL_TODO

Definition at line 187 of file Socket.h.

7.147.5.8 int Socket::recvBufSize [private]

private ALOKL_TODO

Definition at line 184 of file Socket.h.

7.147.5.9 byte * Socket::recvData [private]

private ALOKL_TODO

Definition at line 188 of file Socket.h.

7.147.5.10 `int Socket::recvSize` [private]

private ALOKL_TODO

Definition at line 184 of file Socket.h.

7.147.5.11 `bool Socket::rx` [private]

private ALOKL_TODO

Definition at line 185 of file Socket.h.

7.147.5.12 `antSharedBuffer Socket::sendBuffer` [private]

private ALOKL_TODO

Definition at line 187 of file Socket.h.

7.147.5.13 `int Socket::sendBufSize` [private]

private ALOKL_TODO

Definition at line 184 of file Socket.h.

7.147.5.14 `byte* Socket::sendData` [private]

private ALOKL_TODO

Definition at line 188 of file Socket.h.

7.147.5.15 `int Socket::sendSize` [private]

private ALOKL_TODO

Definition at line 184 of file Socket.h.

7.147.5.16 `int Socket::server_port` [private]

private ALOKL_TODO

Definition at line 190 of file Socket.h.

7.147.5.17 int [Socket::sock](#)

unique non-negative integer representing socket. Serves as index into socket Objects array

Definition at line 51 of file Socket.h.

7.147.5.18 ConnectionState [Socket::state](#) [private]

private ALOKL_TODO

Definition at line 182 of file Socket.h.

7.147.5.19 bool [Socket::textForward](#) [private]

private ALOKL_TODO

Definition at line 193 of file Socket.h.

7.147.5.20 char* [Socket::textForwardBuf](#) [private]

private ALOKL_TODO

Definition at line 194 of file Socket.h.

7.147.5.21 TransportType_t [Socket::trType](#) [private]

private ALOKL_TODO

Definition at line 176 of file Socket.h.

7.147.5.22 bool [Socket::tx](#) [private]

private ALOKL_TODO

Definition at line 185 of file Socket.h.

7.147.5.23 int [Socket::verbosity](#) [private]

private ALOKL_TODO

Definition at line 179 of file Socket.h.

7.147.5.24 `byte * Socket::writeData` [private]

private ALOKL_TODO

Definition at line 189 of file Socket.h.

7.147.5.25 `int Socket::writeSize` [private]

private ALOKL_TODO

Definition at line 184 of file Socket.h.

The documentation for this class was generated from the following files:

- [Socket.h](#)
- [Socket.cc](#)

7.148 SoundManager Class Reference

```
#include <SoundManager.h>
```

7.148.1 Detailed Description

Provides sound effects and caching services, as well as mixing buffers for the [Sound-Play](#) process.

Provides easy methods for playing back sounds, either from files on the memory stick, or from dynamically generated buffers. You can chain playback commands so that when one sound finishes, another picks up automatically. This might be handy if, say, someone wants to write an MP3 player ;) The sounds would be too large to load into memory all at once, but you could load a block at a time and chain them so it seamlessly moves from one to the other.

All functions will attempt to lock the SoundManager.

Todo

- Volume control, variable playback speed, support more wav file formats (all go together)
- Add functions to hand out regions to be filled out to avoid copying into the buffer.

Definition at line 32 of file SoundManager.h.

Public Types

- typedef [SoundManagerMsg::Snd_ID](#) [Snd_ID](#)
This is used for referring to sound data so you can start playing it or release it.
- typedef unsigned short [Play_ID](#)
This is for referring to instances of the play command so you can stop, pause, or monitor progress (later versions will send events upon completion).
- enum [MixMode_t](#) { [Fast](#), [Quality](#) }
Used to set the mode for mixing multiple sound channels.
- enum [QueueMode_t](#) { [Enqueue](#), [Pause](#), [Stop](#), [Override](#) }

Public Member Functions

- [SoundManager](#) ()
constructor

- void [InitAccess](#) (OSubject *subj)
Needed to send sounds to the [SoundPlay](#) process.
- [Snd_ID LoadFile](#) (const char *name)
loads a wav file (if it matches [Config::sound_config](#) settings - can't do resampling yet)
- [Snd_ID LoadBuffer](#) (const char buf[], unsigned int len)
loads raw samples from a buffer (assumes matches [Config::sound_config](#) settings)
- void [ReleaseFile](#) (const char *name)
Marks the sound buffer to be released after the last play command completes (or right now if not being played).
- void [Release](#) ([Snd_ID](#) id)
Marks the sound buffer to be released after the last play command completes (or right now if not being played).
- [Play_ID PlayFile](#) (const char *name)
play a wav file (if it matches [Config::sound_config](#) settings - can't do resampling yet)
- [Play_ID PlayBuffer](#) (const char buf[], unsigned int len)
loads raw samples from a buffer (assumes buffer matches [Config::sound_config](#) settings)
- [Play_ID Play](#) ([Snd_ID](#) id)
plays a previously loaded buffer or file
- [Play_ID ChainFile](#) ([Play_ID](#) base, const char *next)
allows automatic queuing of sounds - good for dynamic sound sources!
- [Play_ID ChainBuffer](#) ([Play_ID](#) base, const char buf[], unsigned int len)
allows automatic queuing of sounds - good for dynamic sound sources!
- [Play_ID Chain](#) ([Play_ID](#) base, [Snd_ID](#) next)
allows automatic queuing of sounds - good for dynamic sound sources!
- void [StopPlay](#) ()
Lets you stop playback of all sounds.
- void [StopPlay](#) ([Play_ID](#) id)
Lets you stop playback of a sound.

- void [PausePlay](#) ([Play_ID](#) id)
Lets you pause playback.
- void [ResumePlay](#) ([Play_ID](#) id)
Lets you resume playback.
- void [SetMode](#) (unsigned int max_channels, [MixMode_t](#) mixer_mode, [Queue-Mode_t](#) queuing_mode)
Lets you control the maximum number of channels (currently playing sounds), method for mixing (applies when max_chan>1), and queuing method (for when overflow channels).
- unsigned int [GetRemainTime](#) ([Play_ID](#) id) const
Gives the time until the sound finishes, in milliseconds. Subtract 32 to get guaranteed valid time for this ID.
- unsigned int [CopyTo](#) (OSoundVectorData *data)
Copies the sound data to the OPENR buffer, ready to be passed to the system, only called by [SoundPlay](#).
- void [ReceivedMsg](#) (const ONotifyEvent &event)
updates internal data structures on the [SoundPlay](#) side - you shouldn't be calling this
- unsigned int [GetNumPlaying](#) ()
number of sounds currently playing

Static Public Attributes

- const [Snd_ID](#) [invalid_Snd_ID](#) = ([Snd_ID](#))-1
for reporting errors
- const [Snd_ID](#) [MAX_SND](#) = 50
the number of sounds that can be loaded at any given time
- const [Play_ID](#) [invalid_Play_ID](#) = ([Play_ID](#))-1
for reporting errors
- const [Play_ID](#) [MAX_PLAY](#) = 256
the number of sounds that can be enqueued at the same time (see [MixMode_t](#))
- const unsigned int [MAX_NAME_LEN](#) = 64
maximum length of a path

Protected Types

- typedef [ListMemBuf](#)< [SoundData](#), [MAX_SND](#), [Snd_ID](#) > [sndlist_t](#)
For convenience.
- typedef [ListMemBuf](#)< [PlayState](#), [MAX_PLAY](#), [Play_ID](#) > [playlist_t](#)
For convenience.
- typedef [ListMemBuf](#)< [Play_ID](#), [MAX_PLAY](#), [Play_ID](#) > [chanlist_t](#)
For convenience.

Protected Member Functions

- [Snd_ID lookup](#) (const char *name) const
Looks to see if name matches any of the sounds in sndlist.
- [Snd_ID lookupPath](#) (const char *path) const
Looks to see if name matches any of the sounds in sndlist (assumes is absolute path).
- void [selectChannels](#) (std::vector< [Play_ID](#) > &mix)
selects which of the channels are actually to be mixed together, depending on queue_-mode
- void [updateChannels](#) (const std::vector< [Play_ID](#) > &mixs, size_t used)
update the offsets of sounds which weren't mixed (when needed depending on queue_-mode)
- bool [endPlay](#) ([Play_ID](#) id)
called when a buffer end is reached, may reset buffer to next in chain, or just [Stop-Play\(\)](#)
- [SoundManager](#) (const [SoundManager](#) &)
don't call
- [SoundManager](#) operator= (const [SoundManager](#) &)
don't call

Static Protected Member Functions

- [RCRegion](#) * [initRegion](#) (unsigned int size)

Sets up a shared region to hold a sound - rounds to nearest page size.

- `const char * makePath (const char *name, char tmp[MAX_NAME_LEN])`
prepends `config.sound.root` to the name if necessary

Protected Attributes

- `sndlist_t sndlist`
Holds a list of all currently loaded sounds.
- `playlist_t playlist`
Holds a list of all sounds currently enqueued.
- `chanlist_t chanlist`
Holds a list of all currently playing sounds, ordered newest (front) to oldest(back).
- `MixMode_t mix_mode`
Current mixing mode, set by [SetMode\(\)](#);
- `QueueMode_t queue_mode`
Current queuing mode, set by [SetMode\(\)](#);
- `unsigned int max_chan`
Current maximum number of sounds to mix together.
- `MutexLock< ProcessID::NumProcesses > lock`
Prevents multiple processes from accessing at the same time.
- `OSubject * subjs [ProcessID::NumProcesses]`
For automatic transmission of shared regions to [SoundPlay](#).

7.148.2 Member Typedef Documentation

7.148.2.1 `typedef ListMemBuf<Play_ID,MAX_PLAY,Play_ID>`
`SoundManager::chanlist_t [protected]`

For convenience.

Definition at line 204 of file `SoundManager.h`.

7.148.2.2 `typedef unsigned short SoundManager::Play_ID`

This is for referring to instances of the play command so you can stop, pause, or monitor progress (later versions will send events upon completion).

Definition at line 43 of file SoundManager.h.

7.148.2.3 `typedef ListMemBuf<PlayState,MAX_PLAY,Play_ID> SoundManager::playlist_t [protected]`

For convenience.

Definition at line 200 of file SoundManager.h.

7.148.2.4 `typedef SoundManagerMsg::Snd_ID SoundManager::Snd_ID`

This is used for referring to sound data so you can start playing it or release it.

Definition at line 38 of file SoundManager.h.

7.148.2.5 `typedef ListMemBuf<SoundData,MAX_SND,Snd_ID> SoundManager::sndlist_t [protected]`

For convenience.

Definition at line 187 of file SoundManager.h.

7.148.3 Member Enumeration Documentation

7.148.3.1 `enum SoundManager::MixMode_t`

Used to set the mode for mixing multiple sound channels.

Feel free to add a higher quality mixer if you're an audiophile - I'm pretty new to sound processing

Enumeration values:

Fast uses bit shifting trick, but can result in reduced volume with more active channels, best if you set max_channels to a power of 2

Quality uses real division to maintain volume level, although still a rather naive (but relatively fast) algorithm

Definition at line 51 of file SoundManager.h.

7.148.3.2 enum [SoundManager::QueueMode_t](#)

Enumeration values:

Enqueue newer sounds are played when a channel opens up (when old sound finishes)

Pause newer sounds pause oldest sound, which continues when a channel opens

Stop newer sounds stop oldest sound

Override older sounds have play heads advanced, but don't get mixed until a channel opens

Definition at line 56 of file SoundManager.h.

7.148.4 Constructor & Destructor Documentation

7.148.4.1 [SoundManager::SoundManager](#) ()

constructor

Definition at line 19 of file SoundManager.cc.

7.148.4.2 [SoundManager::SoundManager](#) (const [SoundManager](#) &) [protected]

don't call

7.148.5 Member Function Documentation

7.148.5.1 [SoundManager::Play_ID](#) [SoundManager::Chain](#) ([Play_ID](#) *base*, [Snd_ID](#) *next*)

allows automatic queuing of sounds - good for dynamic sound sources!

if you chain more than once to the same base, the new buffers are appended to the end of the chain - the new buffer doesn't replace the current chain

Returns:

base - just for convenience of multiple calls

Definition at line 241 of file SoundManager.cc.

References [invalid_Play_ID](#), [invalid_Snd_ID](#), [Play_ID](#), and [playlist](#).

7.148.5.2 [SoundManager::Play_ID](#) [SoundManager::ChainBuffer](#) ([Play_ID](#) *base*, *const char buf[]*, *unsigned int len*)

allows automatic queuing of sounds - good for dynamic sound sources!

if you chain more than once to the same base, the new buffers are appended to the end of the chain - the new buffer doesn't replace the current chain

Returns:

base - just for convenience of multiple calls

Definition at line 221 of file SoundManager.cc.

References [invalid_Play_ID](#), [invalid_Snd_ID](#), [LoadBuffer\(\)](#), [Play_ID](#), [playlist](#), and [Snd_ID](#).

7.148.5.3 [SoundManager::Play_ID](#) [SoundManager::ChainFile](#) ([Play_ID](#) *base*, *const char * next*)

allows automatic queuing of sounds - good for dynamic sound sources!

if you chain more than once to the same base, the new buffers are appended to the end of the chain - the new buffer doesn't replace the current chain

Returns:

base - just for convenience of multiple calls

Definition at line 201 of file SoundManager.cc.

References [invalid_Play_ID](#), [invalid_Snd_ID](#), [LoadFile\(\)](#), [Play_ID](#), [playlist](#), and [Snd_ID](#).

7.148.5.4 *unsigned int* [SoundManager::CopyTo](#) ([OSoundVectorData](#) * *data*)

Copies the sound data to the OPENR buffer, ready to be passed to the system, only called by [SoundPlay](#).

Returns:

the number of active sounds

Definition at line 325 of file SoundManager.cc.

References [AutoLock](#), [chanlist](#), [config](#), [endPlay\(\)](#), [Fast](#), [ListMemBuf](#)< [PlayState](#), [MAX_PLAY](#), [Play_ID](#) >::front(), [ProcessID](#)::getID(), [lock](#), [mix_mode](#), [playlist](#), [ListMemBuf](#)< [PlayState](#), [MAX_PLAY](#), [Play_ID](#) >::pop_back(), [ListMemBuf](#)< [PlayState](#), [MAX_PLAY](#), [Play_ID](#) >::push_back(), [Config](#)::sound_config::sample_bits, [selectChannels\(\)](#), [ListMemBuf](#)< [Play_ID](#), [MAX_PLAY](#), [Play_ID](#) >::size(), [Snd_ID](#), [sndlist](#), [Config](#)::sound, and [updateChannels\(\)](#).

7.148.5.5 `bool SoundManager::endPlay (Play_ID id) [protected]`

called when a buffer end is reached, may reset buffer to next in chain, or just [StopPlay\(\)](#)

Definition at line 655 of file SoundManager.cc.

References `EventBase::audioEGID`, `config`, `erouter`, `invalid_Play_ID`, `Play_ID`, `playlist`, `EventRouter::postEvent()`, `Release()`, `Config::sound_config::sample_bits`, `Config::sound_config::sample_rate`, `Config::sound`, `EventBase::statusETID`, and `StopPlay()`.

7.148.5.6 `unsigned int SoundManager::GetNumPlaying () [inline]`

number of sounds currently playing

Definition at line 151 of file SoundManager.h.

References `chanlist`, and `ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::size()`.

7.148.5.7 `unsigned int SoundManager::GetRemainTime (Play_ID id) const`

Gives the time until the sound finishes, in milliseconds. Subtract 32 to get guaranteed valid time for this ID.

You should be passing the beginning of a chain to get proper results...

May be slightly conservative (will report too small a time) because this does not account for delay until [SoundPlay](#) picks up the message that a sound has been added.

However, it is slightly optimistic (will report too large a time) because it processes a buffer all at one go, so it could mark the sound as finished (and cause the ID to go invalid) up to `RobotInfo::SoundBufferTime` (32 ms) before the sound finishes. So subtract `SoundBufferTime` if you want to be absolutely sure the ID will still valid.

Definition at line 313 of file SoundManager.cc.

References `AutoLock`, `config`, `ProcessID::getID()`, `invalid_Play_ID`, `lock`, `playlist`, `Config::sound_config::sample_bits`, `Config::sound_config::sample_rate`, `sndlist`, and `Config::sound`.

7.148.5.8 `void SoundManager::InitAccess (OSubject * subj)`

Needed to send sounds to the [SoundPlay](#) process.

Definition at line 24 of file SoundManager.cc.

References `ProcessID::getID()`, and `subjs`.

7.148.5.9 **RCRegion * SoundManager::initRegion (unsigned int size)** [static, protected]

Sets up a shared region to hold a sound - rounds to nearest page size.

Definition at line 539 of file SoundManager.cc.

References ASSERT.

7.148.5.10 **SoundManager::Snd_ID SoundManager::LoadBuffer (const char buf[], unsigned int len)**

loads raw samples from a buffer (assumes matches [Config::sound_config](#) settings)

The sound data will be cached until [Release\(\)](#) is called a matching number of times.

This function is useful for dynamic sound sources. A copy will be made.

Definition at line 68 of file SoundManager.cc.

References [AutoLock](#), [config](#), [SoundManagerMsg::getID\(\)](#), [ProcessID::getID\(\)](#), [initRegion\(\)](#), [invalid_Snd_ID](#), [lock](#), [SoundManagerMsg::MSG_SIZE](#), [ListMemBuf< SoundData, MAX_SND, Snd_ID >::new_front\(\)](#), [region](#), [Config::sound_config::sample_bits](#), [SoundManagerMsg::setAdd\(\)](#), [sndlist](#), [Config::sound](#), [ProcessID::SoundProcess](#), and [subjs](#).

7.148.5.11 **SoundManager::Snd_ID SoundManager::LoadFile (const char * name)**

loads a wav file (if it matches [Config::sound_config](#) settings - can't do resampling yet)

Since the SoundManager does the loading, if the same file is being played more than once, only once copy is stored in memory

Parameters:

name can be either a full path, or a partial path relative to [Config::sound_config::root](#)

Returns:

ID number for future references (can also use name) The sound data will be cached until [ReleaseFile\(\)](#) or [Release\(\)](#) is called a matching number of times

Definition at line 30 of file SoundManager.cc.

References [AutoLock](#), [config](#), [WAV::GetBitsPerSample\(\)](#), [WAV::GetDataEnd\(\)](#), [WAV::GetDataStart\(\)](#), [ProcessID::getID\(\)](#), [WAV::GetSamplingRate\(\)](#), [invalid_Snd_ID](#), [LoadBuffer\(\)](#), [lock](#), [lookupPath\(\)](#), [makePath\(\)](#), [MAX_NAME_LEN](#), [Config::sound_config::sample_bits](#), [Config::sound_config::sample_rate](#), [WAV::Set\(\)](#), [Snd_ID](#), [sndlist](#), [Config::sound](#), [WAV_SUCCESS](#), and [WAVEError](#).

7.148.5.12 [SoundManager::Snd_ID](#) SoundManager::lookup (const char * *name*) const [protected]

Looks to see if *name* matches any of the sounds in sndlist.

Definition at line 548 of file SoundManager.cc.

References lookupPath(), makePath(), and MAX_NAME_LEN.

7.148.5.13 [SoundManager::Snd_ID](#) SoundManager::lookupPath (const char * *path*) const [protected]

Looks to see if *name* matches any of the sounds in sndlist (assumes is absolute path).

Definition at line 553 of file SoundManager.cc.

References ListMemBuf< SoundData, MAX_SND, Snd_ID >::begin(), ListMemBuf< SoundData, MAX_SND, Snd_ID >::end(), invalid_Snd_ID, MAX_NAME_LEN, ListMemBuf< SoundData, MAX_SND, Snd_ID >::next(), and sndlist.

7.148.5.14 const char * SoundManager::makePath (const char * *name*, char *tmp*[MAX_NAME_LEN]) [static, protected]

prepends config.sound.root to the name if necessary

Definition at line 561 of file SoundManager.cc.

References config, Config::sound_config::root, and Config::sound.

7.148.5.15 [SoundManager](#) SoundManager::operator= (const [SoundManager](#) &) [protected]

don't call

7.148.5.16 void SoundManager::PausePlay ([Play_ID](#) *id*)

Lets you pause playback.

Definition at line 282 of file SoundManager.cc.

References AutoLock, ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::begin(), chanlist, ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::end(), ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::erase(), ProcessID::getID(), invalid_Play_ID, lock, and ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::next().

7.148.5.17 [SoundManager::Play_ID](#) SoundManager::Play ([Snd_ID](#) *id*)

plays a previously loaded buffer or file

Definition at line 167 of file SoundManager.cc.

References [EventBase::activateETID](#), [ASSERT](#), [EventBase::audioEGID](#), [AutoLock](#), [chanlist](#), [erouter](#), [ProcessID::getID\(\)](#), [initRegion\(\)](#), [invalid_Play_ID](#), [invalid_Snd_ID](#), [lock](#), [SoundManagerMsg::MSG_SIZE](#), [ListMemBuf< PlayState, MAX_PLAY, Play_ID >::new_front\(\)](#), [Play_ID](#), [playlist](#), [EventRouter::postEvent\(\)](#), [ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::push_front\(\)](#), [region](#), [SoundManagerMsg::setWakeUp\(\)](#), [sndlist](#), [ProcessID::SoundProcess](#), and [subjs](#).

7.148.5.18 [SoundManager::Play_ID](#) SoundManager::PlayBuffer (const char *buf*[], unsigned int *len*)

loads raw samples from a buffer (assumes buffer matches [Config::sound_config](#) settings)

The sound data will be released after done playing

Definition at line 155 of file SoundManager.cc.

References [AutoLock](#), [ProcessID::getID\(\)](#), [invalid_Play_ID](#), [invalid_Snd_ID](#), [LoadBuffer\(\)](#), [lock](#), [Play\(\)](#), [playlist](#), [ListMemBuf< PlayState, MAX_PLAY, Play_ID >::size\(\)](#), [Snd_ID](#), and [sndlist](#).

7.148.5.19 [SoundManager::Play_ID](#) SoundManager::PlayFile (const char * *name*)

play a wav file (if it matches [Config::sound_config](#) settings - can't do resampling yet)

Will do a call to [LoadFile\(\)](#) first, and then automatically release the sound again when complete.

Parameters:

name can be either a full path, or a partial path relative to [Config::sound_config::root](#)

Returns:

ID number for future references The sound data will not be cached after done playing unless a call to [LoadFile](#) is made

Definition at line 143 of file SoundManager.cc.

References [AutoLock](#), [ProcessID::getID\(\)](#), [invalid_Play_ID](#), [invalid_Snd_ID](#), [LoadFile\(\)](#), [lock](#), [Play\(\)](#), [playlist](#), [ListMemBuf< PlayState, MAX_PLAY, Play_ID >::size\(\)](#), [Snd_ID](#), and [sndlist](#).

7.148.5.20 void SoundManager::ReceivedMsg (const ONotifyEvent & event)

updates internal data structures on the [SoundPlay](#) side - you shouldn't be calling this

Definition at line 509 of file SoundManager.cc.

References [EventBase::activateETID](#), [SoundManagerMsg::add](#), [EventBase::audio-EGID](#), [ListMemBuf< PlayState, MAX_PLAY, Play_ID >::begin\(\)](#), [SoundManagerMsg::del](#), [ListMemBuf< PlayState, MAX_PLAY, Play_ID >::end\(\)](#), [erouter](#), [SoundManagerMsg::id](#), [SoundManagerMsg::MSG_SIZE](#), [ListMemBuf< PlayState, MAX_PLAY, Play_ID >::next\(\)](#), [playlist](#), [EventRouter::postEvent\(\)](#), [SoundManagerMsg::region](#), [sndlist](#), [SoundManagerMsg::type](#), and [SoundManagerMsg::wakeup](#).

7.148.5.21 void SoundManager::Release ([Snd_ID](#) id)

Marks the sound buffer to be released after the last play command completes (or right now if not being played).

Definition at line 114 of file SoundManager.cc.

References [AutoLock](#), [ListMemBuf< SoundData, MAX_SND, Snd_ID >::erase\(\)](#), [ProcessID::getID\(\)](#), [initRegion\(\)](#), [invalid_Snd_ID](#), [lock](#), [SoundManagerMsg::MSG_SIZE](#), [SoundManagerMsg::region](#), [region](#), [SoundManagerMsg::setDelete\(\)](#), [sndlist](#), [ProcessID::SoundProcess](#), and [subjs](#).

7.148.5.22 void SoundManager::ReleaseFile (const char * name)

Marks the sound buffer to be released after the last play command completes (or right now if not being played).

Definition at line 106 of file SoundManager.cc.

References [AutoLock](#), [ProcessID::getID\(\)](#), [lock](#), [lookup\(\)](#), and [Release\(\)](#).

7.148.5.23 void SoundManager::ResumePlay ([Play_ID](#) id)

Lets you resume playback.

Definition at line 294 of file SoundManager.cc.

References [AutoLock](#), [ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::begin\(\)](#), [chanlist](#), [ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::end\(\)](#), [ProcessID::getID\(\)](#), [invalid_Play_ID](#), [lock](#), [ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::next\(\)](#), and [ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::push_front\(\)](#).

7.148.5.24 **void SoundManager::selectChannels** (std::vector< [Play_ID](#) > & *mix*) [*protected*]

selects which of the channels are actually to be mixed together, depending on queue_ mode

Definition at line 574 of file SoundManager.cc.

References ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::begin(), chanlist, ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::end(), endPlay(), Enqueue, max_chan, ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::next(), Override, Pause, playlist, ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::prev(), queue_mode, sndlist, and Stop.

7.148.5.25 **void SoundManager::SetMode** (unsigned int *max_channels*, [MixMode_t](#) *mixer_mode*, [QueueMode_t](#) *queuing_mode*)

Lets you control the maximum number of channels (currently playing sounds), method for mixing (applies when max_chan>1), and queuing method (for when overflow channels).

Definition at line 305 of file SoundManager.cc.

References AutoLock, ProcessID::getID(), lock, max_chan, mix_mode, and queue_ mode.

7.148.5.26 **void SoundManager::StopPlay** ([Play_ID](#) *id*)

Lets you stop playback of a sound.

Definition at line 262 of file SoundManager.cc.

References EventBase::audioEGID, AutoLock, chanlist, config, EventBase::deactivateETID, ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::end(), ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::erase(), ListMemBuf< PlayState, MAX_PLAY, Play_ID >::erase(), erouter, ProcessID::getID(), invalid_Play_ID, lock, playlist, EventRouter::postEvent(), ListMemBuf< Play_ID, MAX_PLAY, Play_ID >::prev(), Release(), Config::sound_config::sample_bits, Config::sound_config::sample_rate, and Config::sound.

7.148.5.27 **void SoundManager::StopPlay** ()

Lets you stop playback of all sounds.

Definition at line 256 of file SoundManager.cc.

References ListMemBuf< PlayState, MAX_PLAY, Play_ID >::begin(), ListMemBuf< PlayState, MAX_PLAY, Play_ID >::empty(), and playlist.

7.148.5.28 void SoundManager::updateChannels (const std::vector< [Play_ID](#) > & *mixs*, *size_t used*) [protected]

update the offsets of sounds which weren't mixed (when needed depending on queue.-mode)

Definition at line 619 of file SoundManager.cc.

References ListMemBuf< [Play_ID](#), MAX_PLAY, [Play_ID](#) >::begin(), chanlist, ListMemBuf< [Play_ID](#), MAX_PLAY, [Play_ID](#) >::end(), endPlay(), Enqueue, ListMemBuf< [Play_ID](#), MAX_PLAY, [Play_ID](#) >::next(), Override, Pause, playlist, queue.-mode, Snd_ID, sndlist, and Stop.

7.148.6 Member Data Documentation

7.148.6.1 [chanlist_t](#) SoundManager::chanlist [protected]

Holds a list of all currently playing sounds, ordered newest (front) to oldest(back).

Definition at line 206 of file SoundManager.h.

7.148.6.2 const [Play_ID](#) SoundManager::invalid_Play_ID = ([Play_ID](#))-1 [static]

for reporting errors

Definition at line 44 of file SoundManager.h.

7.148.6.3 const [Snd_ID](#) SoundManager::invalid_Snd_ID = ([Snd_ID](#))-1 [static]

for reporting errors

Definition at line 39 of file SoundManager.h.

7.148.6.4 [MutexLock](#)<[ProcessID::NumProcesses](#)> SoundManager::lock [mutable, protected]

Prevents multiple processes from accessing at the same time.

Definition at line 218 of file SoundManager.h.

7.148.6.5 unsigned int SoundManager::max_chan [protected]

Current maximum number of sounds to mix together.

Definition at line 215 of file SoundManager.h.

7.148.6.6 `const unsigned int SoundManager::MAX_NAME_LEN = 64`
[static]

maximum length of a path

Definition at line 47 of file SoundManager.h.

7.148.6.7 `const Play_ID SoundManager::MAX_PLAY = 256` [static]

the number of sounds that can be enqueued at the same time (see MixMode_t)

Definition at line 45 of file SoundManager.h.

7.148.6.8 `const Snd_ID SoundManager::MAX_SND = 50` [static]

the number of sounds that can be loaded at any given time

Definition at line 40 of file SoundManager.h.

7.148.6.9 `MixMode_t SoundManager::mix_mode` [protected]

Current mixing mode, set by `SetMode()`;

Definition at line 209 of file SoundManager.h.

7.148.6.10 `playlist_t SoundManager::playlist` [protected]

Holds a list of all sounds currently enqueued.

Definition at line 202 of file SoundManager.h.

7.148.6.11 `QueueMode_t SoundManager::queue_mode` [protected]

Current queuing mode, set by `SetMode()`;

Definition at line 212 of file SoundManager.h.

7.148.6.12 `sndlist_t SoundManager::sndlist` [protected]

Holds a list of all currently loaded sounds.

Definition at line 189 of file SoundManager.h.

7.148.6.13 OSubject* [SoundManager::subjs](#)[ProcessID::NumProcesses] [protected]

For automatic transmission of shared regions to [SoundPlay](#).

Definition at line 221 of file SoundManager.h.

The documentation for this class was generated from the following files:

- [SoundManager.h](#)
- [SoundManager.cc](#)

7.149 SoundManager::PlayState Struct Reference

```
#include <SoundManager.h>
```

7.149.1 Detailed Description

Holds data about sounds currently being played.

Definition at line 192 of file SoundManager.h.

Public Member Functions

- [PlayState](#) ()
constructor

Public Attributes

- [Snd_ID](#) [snd_id](#)
index of sound
- unsigned int [offset](#)
position in the sound
- unsigned int [cumulative](#)
total time of playing (over queued sounds)
- [Play_ID](#) [next_id](#)
lets you queue for continuous sound, or loop

7.149.2 Constructor & Destructor Documentation

7.149.2.1 SoundManager::PlayState::PlayState ()

constructor

Definition at line 681 of file SoundManager.cc.

References [SoundManager::invalid_Play_ID](#), and [SoundManager::invalid_Snd_ID](#).

7.149.3 Member Data Documentation

7.149.3.1 unsigned int [SoundManager::PlayState::cumulative](#)

total time of playing (over queued sounds)

Definition at line 196 of file SoundManager.h.

7.149.3.2 [Play_ID SoundManager::PlayState::next_id](#)

lets you queue for continuous sound, or loop

Definition at line 197 of file SoundManager.h.

7.149.3.3 unsigned int [SoundManager::PlayState::offset](#)

position in the sound

Definition at line 195 of file SoundManager.h.

7.149.3.4 [Snd_ID SoundManager::PlayState::snd_id](#)

index of sound

Definition at line 194 of file SoundManager.h.

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

- [SoundManager.h](#)
- [SoundManager.cc](#)

7.150 SoundManager::SoundData Struct Reference

```
#include <SoundManager.h>
```

7.150.1 Detailed Description

Holds data about the loaded sounds.

Definition at line 175 of file SoundManager.h.

Public Member Functions

- [SoundData](#) ()
constructor

Public Attributes

- `RCRegion * rcr`
shared region - don't need to share among processes, just collect in [SoundPlay](#)
- `byte * data`
point to data in region (for convenience, only valid in [SoundPlay](#))
- `unsigned int len`
size of the sound
- `unsigned int ref`
reference counter
- `char name [SoundManager::MAX_NAME_LEN]`
stores the path to the file, empty if from a buffer

Private Member Functions

- [SoundData](#) (const [SoundData](#) &)
don't call
- [SoundData](#) operator= (const [SoundData](#) &)
don't call

7.150.2 Constructor & Destructor Documentation

7.150.2.1 SoundManager::SoundData::SoundData ()

constructor

Definition at line 675 of file SoundManager.cc.

References name.

7.150.2.2 SoundManager::SoundData::SoundData (const [SoundData](#) &) [private]

don't call

7.150.3 Member Function Documentation

7.150.3.1 [SoundData](#) SoundManager::SoundData::operator= (const [SoundData](#) &) [private]

don't call

7.150.4 Member Data Documentation

7.150.4.1 byte* [SoundManager::SoundData::data](#)

point to data in region (for convenience, only valid in [SoundPlay](#))

Definition at line 178 of file SoundManager.h.

7.150.4.2 unsigned int [SoundManager::SoundData::len](#)

size of the sound

Definition at line 179 of file SoundManager.h.

7.150.4.3 char [SoundManager::SoundData::name](#)[[SoundManager::MAX_NAME_LEN](#)]

stores the path to the file, empty if from a buffer

Definition at line 181 of file SoundManager.h.

7.150.4.4 RCRegion* [SoundManager::SoundData::rcr](#)

shared region - don't need to share among processes, just collect in [SoundPlay](#)

Definition at line 177 of file SoundManager.h.

7.150.4.5 unsigned int [SoundManager::SoundData::ref](#)

reference counter

Definition at line 180 of file SoundManager.h.

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

- [SoundManager.h](#)
- [SoundManager.cc](#)

7.151 SoundManagerMsg Struct Reference

```
#include <SoundManagerMsg.h>
```

7.151.1 Detailed Description

A small header that preceeds data sent by [SoundManager](#) between processes.

Definition at line 8 of file SoundManagerMsg.h.

Public Types

- typedef unsigned short [Snd_ID](#)
the type to use when referring to Sounds

Public Member Functions

- [SoundManagerMsg](#) ()
constructor
- virtual [~SoundManagerMsg](#) ()
virtual destructor
- [Snd_ID](#) getID () const
Accessor for the id number, set by [SoundManager](#).

Static Public Attributes

- const unsigned int [MSG_SIZE](#) = 16
maintains even word alignment

Private Types

- enum [MsgType](#) { [add](#), [del](#), [wakeup](#), [unknown](#) }
Denotes what type of message this is.

Private Member Functions

- void [setAdd](#) ([Snd_ID](#) sndid)
Sets up the header as an add message.
- void [setDelete](#) ([RCRegion](#) *rcregion)
Sets up the header as an erase message.
- void [setWakeup](#) ()
Sets up the header as a wakeup message.
- [SoundManagerMsg](#) (const [SoundManagerMsg](#) &)
don't call
- [SoundManagerMsg](#) operator= (const [SoundManagerMsg](#) &)
don't call

Private Attributes

- enum [SoundManagerMsg::MsgType](#) type
Denotes what type of message this is.
- [Snd_ID](#) id
The id of the sound this is in reference to.
- [RCRegion](#) * [region](#)
The RCRegion to free, if it's a deletion.

Friends

- class [SoundManager](#)

7.151.2 Member Typedef Documentation

7.151.2.1 typedef unsigned short [SoundManagerMsg::Snd_ID](#)

the type to use when referring to Sounds

Definition at line 10 of file [SoundManagerMsg.h](#).

7.151.3 Member Enumeration Documentation

7.151.3.1 enum [SoundManagerMsg::MsgType](#) [private]

Denotes what type of message this is.

Enumeration values:

- add**
- del**
- wakeup**
- unknown**

Definition at line 28 of file SoundManagerMsg.h.

7.151.4 Constructor & Destructor Documentation

7.151.4.1 [SoundManagerMsg::SoundManagerMsg \(\)](#) [inline]

constructor

Definition at line 15 of file SoundManagerMsg.h.

References [id](#), [region](#), [Snd_ID](#), [type](#), and [unknown](#).

7.151.4.2 [virtual SoundManagerMsg::~~SoundManagerMsg \(\)](#) [inline, virtual]

virtual destructor

doesn't do anything, but don't remove it, otherwise this would no longer be a virtual base class

Definition at line 19 of file SoundManagerMsg.h.

7.151.4.3 [SoundManagerMsg::SoundManagerMsg \(const \[SoundManagerMsg\]\(#\) &\)](#) [private]

don't call

7.151.5 Member Function Documentation

7.151.5.1 [Snd_ID](#) [SoundManagerMsg::getID \(\) const](#) [inline]

Accessor for the id number, set by [SoundManager](#).

Definition at line 22 of file SoundManagerMsg.h.

References id, and Snd_ID.

7.151.5.2 [SoundManagerMsg](#) SoundManagerMsg::operator= (const [SoundManagerMsg](#) &) [private]

don't call

7.151.5.3 void SoundManagerMsg::setAdd ([Snd_ID](#) sndid) [inline, private]

Sets up the header as an add message.

Definition at line 37 of file SoundManagerMsg.h.

References add, id, and type.

7.151.5.4 void SoundManagerMsg::setDelete (RCRegion * rcregion) [inline, private]

Sets up the header as an erase message.

Definition at line 43 of file SoundManagerMsg.h.

References del, region, and type.

7.151.5.5 void SoundManagerMsg::setWakeup () [inline, private]

Sets up the header as a wakeup message.

Definition at line 49 of file SoundManagerMsg.h.

References type, and wakeup.

7.151.6 Friends And Related Function Documentation

7.151.6.1 friend class [SoundManager](#) [friend]

Definition at line 25 of file SoundManagerMsg.h.

7.151.7 Member Data Documentation

7.151.7.1 [Snd_ID](#) [SoundManagerMsg::id](#) [private]

The id of the sound this is in reference to.

Definition at line 31 of file SoundManagerMsg.h.

7.151.7.2 `const unsigned int` [SoundManagerMsg::MSG_SIZE](#) = 16 [static]

maintains even word alignment

Definition at line 12 of file SoundManagerMsg.h.

7.151.7.3 `RCRegion*` [SoundManagerMsg::region](#) [private]

The RCRegion to free, if it's a deletion.

Definition at line 34 of file SoundManagerMsg.h.

7.151.7.4 `enum` [SoundManagerMsg::MsgType](#) [SoundManagerMsg::type](#) [private]

Denotes what type of message this is.

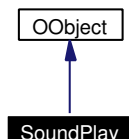
The documentation for this struct was generated from the following file:

- [SoundManagerMsg.h](#)

7.152 SoundPlay Class Reference

```
#include <SoundPlay.h>
```

Inheritance diagram for SoundPlay:



7.152.1 Detailed Description

The process (a.k.a. [OObject](#)), which is responsible for sending sound buffers to the system to play.

This sound process will purposely starve the system of sound buffers when nothing is playing, both to eliminate needless zeroing of entire buffers, as well as to reduce system overhead of playing empty buffers.

If you want to know how to play sounds, you should be looking at SoundManager's documentation.

Basically a slightly modified version of the SoundPlay example code from Sony. Here's their license: Copyright 2002,2003 Sony Corporation

Permission to use, copy, modify, and redistribute this software for non-commercial use is hereby granted.

This software is provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of fitness for a particular purpose.

Definition at line 28 of file SoundPlay.h.

Public Member Functions

- [SoundPlay](#) ()
constructor
- virtual [~SoundPlay](#) ()
destructor
- virtual OStatus [DoInit](#) (const OSystemEvent &event)

called by system when time to do init

- virtual OStatus [DoStart](#) (const OSystemEvent &event)
called by system when time to start running
- virtual OStatus [DoStop](#) (const OSystemEvent &event)
called by system when time to stop running
- virtual OStatus [DoDestroy](#) (const OSystemEvent &event)
called by system when time to free
- void [ReadySendSound](#) (const OReadyEvent &event)
called by system when it's ready for another sound buffer
- void [ReadyRegisterSoundManager](#) (const OReadyEvent &event)
called by system when observers are ready to receive the [SoundManager](#)
- void [GotEventTranslatorQueue](#) (const ONotifyEvent &event)
called by system when the queue for sending events to Main is being published
- void [GotSoundMsg](#) (const ONotifyEvent &event)
called by system when [SoundManager](#) has sent itself a message on a different process (either to add or remove sounds from memory)

Public Attributes

- OSubject * [subject](#) [numOfSubject]
array of subject IDs, used to identify outgoing data
- OObserver * [observer](#) [numOfObserver]
array of observer IDs, used to identify what's ready

Private Member Functions

- void [doSendSound](#) ()
called to send sound buffer(s) to system
- void [OpenSpeaker](#) ()
initializes speaker

- void [NewSoundVectorData](#) ()
sets up sound buffers
- void [SetPowerAndVolume](#) ()
sets volume to max
- RCRegion * [InitRegion](#) (unsigned int size)
inits each buffer
- RCRegion * [FindFreeRegion](#) ()
finds the first sound buffer which system isn't using (buffers are recycled)

Private Attributes

- unsigned int [active](#)
number of active sound channels - if it's 0, we'll purposely starve system of sound buffers
- RCRegion * [soundManagerMemRgn](#)
SoundPlay creates, Main & Motion receive - Shared region used by [SoundManager](#).
- RCRegion * [eventTranslatorQueueMemRgn](#)
Main creates, Motion (& SoundPlay) receive.
- [EventTranslator](#) etrans
will be given all events created by [SoundManager](#) to be forwarded to Main
- OPrimitiveID [speakerID](#)
ID returned to system after opening `SPEAKER_LOCATOR`.
- RCRegion * [region](#) [[SOUND_NUM_BUFFER](#)]
holds references to shared regions holding sound clips

Static Private Attributes

- const char *const [SPEAKER_LOCATOR](#) = "PRM:/r1/c1/c2/c3/s1-Speaker:S1"
ID for speaker in system.
- const size_t [SOUND_NUM_BUFFER](#) = 2

number of buffers to use

7.152.2 Constructor & Destructor Documentation

7.152.2.1 SoundPlay::SoundPlay ()

constructor

Definition at line 26 of file SoundPlay.cc.

References region, and SOUND_NUM_BUFFER.

7.152.2.2 virtual SoundPlay::~SoundPlay () [inline, virtual]

destructor

Definition at line 31 of file SoundPlay.h.

7.152.3 Member Function Documentation

7.152.3.1 OStatus SoundPlay::DoDestroy (const OSystemEvent & event) [virtual]

called by system when time to free

Definition at line 90 of file SoundPlay.cc.

References config, erouter, eventTranslatorQueueMemRgn, Config::sound_-config::preload, SoundManager::ReleaseFile(), sndman, and Config::sound.

7.152.3.2 OStatus SoundPlay::DoInit (const OSystemEvent & event) [virtual]

called by system when time to do init

Definition at line 34 of file SoundPlay.cc.

References config, erouter, SoundManager::InitAccess(), InitRegion(), SoundManager::LoadFile(), SoundManager::MAX_SND, NewSoundVectorData(), observer, OpenSpeaker(), Config::sound_config::preload, ProcessID::setID(), SetPowerAndVolume(), sndman, Config::sound, soundManagerMemRgn, ProcessID::SoundProcess, and subject.

7.152.3.3 void SoundPlay::doSendSound () [private]

called to send sound buffer(s) to system

Definition at line 144 of file SoundPlay.cc.

References [active](#), [SoundManager::CopyTo\(\)](#), [FindFreeRegion\(\)](#), [SoundManager::GetNumPlaying\(\)](#), [sndman](#), [SOUND_NUM_BUFFER](#), and [subject](#).

**7.152.3.4 OStatus SoundPlay::DoStart (const OSystemEvent & event)
[virtual]**

called by system when time to start running

Definition at line 68 of file SoundPlay.cc.

**7.152.3.5 OStatus SoundPlay::DoStop (const OSystemEvent & event)
[virtual]**

called by system when time to stop running

Definition at line 79 of file SoundPlay.cc.

7.152.3.6 RCRegion * SoundPlay::FindFreeRegion () [private]

finds the first sound buffer which system isn't using (buffers are recycled)

Definition at line 244 of file SoundPlay.cc.

References [region](#), and [SOUND_NUM_BUFFER](#).

7.152.3.7 void SoundPlay::GotEventTranslatorQueue (const ONotifyEvent & event)

called by system when the queue for sending events to Main is being published

Definition at line 119 of file SoundPlay.cc.

References [EventRouter::addTrapper\(\)](#), [ASSERT](#), [erouter](#), [etrans](#), [eventTranslatorQueueMemRgn](#), [observer](#), and [EventTranslator::setQueue\(\)](#).

7.152.3.8 void SoundPlay::GotSoundMsg (const ONotifyEvent & event)

called by system when [SoundManager](#) has sent itself a message on a different process (either to add or remove sounds from memory)

Definition at line 131 of file SoundPlay.cc.

References [active](#), [doSendSound\(\)](#), [SoundManager::GetNumPlaying\(\)](#), [observer](#), [SoundManager::ReceivedMsg\(\)](#), and [sndman](#).

7.152.3.9 RCRegion * SoundPlay::InitRegion (unsigned int *size*) [private]

inits each buffer

Will round up size to the nearest page

Definition at line 235 of file [SoundPlay.cc](#).

References [ASSERT](#).

7.152.3.10 void SoundPlay::NewSoundVectorData () [private]

sets up sound buffers

Definition at line 177 of file [SoundPlay.cc](#).

References [config](#), [region](#), [Config::sound_config::sample_bits](#), [Config::sound_config::sample_rate](#), [Config::sound](#), [SOUND_NUM_BUFFER](#), [ERS210Info::Sound-BufferTime](#), and [speakerID](#).

7.152.3.11 void SoundPlay::OpenSpeaker () [private]

initializes speaker

Definition at line 169 of file [SoundPlay.cc](#).

References [SPEAKER_LOCATOR](#), and [speakerID](#).

7.152.3.12 void SoundPlay::ReadyRegisterSoundManager (const OReadyEvent & *event*)

called by system when observers are ready to receive the [SoundManager](#)

Definition at line 108 of file [SoundPlay.cc](#).

References [soundManagerMemRgn](#), and [subject](#).

7.152.3.13 void SoundPlay::ReadySendSound (const OReadyEvent & *event*)

called by system when it's ready for another sound buffer

Definition at line 101 of file [SoundPlay.cc](#).

References [doSendSound\(\)](#).

7.152.3.14 void SoundPlay::SetPowerAndVolume () [private]

sets volume to max

Definition at line 212 of file SoundPlay.cc.

References `config`, `Config::sound_config::sample_bits`, `Config::sound_config::sample_rate`, `Config::sound`, and `speakerID`.

7.152.4 Member Data Documentation**7.152.4.1 unsigned int SoundPlay::active [private]**

number of active sound channels - if it's 0, we'll purposely starve system of sound buffers

Definition at line 58 of file SoundPlay.h.

7.152.4.2 EventTranslator SoundPlay::etrans [private]

will be given all events created by [SoundManager](#) to be forwarded to Main

Definition at line 62 of file SoundPlay.h.

7.152.4.3 RCRegion* SoundPlay::eventTranslatorQueueMemRgn [private]

Main creates, Motion (& SoundPlay) receive.

Definition at line 61 of file SoundPlay.h.

7.152.4.4 OObserver* SoundPlay::observer[numOfObserver]

array of observer IDs, used to identify what's ready

Definition at line 45 of file SoundPlay.h.

7.152.4.5 RCRegion* SoundPlay::region[SOUND_NUM_BUFFER] [private]

holds references to shared regions holding sound clips

Definition at line 65 of file SoundPlay.h.

7.152.4.6 `const size_t SoundPlay::SOUND_NUM_BUFFER = 2` [static, private]

number of buffers to use

Definition at line 56 of file SoundPlay.h.

7.152.4.7 `RCRegion* SoundPlay::soundManagerMemRgn` [private]

SoundPlay creates, Main & Motion receive - Shared region used by [SoundManager](#).

Definition at line 60 of file SoundPlay.h.

7.152.4.8 `const char* const SoundPlay::SPEAKER_LOCATOR = "PRM:/r1/c1/c2/c3/s1-Speaker:S1"` [static, private]

ID for speaker in system.

Definition at line 55 of file SoundPlay.h.

7.152.4.9 `OPrimitiveID SoundPlay::speakerID` [private]

ID returned to system after opening SPEAKER_LOCATOR.

Definition at line 64 of file SoundPlay.h.

7.152.4.10 `OSubject* SoundPlay::subject[numOfSubject]`

array of subject IDs, used to identify outgoing data

Definition at line 44 of file SoundPlay.h.

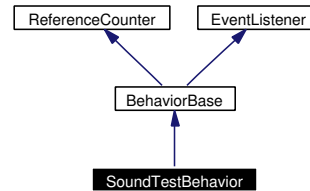
The documentation for this class was generated from the following files:

- [SoundPlay.h](#)
- [SoundPlay.cc](#)

7.153 SoundTestBehavior Class Reference

```
#include <SoundTestBehavior.h>
```

Inheritance diagram for SoundTestBehavior:



7.153.1 Detailed Description

allows you to experiment with playing sounds different ways.

A different sound will be played for each of the buttons, except the head buttons. When the chin button is held down, any sounds (from this behavior) will be queued up and then played successively once the chin button is released.

Notice that this doesn't preload all needed sounds:

- `barkmed.wav` is listed in `/ms/config/tekkotsu.cfg` as a preloaded system sound
- `growl.wav` will be loaded before being played automatically - notice the hiccup this can cause.

Definition at line 18 of file `SoundTestBehavior.h`.

Public Member Functions

- [SoundTestBehavior\(\)](#)
Constructor.
- virtual void [DoStart\(\)](#)
Load some sounds, listen for button events.
- virtual void [DoStop\(\)](#)
Release sounds we loaded in [DoStart\(\)](#).
- virtual void [processEvent](#) (const [EventBase](#) &event)

Play the sound corresponding to the button.

- virtual std::string [getName](#) () const
returns name to system

Static Public Member Functions

- std::string [getClassDescription](#) ()
Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Member Functions

- void [play](#) (const char *name)
called when a button is pressed - checks if it should enqueue or just play

Protected Attributes

- [SoundManager::Play_ID](#) curplay
current chain (may not be valid if chin button not down or time is past [endtime](#))
- unsigned int [endtime](#)
the expected end of play time for the current chain

Event Templates

Used to match against the different buttons that have sounds mapped to them

- [EventBase LFr](#)
- [EventBase RFr](#)
- [EventBase LBk](#)
- [EventBase RBk](#)
- [EventBase Back](#)

Static Protected Attributes

- const bool [pauseWhileChin](#) = true
if this is true, won't start playing chain until you release the chin button

7.153.2 Constructor & Destructor Documentation

7.153.2.1 `SoundTestBehavior::SoundTestBehavior ()` `[inline]`

Constructor.

Definition at line 21 of file `SoundTestBehavior.h`.

References `Back`, `ButtonSourceID::BackButSID`, `curplay`, `endtime`, `LBk`, `ButtonSourceID::LBkPawSID`, `LFr`, `ButtonSourceID::LFrPawSID`, `RBk`, `ButtonSourceID::RBkPawSID`, `RFr`, and `ButtonSourceID::RFrPawSID`.

7.153.3 Member Function Documentation

7.153.3.1 `virtual void SoundTestBehavior::DoStart ()` `[inline, virtual]`

Load some sounds, listen for button events.

Reimplemented from [BehaviorBase](#).

Definition at line 32 of file `SoundTestBehavior.h`.

References `EventRouter::addListener()`, `EventBase::buttonEGID`, `BehaviorBase::DoStart()`, `erouter`, `SoundManager::LoadFile()`, and `sndman`.

7.153.3.2 `virtual void SoundTestBehavior::DoStop ()` `[inline, virtual]`

Release sounds we loaded in [DoStart\(\)](#).

Reimplemented from [BehaviorBase](#).

Definition at line 41 of file `SoundTestBehavior.h`.

References `BehaviorBase::DoStop()`, `erouter`, `SoundManager::ReleaseFile()`, `EventRouter::removeListener()`, and `sndman`.

7.153.3.3 `std::string SoundTestBehavior::getClassDescription ()` `[inline, static]`

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 72 of file `SoundTestBehavior.h`.

7.153.3.4 `virtual std::string SoundTestBehavior::getName () const` [inline, virtual]

returns name to system

Implements [BehaviorBase](#).

Definition at line 71 of file SoundTestBehavior.h.

7.153.3.5 `void SoundTestBehavior::play (const char * name)` [inline, protected]

called when a button is pressed - checks if it should enqueue or just play

Definition at line 75 of file SoundTestBehavior.h.

References [WorldState::buttons](#), [SoundManager::ChainFile\(\)](#), [ERS210Info::ChinBut-Offset](#), [curplay](#), [endtime](#), [get_time\(\)](#), [SoundManager::GetRemainTime\(\)](#), [SoundManager::invalid_Play_ID](#), [SoundManager::PausePlay\(\)](#), [pauseWhileChin](#), [SoundManager::PlayFile\(\)](#), [sndman](#), [ERS210Info::SoundBufferTime](#), and [state](#).

7.153.3.6 `virtual void SoundTestBehavior::processEvent (const EventBase & event)` [inline, virtual]

Play the sound corresponding to the button.

Reimplemented from [BehaviorBase](#).

Definition at line 50 of file SoundTestBehavior.h.

References [EventBase::activateETID](#), [Back](#), [curplay](#), [endtime](#), [EventBase::getSource-ID\(\)](#), [EventBase::getTypeID\(\)](#), [SoundManager::invalid_Play_ID](#), [LBk](#), [LFr](#), [pause-WhileChin](#), [play\(\)](#), [RBk](#), [SoundManager::ResumePlay\(\)](#), [RFr](#), and [sndman](#).

7.153.4 Member Data Documentation

7.153.4.1 [EventBase](#) [SoundTestBehavior::Back](#) [protected]

Definition at line 103 of file SoundTestBehavior.h.

7.153.4.2 [SoundManager::Play_ID](#) [SoundTestBehavior::curplay](#) [protected]

current chain (may not be valid if chin button not down or time is past [endtime](#))

Definition at line 98 of file SoundTestBehavior.h.

7.153.4.3 `unsigned int SoundTestBehavior::endtime` [protected]

the expected end of play time for the current chain

Definition at line 99 of file SoundTestBehavior.h.

7.153.4.4 `EventBase SoundTestBehavior::LBk` [protected]

Definition at line 103 of file SoundTestBehavior.h.

7.153.4.5 `EventBase SoundTestBehavior::LFr` [protected]

Definition at line 103 of file SoundTestBehavior.h.

7.153.4.6 `const bool SoundTestBehavior::pauseWhileChin = true` [static, protected]

if this is true, won't start playing chain until you release the chin button

Definition at line 97 of file SoundTestBehavior.h.

7.153.4.7 `EventBase SoundTestBehavior::RBk` [protected]

Definition at line 103 of file SoundTestBehavior.h.

7.153.4.8 `EventBase SoundTestBehavior::RFr` [protected]

Definition at line 103 of file SoundTestBehavior.h.

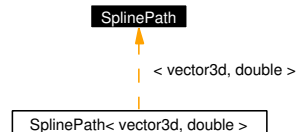
The documentation for this class was generated from the following file:

- [SoundTestBehavior.h](#)

7.154 SplinePath Class Reference

```
#include <Path.h>
```

Inheritance diagram for SplinePath:



Public Member Functions

- [SplinePath](#) ()
- void [init](#) (point x, point [dx](#))
- void [add](#) (point x, point [dx](#), double length, double t)
- point [eval](#) (double t)
- point [eval_deriv](#) (double t)

Private Attributes

- [NonUniformHermiteSplineSegment](#)< point, fnum > [s](#)
- double [t0](#)
- double [t1](#)
- point [x0](#)
- point [x1](#)
- point [dx0](#)
- point [dx1](#)

7.154.1 Constructor & Destructor Documentation

7.154.1.1 SplinePath::SplinePath () [inline]

Definition at line 31 of file Path.h.

7.154.2 Member Function Documentation

7.154.2.1 void `SplinePath::add` (point *x*, point *dx*, double *length*, double *t*)

7.154.2.2 point `SplinePath::eval` (double *t*)

7.154.2.3 point `SplinePath::eval_deriv` (double *t*)

7.154.2.4 void `SplinePath::init` (point *x*, point *dx*)

7.154.3 Member Data Documentation

7.154.3.1 point `SplinePath::dx0` [private]

Definition at line 29 of file Path.h.

7.154.3.2 point `SplinePath::dx1` [private]

Definition at line 29 of file Path.h.

7.154.3.3 `NonUniformHermiteSplineSegment`<point,fnum> `SplinePath::s`
[private]

Definition at line 27 of file Path.h.

7.154.3.4 double `SplinePath::t0` [private]

Definition at line 28 of file Path.h.

7.154.3.5 double `SplinePath::t1` [private]

Definition at line 28 of file Path.h.

7.154.3.6 point `SplinePath::x0` [private]

Definition at line 29 of file Path.h.

7.154.3.7 point `SplinePath::x1` [private]

Definition at line 29 of file Path.h.

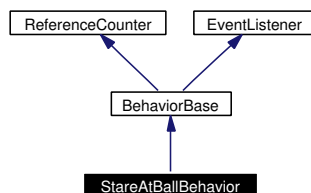
The documentation for this class was generated from the following file:

- [Path.h](#)

7.155 StareAtBallBehavior Class Reference

```
#include <StareAtBallBehavior.h>
```

Inheritance diagram for StareAtBallBehavior:



7.155.1 Detailed Description

A simple behavior to chase after any objects seen by the vision system.

Definition at line 9 of file StareAtBallBehavior.h.

Public Member Functions

- [StareAtBallBehavior](#) ()
constructor
- virtual [~StareAtBallBehavior](#) ()
destructor
- virtual void [DoStart](#) ()
adds a headpointer and a listens for vision events
- virtual void [DoStop](#) ()
removes motion commands and stops listening
- virtual void [processEvent](#) (const [EventBase](#) &event)
sets the head to point at the object and sets the body to move where the head points
- virtual std::string [getName](#) () const
Identifies the behavior in menus and such.

Static Public Member Functions

- `std::string getClassDescription ()`

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- `MotionManager::MC_ID headpointer_id`

a [HeadPointerMC](#) object

7.155.2 Constructor & Destructor Documentation

7.155.2.1 `StareAtBallBehavior::StareAtBallBehavior () [inline]`

constructor

Definition at line 12 of file `StareAtBallBehavior.h`.

References `headpointer_id`.

7.155.2.2 `virtual StareAtBallBehavior::~~StareAtBallBehavior () [inline, virtual]`

destructor

Definition at line 16 of file `StareAtBallBehavior.h`.

7.155.3 Member Function Documentation

7.155.3.1 `void StareAtBallBehavior::DoStart () [virtual]`

adds a headpointer and a listens for vision events

Reimplemented from [BehaviorBase](#).

Definition at line 11 of file `StareAtBallBehavior.cc`.

References `EventRouter::addListener()`, `MotionManager::addMotion()`, `BehaviorBase::DoStart()`, `Vision::enableEvents()`, `erouter`, `headpointer_id`, `motman`, `vision`, and `EventBase::visionEGID`.

7.155.3.2 void StareAtBallBehavior::DoStop () [virtual]

removes motion commands and stops listening

Reimplemented from [BehaviorBase](#).

Definition at line 19 of file StareAtBallBehavior.cc.

References [Vision::disableEvents\(\)](#), [BehaviorBase::DoStop\(\)](#), [erouter](#), [EventManager::forgetListener\(\)](#), [headpointer_id](#), [motman](#), [MotionManager::removeMotion\(\)](#), and [vision](#).

7.155.3.3 std::string StareAtBallBehavior::getClassDescription () [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 28 of file StareAtBallBehavior.h.

7.155.3.4 virtual std::string StareAtBallBehavior::getName () const [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 27 of file StareAtBallBehavior.h.

7.155.3.5 void StareAtBallBehavior::processEvent (const [EventBase](#) & event) [virtual]

sets the head to point at the object and sets the body to move where the head points

Reimplemented from [BehaviorBase](#).

Definition at line 28 of file StareAtBallBehavior.cc.

References [MotionManager::checkinMotion\(\)](#), [MotionManager::checkoutMotion\(\)](#), [DtoR\(\)](#), [EventBase::getGeneratorID\(\)](#), [EventBase::getTypeID\(\)](#), [ERS210Info::HeadOffset](#), [headpointer_id](#), [motman](#), [WorldState::outputs](#), [ERS210Info::PanOffset](#), [HeadPointerMC::setJoints\(\)](#), [state](#), [EventBase::statusETID](#), [ERS210Info::TiltOffset](#), and [EventBase::visionEGID](#).

7.155.4 Member Data Documentation

7.155.4.1 [MotionManager::MC_ID StareAtBallBehavior::headpointer_id](#) [protected]

a [HeadPointerMC](#) object

Definition at line 31 of file StareAtBallBehavior.h.

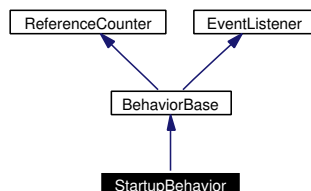
The documentation for this class was generated from the following files:

- [StareAtBallBehavior.h](#)
- [StareAtBallBehavior.cc](#)

7.156 StartupBehavior Class Reference

```
#include <StartupBehavior.h>
```

Inheritance diagram for StartupBehavior:



7.156.1 Detailed Description

This Behavior is the only hardcoded behavior by the framework to start up once all the data structures and such are set up.

Similar in idea to the init process in unix/linux.

Definition at line 13 of file StartupBehavior.h.

[NOHEADER]

- virtual void [DoStart](#) ()

By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.

- virtual void [DoStop](#) ()

By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).

- virtual void [processEvent](#) (const [EventBase](#) &)

- virtual std::string [getName](#) () const

Identifies the behavior in menus and such.

- std::string [getClassDescription](#) ()

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Public Member Functions

- [StartupBehavior](#) ()
Constructor.
- virtual [~StartupBehavior](#) ()
Destructor.

Protected Member Functions

- virtual [ControlBase](#) * [SetupMenus](#) ()
Initializes the [Controller](#) menu structure.

Protected Attributes

- `std::vector< BehaviorBase * >` [spawned](#)
Behaviors spawned from DoStart, so they can automatically be stopped on DoStop.
- [MotionManager::MC_ID](#) [stop_id](#)
the main [EmergencyStopMC](#)
- [MotionManager::MC_ID](#) [pid_id](#)
used to fade in the PIDs up to full strength (from initial zero) This is so the joints don't jerk on startup.

7.156.2 Constructor & Destructor Documentation

7.156.2.1 StartupBehavior::StartupBehavior ()

Constructor.

Definition at line 62 of file StartupBehavior.cc.

References [ReferenceCounter::AddReference\(\)](#).

7.156.2.2 StartupBehavior::~~StartupBehavior () [virtual]

Destructor.

Definition at line 70 of file StartupBehavior.cc.

7.156.3 Member Function Documentation

7.156.3.1 void StartupBehavior::DoStart () [virtual]

By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.

Reimplemented from [BehaviorBase](#).

Definition at line 72 of file StartupBehavior.cc.

References [BatteryCheckControl::activate\(\)](#), [MotionManager::addMotion\(\)](#), [EventRouter::addTimer\(\)](#), [Controller::console_callback\(\)](#), [BatteryCheckControl::deactivate\(\)](#), [Controller::DoStart\(\)](#), [BehaviorBase::DoStart\(\)](#), [erouter](#), [ERS210Info::FrameTime](#), [MotionManager::invalid_MC_ID](#), [MotionManager::k-EmergencyPriority](#), [motman](#), [ERS210Info::NumFrames](#), [pid_id](#), [SoundManager::PlayFile\(\)](#), [Controller::setEStopID\(\)](#), [Wireless::setReceiver\(\)](#), [Controller::setRoot\(\)](#), [SetupMenus\(\)](#), [sndman](#), [sout](#), [spawned](#), [stop_id](#), and [wireless](#).

7.156.3.2 void StartupBehavior::DoStop () [virtual]

By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).

Reimplemented from [BehaviorBase](#).

Definition at line 112 of file StartupBehavior.cc.

References [BehaviorBase::DoStop\(\)](#), [motman](#), [MotionManager::removeMotion\(\)](#), [spawned](#), and [stop_id](#).

7.156.3.3 std::string StartupBehavior::getClassDescription () [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 26 of file StartupBehavior.h.

7.156.3.4 virtual std::string StartupBehavior::getName () const [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 25 of file StartupBehavior.h.

7.156.3.5 void StartupBehavior::processEvent (const [EventBase](#) &) [virtual]

Uses a few timer events at the beginning to fade in the PID values, and closes the mouth too

Reimplemented from [BehaviorBase](#).

Definition at line 120 of file StartupBehavior.cc.

References [erouter](#), [WorldState::ERS210Mask](#), [get_time\(\)](#), [MotionManager::invalid_MC_ID](#), [ERS210Info::MaxRange](#), [motman](#), [ERS210Info::outputRanges](#), [pid_id](#), [MotionManager::removeMotion\(\)](#), [EventRouter::removeTimer\(\)](#), [WorldState::robotDesign](#), [state](#), and [stop_id](#).

7.156.3.6 [ControlBase](#) * StartupBehavior::SetupMenus () [protected, virtual]

Initializes the [Controller](#) menu structure.

Definition at line 148 of file StartupBehavior.cc.

References [FreeMemReportControl::DoStart\(\)](#), [WorldState::ERS220Mask](#), [WorldState::robotDesign](#), [state](#), and [stop_id](#).

7.156.4 Member Data Documentation

7.156.4.1 [MotionManager::MC_ID](#) StartupBehavior::pid_id [protected]

used to fade in the PIDs up to full strength (from initial zero) This is so the joints don't jerk on startup.

Definition at line 34 of file StartupBehavior.h.

7.156.4.2 std::vector<[BehaviorBase](#)*> StartupBehavior::spawned [protected]

Behaviors spawned from DoStart, so they can automatically be stopped on DoStop.

Definition at line 32 of file StartupBehavior.h.

7.156.4.3 [MotionManager::MC_ID](#) StartupBehavior::stop_id [protected]

the main [EmergencyStopMC](#)

Definition at line 33 of file StartupBehavior.h.

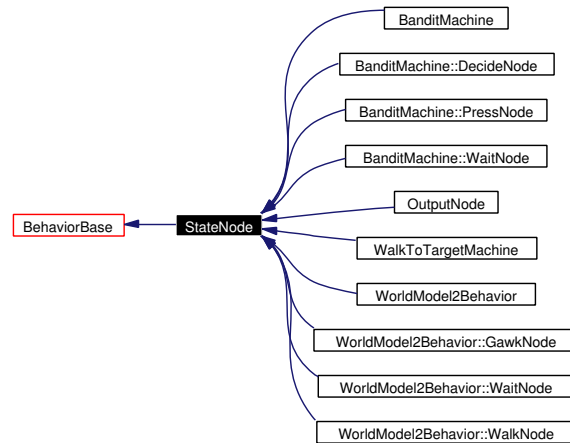
The documentation for this class was generated from the following files:

- [StartupBehavior.h](#)
- [StartupBehavior.cc](#)

7.157 StateNode Class Reference

```
#include <StateNode.h>
```

Inheritance diagram for StateNode:



7.157.1 Detailed Description

Recursive data structure - both a state machine controller as well as a node within a state machine itself.

Override [setup\(\)](#) to setup your own [Transition](#) and StateNode network.

Definition at line 12 of file StateNode.h.

Public Member Functions

- [StateNode](#) ()
constructor
- [StateNode](#) (const char *n, [StateNode](#) *p=NULL)
constructor, pass a name to use, calls setName(n)
- virtual [~StateNode](#) ()
destructor, frees memory used by its outgoing transitions (be careful of incoming ones - they're still around!), and calls RemoveReference() on subnodes
- virtual void [addTransition](#) ([Transition](#) *trans)
Adds the specified StateTransition to the transition table.

- `std::vector< Transition * > & getTransitions ()`
Returns the `std::vector` of transitions so you can modify them yourself if need be.
- `virtual StateNode * addNode (StateNode *node)`
Adds a `StateNode` to [nodes](#) so it can be automatically deleted later, returns what it's passed (for convenience), calls [AddReference\(\)](#) on node.
- `std::vector< StateNode * > & getNodes ()`
Returns the `std::vector` of nodes so you can modify them yourself if need be.
- `void setRetain (bool f)`
Sets the retain flag - if not retained, will [RemoveReference\(\)](#) subnodes upon [DoStop\(\)](#) and recreate them on [DoStart](#) (by calling [setup\(\)](#)) - may be too expensive to be worth saving memory...
- `void DoStart ()`
Transitions should call this when you are entering the state, so it can enable its transitions.
- `virtual void setup ()`
This is called by [DoStart\(\)](#) when you should setup the network.
- `void DoStop ()`
Transitions should call this when you are leaving the state, so it can disable its transitions.
- `void setName (const std::string &n)`
set name to n. Makes a copy of n, so you can throw it away later.
- `virtual std::string getName () const`
returns name of `StateNode`
- `virtual std::string getDescription () const`
returns name again (you should override this for top-level nodes to give users better descriptions)
- `virtual void processEvent (const EventBase &)`
Doesn't do anything, supplied here so you don't have to (since events will probably be going to `Transition`'s, not here).
- `void transitionTo (StateNode *n)`
called by a subnode when it is being [DoStart\(\)](#)'ed

- void `transitionFrom` (`StateNode *n`)
called by a subnode when it is being `DoStop()`'ed

Protected Attributes

- `StateNode * parent`
pointer to the machine that contains this node
- `std::vector< Transition * > transitions`
a vector of outgoing transitions
- bool `issetup`
this is set to true if the network has been setup but not destroyed (i.e. don't need to call `setupSubNodes` again)
- bool `retain`
this is set to true if the network should be retained between activations. Otherwise it's deleted upon `DoStop()`. (or at least `RemoveReference()` is called on subnodes)
- `std::vector< StateNode * > nodes`
vector of StateNodes, just so they can be deleted again on `DoStop()` (unless retained) or `~StateNode()`
- `std::string name`
holds the name of the Node/Machine

Private Member Functions

- `StateNode` (`const StateNode &node`)
don't call this
- `StateNode operator=` (`const StateNode &node`)
don't call this

7.157.2 Constructor & Destructor Documentation

7.157.2.1 `StateNode::StateNode ()` [inline]

constructor

Definition at line 15 of file `StateNode.h`.

References `issetup`, `name`, `nodes`, `parent`, `retain`, and `transitions`.

7.157.2.2 `StateNode::StateNode (const char * n, StateNode * p = NULL)` [inline]

constructor, pass a name to use, calls `setName(n)`

Definition at line 18 of file `StateNode.h`.

References `issetup`, `name`, `nodes`, `parent`, `retain`, `setName()`, and `transitions`.

7.157.2.3 `StateNode::~~StateNode ()` [virtual]

destructor, frees memory used by its outgoing transitions (be careful of incoming ones - they're still around!), and calls [RemoveReference\(\)](#) on subnodes

Definition at line 4 of file `StateNode.cc`.

References `DoStop()`, `BehaviorBase::isActive()`, `nodes`, and `transitions`.

7.157.2.4 `StateNode::StateNode (const StateNode & node)` [private]

don't call this

7.157.3 Member Function Documentation

7.157.3.1 `virtual StateNode* StateNode::addNode (StateNode * node)` [inline, virtual]

Adds a `StateNode` to [nodes](#) so it can be automatically deleted later, returns what it's passed (for convenience), calls [AddReference\(\)](#) on *node*.

Definition at line 32 of file `StateNode.h`.

References `ReferenceCounter::AddReference()`, and `nodes`.

7.157.3.2 void StateNode::addTransition ([Transition](#) * *trans*) [virtual]

Adds the specified StateTransition to the transition table.

Definition at line 13 of file StateNode.cc.

References [Transition::enable\(\)](#), [BehaviorBase::isActive\(\)](#), and [transitions](#).

7.157.3.3 void StateNode::DoStart () [virtual]

Transitions should call this when you are entering the state, so it can enable its transitions.

Reimplemented from [BehaviorBase](#).

Reimplemented in [BanditMachine](#), [BanditMachine::PressNode](#), [BanditMachine::DecideNode](#), [BanditMachine::WaitNode](#), [WalkToTargetMachine](#), [WorldModel2Behavior](#), [WorldModel2Behavior::WalkNode](#), [WorldModel2Behavior::GawkNode](#), [WorldModel2Behavior::WaitNode](#), and [OutputNode](#).

Definition at line 19 of file StateNode.cc.

References [EventBase::activateETID](#), [BehaviorBase::DoStart\(\)](#), [erouter](#), [getName\(\)](#), [issetup](#), [parent](#), [EventRouter::postEvent\(\)](#), [setup\(\)](#), [EventBase::stateMachineEGID](#), [transitions](#), and [transitionTo\(\)](#).

7.157.3.4 void StateNode::DoStop () [virtual]

Transitions should call this when you are leaving the state, so it can disable its transitions.

Reimplemented from [BehaviorBase](#).

Reimplemented in [BanditMachine](#), [BanditMachine::PressNode](#), [BanditMachine::WaitNode](#), [WalkToTargetMachine](#), [WorldModel2Behavior](#), [WorldModel2Behavior::WalkNode](#), [WorldModel2Behavior::GawkNode](#), and [WorldModel2Behavior::WaitNode](#).

Definition at line 32 of file StateNode.cc.

References [EventBase::deactivateETID](#), [BehaviorBase::DoStop\(\)](#), [erouter](#), [getName\(\)](#), [issetup](#), [nodes](#), [EventRouter::postEvent\(\)](#), [retain](#), [EventBase::stateMachineEGID](#), and [transitions](#).

7.157.3.5 virtual std::string StateNode::getDescription () const [inline, virtual]

returns name again (you should override this for top-level nodes to give users better descriptions)

Reimplemented from [BehaviorBase](#).

Definition at line 56 of file StateNode.h.

References name.

7.157.3.6 `virtual std::string StateNode::getName () const` `[inline, virtual]`

returns name of StateNode

Implements [BehaviorBase](#).

Definition at line 53 of file StateNode.h.

References name.

7.157.3.7 `std::vector<StateNode*>& StateNode::getNodes ()` `[inline]`

Returns the std::vector of nodes so you can modify them yourself if need be.

Definition at line 35 of file StateNode.h.

References nodes.

7.157.3.8 `std::vector<Transition*>& StateNode::getTransitions ()` `[inline]`

Returns the std::vector of transitions so you can modify them yourself if need be.

Definition at line 29 of file StateNode.h.

References transitions.

7.157.3.9 `StateNode StateNode::operator= (const StateNode & node)` `[private]`

don't call this

7.157.3.10 `virtual void StateNode::processEvent (const EventBase &)` `[inline, virtual]`

Doesn't do anything, supplied here so you don't have to (since events will probably be going to Transition's, not here).

Reimplemented from [BehaviorBase](#).

Reimplemented in [BanditMachine::WaitNode](#), [WalkToTargetMachine](#), [WorldModel2Behavior::WalkNode](#), and [WorldModel2Behavior::GawkNode](#).

Definition at line 59 of file StateNode.h.

7.157.3.11 void StateNode::setName (const std::string & n)

set name to *n*. Makes a copy of *n*, so you can throw it away later.

Definition at line 46 of file StateNode.cc.

References [name](#).

7.157.3.12 void StateNode::setRetain (bool f) [inline]

Sets the retain flag - if not retained, will [RemoveReference\(\)](#) subnodes upon [DoStop\(\)](#) and recreate them on DoStart (by calling [setup\(\)](#)) - may be too expensive to be worth saving memory...

Definition at line 38 of file StateNode.h.

References [retain](#).

7.157.3.13 virtual void StateNode::setup () [inline, virtual]

This is called by [DoStart\(\)](#) when you should setup the network.

Reimplemented in [BanditMachine](#), [WalkToTargetMachine](#), and [WorldModel2Behavior](#).

Definition at line 44 of file StateNode.h.

References [issetup](#).

7.157.3.14 void StateNode::transitionFrom (StateNode * n)

called by a subnode when it is being [DoStop\(\)](#)'ed

Definition at line 54 of file StateNode.cc.

7.157.3.15 void StateNode::transitionTo (StateNode * n)

called by a subnode when it is being [DoStart\(\)](#)'ed

Definition at line 50 of file StateNode.cc.

7.157.4 Member Data Documentation

7.157.4.1 `bool StateNode::issetup` [protected]

this is set to true if the network has been setup but not destroyed (i.e. don't need to call `setupSubNodes` again)

Definition at line 76 of file `StateNode.h`.

7.157.4.2 `std::string StateNode::name` [protected]

holds the name of the Node/Machine

Definition at line 84 of file `StateNode.h`.

7.157.4.3 `std::vector<StateNode*> StateNode::nodes` [protected]

vector of `StateNodes`, just so they can be deleted again on `DoStop()` (unless retained) or `~StateNode()`

Definition at line 80 of file `StateNode.h`.

7.157.4.4 `StateNode* StateNode::parent` [protected]

pointer to the machine that contains this node

Definition at line 70 of file `StateNode.h`.

7.157.4.5 `bool StateNode::retain` [protected]

this is set to true if the network should be retained between activations. Otherwise it's deleted upon `DoStop()`. (or at least `RemoveReference()` is called on subnodes)

Definition at line 78 of file `StateNode.h`.

7.157.4.6 `std::vector<Transition*> StateNode::transitions` [protected]

a vector of outgoing transitions

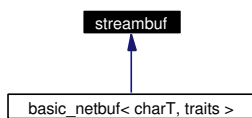
Definition at line 72 of file `StateNode.h`.

The documentation for this class was generated from the following files:

- [StateNode.h](#)
- [StateNode.cc](#)

7.158 streambuf Class Reference

Inheritance diagram for streambuf:



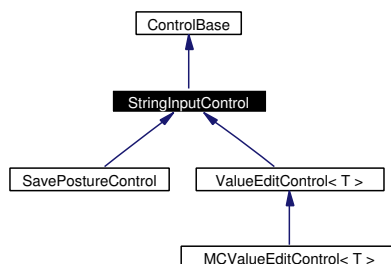
The documentation for this class was generated from the following file:

- [ionetstream.h](#)

7.159 StringInputControl Class Reference

```
#include <StringInputControl.h>
```

Inheritance diagram for StringInputControl:



7.159.1 Detailed Description

Upon activation, prompts the user for a string and stores it.

Definition at line 10 of file StringInputControl.h.

Public Member Functions

- [StringInputControl](#) (const std::string &n, const std::string &prompt)
Constructor.
- [StringInputControl](#) (const std::string &n, const std::string &desc, const std::string &prompt)
Constructor.
- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#) disp_id, [Socket](#) *gui)
Called when the control is activated (or the control system is reactivating).
- virtual void [refresh](#) ()
called when the child has died and this control should refresh its display
- virtual [ControlBase](#) * [doReadStdIn](#) (const std::string &prompt)
prompt the user for text input on the current input device (cin, tekkotsu console (sout), or GUI)
- virtual [ControlBase](#) * [takeInput](#) (const std::string &msg)

called when the user has supplied a text string (may not have been prompted by [do-ReadStdIn\(!\)](#))

- virtual std::string [getLastInput](#) ()
returns last call to [takeInput\(\)](#)
- virtual void [setPrompt](#) (const std::string &prompt)
sets the prompt to give to the user

Protected Attributes

- std::string [lastInput](#)
stores the last input to [takeInput\(\)](#)
- std::string [userPrompt](#)
stores the prompt to send out

7.159.2 Constructor & Destructor Documentation

7.159.2.1 StringInputControl::StringInputControl (const std::string & *n*, const std::string & *prompt*) [inline]

Constructor.

Definition at line 13 of file StringInputControl.h.

References [lastInput](#), and [userPrompt](#).

7.159.2.2 StringInputControl::StringInputControl (const std::string & *n*, const std::string & *desc*, const std::string & *prompt*) [inline]

Constructor.

Definition at line 15 of file StringInputControl.h.

References [lastInput](#), and [userPrompt](#).

7.159.3 Member Function Documentation

7.159.3.1 [ControlBase](#) * [StringInputControl](#)::[activate](#) ([MotionManager](#)::[MC_ID](#) *disp_id*, [Socket](#) * *gui*) [virtual]

Called when the control is activated (or the control system is reactivating).

Takes the id number of a [LedMC](#) which the control should use, maintained by [Controller](#). Controls share the display which is passed, and may use the socket *gui* to communicate with the GUI controller, if it is connected.

Returns:

a [ControlBase](#) pointer. Return:

- *this* if the control should stay active (if it's not a one-shot command)
- `NULL` to return to parent
- other address to spawn a child control

Reimplemented from [ControlBase](#).

Reimplemented in [ValueEditControl< T >](#).

Definition at line 7 of file [StringInputControl.cc](#).

References [ControlBase::display_id](#), [doReadStdIn\(\)](#), [ControlBase::gui_comm](#), and [userPrompt](#).

7.159.3.2 [ControlBase](#) * [StringInputControl](#)::[doReadStdIn](#) (const std::string & *prompt*) [virtual]

prompt the user for text input on the current input device (cin, tekkotsu console (sout), or GUI)

Reimplemented from [ControlBase](#).

Definition at line 17 of file [StringInputControl.cc](#).

References [ControlBase::doReadStdIn\(\)](#), and [userPrompt](#).

7.159.3.3 virtual std::string [StringInputControl](#)::[getLastInput](#) () [inline, virtual]

returns last call to [takeInput\(\)](#)

Definition at line 29 of file [StringInputControl.h](#).

References [lastInput](#).

7.159.3.4 void StringInputControl::refresh () [virtual]

called when the child has died and this control should refresh its display

Reimplemented from [ControlBase](#).

Definition at line 13 of file StringInputControl.cc.

References [ControlBase::doReadStdIn\(\)](#), and [userPrompt](#).

7.159.3.5 virtual void StringInputControl::setPrompt (const std::string & *prompt*) [inline, virtual]

sets the prompt to give to the user

Definition at line 32 of file StringInputControl.h.

References [userPrompt](#).

7.159.3.6 virtual [ControlBase](#)* StringInputControl::takeInput (const std::string & *msg*) [inline, virtual]

called when the user has supplied a text string (may not have been prompted by [doReadStdIn\(\)](#)!)

Reimplemented from [ControlBase](#).

Reimplemented in [SavePostureControl](#), and [ValueEditControl< T >](#).

Definition at line 23 of file StringInputControl.h.

References [lastInput](#).

7.159.4 Member Data Documentation**7.159.4.1 std::string [StringInputControl::lastInput](#) [protected]**

stores the last input to [takeInput\(\)](#)

Definition at line 35 of file StringInputControl.h.

7.159.4.2 std::string [StringInputControl::userPrompt](#) [protected]

stores the prompt to send out

Definition at line 36 of file StringInputControl.h.

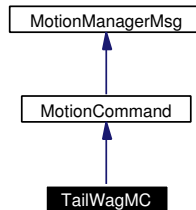
The documentation for this class was generated from the following files:

- [StringInputControl.h](#)
- [StringInputControl.cc](#)

7.160 TailWagMC Class Reference

```
#include <TailWagMC.h>
```

Inheritance diagram for TailWagMC:



7.160.1 Detailed Description

A simple motion command for wagging the tail - you can specify period, magnitude, and tilt.

Definition at line 12 of file TailWagMC.h.

Public Member Functions

- [TailWagMC](#) ()
constructor
- virtual [~TailWagMC](#) ()
destructor
- virtual int [updateOutputs](#) ()
is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)
- virtual int [isDirty](#) ()
not used by [MotionManager](#) at the moment, but could be used to reduce recomputation, and you may find it useful
- virtual int [isAlive](#) ()
used to prune "dead" motions from the [MotionManager](#)
- void [setPeriod](#) (unsigned int p)
sets the period of time between swings, in milliseconds

- unsigned int `getPeriod ()`
gets the period of time between swings, in milliseconds
- void `setMagnitude (double mag)`
sets the magnitude of swings, in radians
- double `getMagnitude ()`
gets the magnitude of swings, in radians
- void `setTilt (double r)`
sets the tilt of the tail while wagging, in radians
- void `unsetTilt ()`
makes the tilt control unspecified, will let something else control tilt
- double `getTilt ()`
sets the tilt of the tail while wagging, in radians
- void `setActive (bool a)`
turns the tail wagger on or off
- bool `getActive ()`
returns true if this is currently trying to wag the tail

Protected Attributes

- unsigned int `period`
period of time between swings, in milliseconds
- double `magnitude`
magnitude of swings, in radians
- bool `active`
true if this is currently trying to wag the tail
- `OutputCmd tilt`
holds current setting for the tilt joint
- `OutputCmd pans [NumFrames]`
holds commands for planning ahead the wagging

7.160.2 Constructor & Destructor Documentation

7.160.2.1 TailWagMC::TailWagMC () [inline]

constructor

Definition at line 15 of file TailWagMC.h.

References active, magnitude, period, and tilt.

7.160.2.2 virtual TailWagMC::~~TailWagMC () [inline, virtual]

destructor

Definition at line 17 of file TailWagMC.h.

7.160.3 Member Function Documentation

7.160.3.1 bool TailWagMC::getActive () [inline]

returns true if this is currently trying to wag the tail

Definition at line 39 of file TailWagMC.h.

References active.

7.160.3.2 double TailWagMC::getMagnitude () [inline]

gets the magnitude of swings, in radians

Definition at line 34 of file TailWagMC.h.

References magnitude.

7.160.3.3 unsigned int TailWagMC::getPeriod () [inline]

gets the period of time between swings, in milliseconds

Definition at line 32 of file TailWagMC.h.

References period.

7.160.3.4 double TailWagMC::getTilt () [inline]

sets the tilt of the tail while wagging, in radians

Definition at line 37 of file TailWagMC.h.

References tilt, and OutputCmd::value.

7.160.3.5 **virtual int TailWagMC::isAlive ()** [inline, virtual]

used to prune "dead" motions from the [MotionManager](#)

note that a motion could be "paused" or inactive and therefore not dirty, but still alive, biding its time to "strike" ;)

Returns:

zero if the motion is still processing, non-zero otherwise

Implements [MotionCommand](#).

Definition at line 29 of file TailWagMC.h.

7.160.3.6 **virtual int TailWagMC::isDirty ()** [inline, virtual]

not used by [MotionManager](#) at the moment, but could be used to reduce recomputation, and you may find it useful

Returns:

zero if none of the commands have changed since last getJointCmd(), else non-zero

Implements [MotionCommand](#).

Definition at line 28 of file TailWagMC.h.

References active.

7.160.3.7 **void TailWagMC::setActive (bool *a*)** [inline]

turns the tail wagger on or off

Definition at line 38 of file TailWagMC.h.

References active.

7.160.3.8 **void TailWagMC::setMagnitude (double *mag*)** [inline]

sets the magnitude of swings, in radians

Definition at line 33 of file TailWagMC.h.

References magnitude.

7.160.3.9 void TailWagMC::setPeriod (unsigned int *p*) [inline]

sets the period of time between swings, in milliseconds

Definition at line 31 of file TailWagMC.h.

References `period`.

7.160.3.10 void TailWagMC::setTilt (double *r*) [inline]

sets the tilt of the tail while wagging, in radians

Definition at line 35 of file TailWagMC.h.

References `OutputCmd::set()`, and `tilt`.

7.160.3.11 void TailWagMC::unsetTilt () [inline]

makes the tilt control unspecified, will let something else control tilt

Definition at line 36 of file TailWagMC.h.

References `tilt`, and `OutputCmd::unset()`.

7.160.3.12 virtual int TailWagMC::updateOutputs () [inline, virtual]

is called once per update cycle, can do any processing you need to change your priorities or set output commands on the [MotionManager](#)

Returns:

zero if no changes were made, non-zero otherwise

See also:

`RobotInfo::NumFrames`

`RobotInfo::FrameTime`

Implements [MotionCommand](#).

Definition at line 18 of file TailWagMC.h.

References `active`, `WorldState::ERS210Mask`, `ERS210Info::FrameTime`, `get_time()`, `magnitude`, `motman`, `ERS210Info::NumFrames`, `ERS210Info::PanOffset`, `pans`, `period`, `WorldState::robotDesign`, `OutputCmd::set()`, `MotionManager::setOutput()`, `state`, `tilt`, and `ERS210Info::TiltOffset`.

7.160.4 Member Data Documentation

7.160.4.1 bool [TailWagMC::active](#) [protected]

true if this is currently trying to wag the tail

Definition at line 44 of file TailWagMC.h.

7.160.4.2 double [TailWagMC::magnitude](#) [protected]

magnitude of swings, in radians

Definition at line 43 of file TailWagMC.h.

7.160.4.3 [OutputCmd TailWagMC::pans\[NumFrames\]](#) [protected]

holds commands for planning ahead the wagging

Definition at line 46 of file TailWagMC.h.

7.160.4.4 unsigned int [TailWagMC::period](#) [protected]

period of time between swings, in milliseconds

Definition at line 42 of file TailWagMC.h.

7.160.4.5 [OutputCmd TailWagMC::tilt](#) [protected]

holds current setting for the tilt joint

Definition at line 45 of file TailWagMC.h.

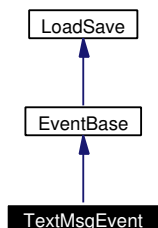
The documentation for this class was generated from the following file:

- [TailWagMC.h](#)

7.161 TextMsgEvent Class Reference

```
#include <TextMsgEvent.h>
```

Inheritance diagram for TextMsgEvent:



7.161.1 Detailed Description

Extends [EventBase](#) to also include actual message text.

Definition at line 8 of file TextMsgEvent.h.

Public Member Functions

- [TextMsgEvent](#) ()
Constructor.
- [TextMsgEvent](#) (const std::string &text)
Constructor, pass a text msg.
- std::string [getText](#) () const
returns the text
- [TextMsgEvent](#) & [setText](#) (const std::string &text)
sets the text
- int [getToken](#) () const
returns the token
- [TextMsgEvent](#) & [setToken](#) (int token)
sets the token
- virtual unsigned int [getBinSize](#) () const

calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.

- virtual unsigned int [LoadBuffer](#) (const char buf[], unsigned int len)
Load from a saved buffer.
- virtual unsigned int [SaveBuffer](#) (char buf[], unsigned int len) const
Save to a given buffer.

Protected Attributes

- std::string [_text](#)
the unmodified arguments passed to the command
- int [_token](#)
for future expansion, to support centralized parsing

7.161.2 Constructor & Destructor Documentation

7.161.2.1 TextMsgEvent::TextMsgEvent () [inline]

Constructor.

Definition at line 11 of file TextMsgEvent.h.

References [_text](#), [_token](#), [EventBase::activateETID](#), and [EventBase::textmsgEGID](#).

7.161.2.2 TextMsgEvent::TextMsgEvent (const std::string & text) [inline]

Constructor, pass a text msg.

Definition at line 14 of file TextMsgEvent.h.

References [_text](#), [_token](#), [EventBase::activateETID](#), and [EventBase::textmsgEGID](#).

7.161.3 Member Function Documentation

7.161.3.1 virtual unsigned int TextMsgEvent::getBinSize () const [inline, virtual]

calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.

Returns:

number of bytes read/written, 0 if error (or empty)

Reimplemented from [EventBase](#).

Definition at line 22 of file TextMsgEvent.h.

References `_text`, `_token`, `LoadSave::creatorSize()`, `EventBase::getBinSize()`, and `LoadSave::stringpad`.

7.161.3.2 `std::string TextMsgEvent::getText () const` [inline]

returns the text

Definition at line 16 of file TextMsgEvent.h.

References `_text`.

7.161.3.3 `int TextMsgEvent::getToken () const` [inline]

returns the token

Definition at line 19 of file TextMsgEvent.h.

References `_token`.

7.161.3.4 `virtual unsigned int TextMsgEvent::LoadBuffer (const char buf[], unsigned int len)` [inline, virtual]

Load from a saved buffer.

Parameters:

buf pointer to the memory where you should begin loading

len length of *buf* available (this isn't all yours, might be more stuff saved after yours)

Returns:

the number of bytes actually used

Reimplemented from [EventBase](#).

Definition at line 30 of file TextMsgEvent.h.

References `_text`, `_token`, `LoadSave::checkCreator()`, `LoadSave::decode()`, and `EventBase::LoadBuffer()`.

7.161.3.5 `virtual unsigned int TextMsgEvent::SaveBuffer (char buf[], unsigned int len) const` `[inline, virtual]`

Save to a given buffer.

Parameters:

buf pointer to the memory where you should begin writing

len length of *buf* available. (this isn't all yours, constrain yourself to what you returned in `getBinSize()`)

Returns:

the number of bytes actually used

Reimplemented from [EventBase](#).

Definition at line 44 of file `TextMsgEvent.h`.

References `_text`, `_token`, `LoadSave::encode()`, `EventBase::SaveBuffer()`, and `LoadSave::saveCreator()`.

7.161.3.6 [TextMsgEvent&](#) `TextMsgEvent::setText (const std::string & text)` `[inline]`

sets the text

Definition at line 17 of file `TextMsgEvent.h`.

References `_text`.

7.161.3.7 [TextMsgEvent&](#) `TextMsgEvent::setToken (int token)` `[inline]`

sets the token

Definition at line 20 of file `TextMsgEvent.h`.

References `_token`.

7.161.4 Member Data Documentation

7.161.4.1 `std::string TextMsgEvent::_text` `[protected]`

the unmodified arguments passed to the command

Definition at line 59 of file `TextMsgEvent.h`.

7.161.4.2 `int` [TextMsgEvent::_token](#) [protected]

for future expansion, to support centralized parsing

Definition at line 60 of file TextMsgEvent.h.

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

- [TextMsgEvent.h](#)

7.162 TimeET Class Reference

```
#include <TimeET.h>
```

7.162.1 Detailed Description

a nice class for handling time values with high precision

Definition at line 23 of file TimeET.h.

Public Member Functions

- [TimeET Age](#) () const
returns the difference between the current time and the time stored
- [TimeET](#) ()
constructor
- [TimeET](#) (long ms)
constructor
- [TimeET](#) (long sec, long usec)
constructor
- [TimeET](#) (double t)
constructor, sepecify t seconds
- double [Value](#) () const
returns the time stored as seconds in a double
- void [Set](#) (long ms)
sets the time stored in the class
- void [Set](#) (long sec, long usec)
sets the time stored in the class
- void [Set](#) (double t)
sets the time stored in the class

- void [Set](#) ()
sets the time to the current time
- bool [operator<](#) (long ms)
for comparing times
- bool [operator<](#) (double t)
for comparing times
- bool [operator<](#) (const [TimeET](#) &t)
for comparing times
- [TimeET operator+](#) (const [TimeET](#) &t) const
for doing math with time
- [TimeET operator+=](#) (const [TimeET](#) &t)
for doing math with time
- [TimeET operator-](#) (const [TimeET](#) &t) const
for doing math with time
- [TimeET operator-=](#) (const [TimeET](#) &t)
for doing math with time

Static Public Attributes

- const long [us_per_sec](#) = 1000000
conversion factor for microseconds to seconds
- const long [ms_per_sec](#) = 1000
conversion factor for milliseconds to seconds
- const long [us_per_ms](#) = 1000
conversion factor for microseconds to milliseconds

Protected Attributes

- [timeval tv](#)
stores the time

Static Protected Attributes

- [timezone tz](#)
stores the timezone (not really used)

Friends

- `std::ostream & operator<< (std::ostream &o, const TimeET &t)`
lets the class be displayed easily

7.162.2 Constructor & Destructor Documentation

7.162.2.1 [TimeET::TimeET \(\)](#) [[inline](#)]

constructor

Definition at line 29 of file [TimeET.h](#).

References [Set\(\)](#), and [tv](#).

7.162.2.2 [TimeET::TimeET \(long ms\)](#) [[inline](#)]

constructor

Definition at line 32 of file [TimeET.h](#).

References [Set\(\)](#), and [tv](#).

7.162.2.3 [TimeET::TimeET \(long sec, long usec\)](#) [[inline](#)]

constructor

Definition at line 35 of file [TimeET.h](#).

References [Set\(\)](#), and [tv](#).

7.162.2.4 [TimeET::TimeET \(double t\)](#) [[inline](#)]

constructor, sepecify *t* seconds

Definition at line 39 of file [TimeET.h](#).

References [Set\(\)](#), and [tv](#).

7.162.3 Member Function Documentation

7.162.3.1 **TimeET** TimeET::Age () const [inline]

returns the difference between the current time and the time stored

Definition at line 45 of file TimeET.h.

References TimeET().

7.162.3.2 **TimeET** TimeET::operator+ (const **TimeET** & *t*) const [inline]

for doing doing math with time

Definition at line 93 of file TimeET.h.

References TimeET(), tv, timeval::tv_sec, timeval::tv_usec, and us_per_sec.

7.162.3.3 **TimeET** TimeET::operator+= (const **TimeET** & *t*) [inline]

for doing doing math with time

Definition at line 99 of file TimeET.h.

References tv, timeval::tv_sec, timeval::tv_usec, and us_per_sec.

7.162.3.4 **TimeET** TimeET::operator- (const **TimeET** & *t*) const [inline]

for doing doing math with time

Definition at line 105 of file TimeET.h.

References TimeET(), tv, timeval::tv_sec, timeval::tv_usec, and us_per_sec.

7.162.3.5 **TimeET** TimeET::operator-= (const **TimeET** & *t*) [inline]

for doing doing math with time

Definition at line 115 of file TimeET.h.

References tv, timeval::tv_sec, timeval::tv_usec, and us_per_sec.

7.162.3.6 **bool** TimeET::operator< (const **TimeET** & *t*) [inline]

for comparing times

Definition at line 86 of file TimeET.h.

References tv, timeval::tv_sec, and timeval::tv_usec.

7.162.3.7 bool TimeET::operator< (double t) [inline]

for comparing times

Definition at line 83 of file TimeET.h.

References Value().

7.162.3.8 bool TimeET::operator< (long ms) [inline]

for comparing times

Definition at line 79 of file TimeET.h.

References ms_per_sec, tv, timeval::tv_sec, timeval::tv_usec, and us_per_ms.

7.162.3.9 void TimeET::Set () [inline]

sets the time to the current time

Todo

not getting timeofday on OPEN-R, is time since boot instead...

Definition at line 66 of file TimeET.h.

References tv, and tz.

7.162.3.10 void TimeET::Set (double t) [inline]

sets the time stored in the class

Definition at line 60 of file TimeET.h.

References tv, timeval::tv_sec, timeval::tv_usec, and us_per_sec.

7.162.3.11 void TimeET::Set (long sec, long usec) [inline]

sets the time stored in the class

Definition at line 56 of file TimeET.h.

References tv, timeval::tv_sec, timeval::tv_usec, and us_per_sec.

7.162.3.12 void TimeET::Set (long *ms*) [inline]

sets the time stored in the class

Definition at line 53 of file TimeET.h.

References Set(), and us_per_ms.

7.162.3.13 double TimeET::Value () const [inline]

returns the time stored as seconds in a double

Definition at line 48 of file TimeET.h.

References tv, timeval::tv_sec, timeval::tv_usec, and us_per_sec.

7.162.4 Friends And Related Function Documentation**7.162.4.1 std::ostream& operator<< (std::ostream & *o*, const TimeET & *t*) [friend]**

lets the class be displayed easily

Definition at line 142 of file TimeET.h.

7.162.5 Member Data Documentation**7.162.5.1 const long TimeET::ms_per_sec = 1000 [static]**

conversion factor for milliseconds to seconds

Definition at line 128 of file TimeET.h.

7.162.5.2 timeval TimeET::tv [protected]

stores the time

Definition at line 131 of file TimeET.h.

7.162.5.3 struct timezone TimeET::tz [static, protected]

stores the timezone (not really used)

Definition at line 132 of file TimeET.h.

7.162.5.4 `const long TimeET::us_per_ms = 1000` [static]

conversion factor for microseconds to milliseconds

Definition at line 129 of file TimeET.h.

7.162.5.5 `const long TimeET::us_per_sec = 1000000` [static]

conversion factor for microseconds to seconds

Definition at line 127 of file TimeET.h.

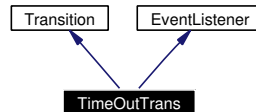
The documentation for this class was generated from the following file:

- [TimeET.h](#)

7.163 TimeOutTrans Class Reference

```
#include <TimeOutTrans.h>
```

Inheritance diagram for TimeOutTrans:



7.163.1 Detailed Description

causes a transition after a specified amount of time has passed

Definition at line 9 of file TimeOutTrans.h.

Public Member Functions

- [TimeOutTrans](#) ([StateNode](#) *source, [StateNode](#) *destination, unsigned int delay)
constructor, specify delay in milliseconds
- virtual void [enable](#) ()
starts timer
- virtual void [disable](#) ()
stops timer
- void [resetTimer](#) ()
resets timer
- virtual void [processEvent](#) (const [EventBase](#) &)
if we receive the timer event, [activate\(\)](#)

Protected Attributes

- unsigned int [d](#)
amount to delay (in milliseconds) before transition

7.163.2 Constructor & Destructor Documentation

7.163.2.1 `TimeoutTrans::TimeoutTrans (StateNode * source, StateNode * destination, unsigned int delay)` [inline]

constructor, specify delay in milliseconds

Definition at line 12 of file TimeoutTrans.h.

References `d`.

7.163.3 Member Function Documentation

7.163.3.1 `virtual void TimeoutTrans::disable ()` [inline, virtual]

stops timer

Implements [Transition](#).

Definition at line 18 of file TimeoutTrans.h.

References `erouter`, and `EventRouter::forgetListener()`.

7.163.3.2 `virtual void TimeoutTrans::enable ()` [inline, virtual]

starts timer

Implements [Transition](#).

Definition at line 15 of file TimeoutTrans.h.

References `resetTimer()`.

7.163.3.3 `virtual void TimeoutTrans::processEvent (const EventBase &)` [inline, virtual]

if we receive the timer event, [activate\(\)](#)

Implements [EventListener](#).

Definition at line 24 of file TimeoutTrans.h.

References `Transition::activate()`.

7.163.3.4 `void TimeoutTrans::resetTimer ()` [inline]

resets timer

Definition at line 21 of file TimeoutTrans.h.

References EventRouter::addTimer(), d, and erouter.

7.163.4 Member Data Documentation

7.163.4.1 unsigned int [TimeOutTrans::d](#) [protected]

amount to delay (in milliseconds) before transition

Definition at line 29 of file TimeOutTrans.h.

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

- [TimeOutTrans.h](#)

7.164 timeval Struct Reference

```
#include <TimeET.h>
```

7.164.1 Detailed Description

would be defined by system - we redefine the same structure in case we're compiling for Aperios

Definition at line 10 of file TimeET.h.

Public Attributes

- unsigned int [tv_sec](#)
seconds
- long [tv_usec](#)
microseconds

7.164.2 Member Data Documentation

7.164.2.1 unsigned int [timeval::tv_sec](#)

seconds

Definition at line 11 of file TimeET.h.

7.164.2.2 long [timeval::tv_usec](#)

microseconds

Definition at line 12 of file TimeET.h.

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

- [TimeET.h](#)

7.165 timezone Struct Reference

```
#include <TimeET.h>
```

7.165.1 Detailed Description

would be defined by system - we redefine the same structure in case we're compiling for Aperios

Definition at line 15 of file TimeET.h.

Public Attributes

- int [tz_minuteswest](#)
minutes west of Greenwich
- int [tz_dsttime](#)
type of dst correction

7.165.2 Member Data Documentation

7.165.2.1 int [timezone::tz_dsttime](#)

type of dst correction

Definition at line 17 of file TimeET.h.

7.165.2.2 int [timezone::tz_minuteswest](#)

minutes west of Greenwich

Definition at line 16 of file TimeET.h.

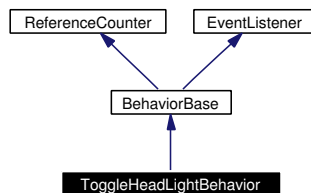
The documentation for this struct was generated from the following file:

- [TimeET.h](#)

7.166 ToggleHeadLightBehavior Class Reference

```
#include <ToggleHeadLightBehavior.h>
```

Inheritance diagram for ToggleHeadLightBehavior:



7.166.1 Detailed Description

opens or closes the head light on an ERS-220

Definition at line 10 of file ToggleHeadLightBehavior.h.

Public Member Functions

- [ToggleHeadLightBehavior](#) ()
constructor
- virtual void [DoStart](#) ()
opens the head light
- virtual void [DoStop](#) ()
resets the head light
- std::string [getName](#) () const
Identifies the behavior in menus and such.

Static Public Member Functions

- std::string [getClassDescription](#) ()
Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- [MotionManager::MC_ID](#) light_id

7.166.2 Constructor & Destructor Documentation

7.166.2.1 ToggleHeadLightBehavior::ToggleHeadLightBehavior () [inline]

constructor

Definition at line 13 of file ToggleHeadLightBehavior.h.

References [light_id](#).

7.166.3 Member Function Documentation

7.166.3.1 virtual void ToggleHeadLightBehavior::DoStart () [inline, virtual]

opens the head light

Reimplemented from [BehaviorBase](#).

Definition at line 16 of file ToggleHeadLightBehavior.h.

References [MotionManager::addMotion\(\)](#), [BehaviorBase::DoStart\(\)](#), [WorldState::ERS220Mask](#), [light_id](#), [motman](#), [WorldState::robotDesign](#), and [state](#).

7.166.3.2 virtual void ToggleHeadLightBehavior::DoStop () [inline, virtual]

resets the head light

Reimplemented from [BehaviorBase](#).

Definition at line 26 of file ToggleHeadLightBehavior.h.

References [light_id](#), [motman](#), and [MotionManager::removeMotion\(\)](#).

7.166.3.3 std::string ToggleHeadLightBehavior::getClassDescription () [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 30 of file ToggleHeadLightBehavior.h.

7.166.3.4 `std::string ToggleHeadLightBehavior::getName () const` `[inline, virtual]`

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 32 of file ToggleHeadLightBehavior.h.

7.166.4 Member Data Documentation

7.166.4.1 `MotionManager::MC_ID ToggleHeadLightBehavior::light_id`
`[protected]`

Definition at line 35 of file ToggleHeadLightBehavior.h.

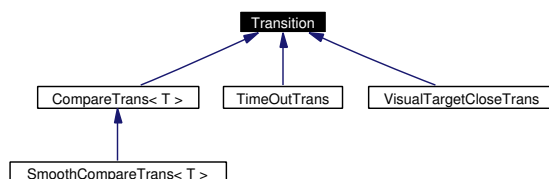
The documentation for this class was generated from the following file:

- [ToggleHeadLightBehavior.h](#)

7.167 Transition Class Reference

```
#include <Transition.h>
```

Inheritance diagram for Transition:



7.167.1 Detailed Description

Represents a transition between StateNodes.

This is an abstract class - you'll want to subclass it to put conditions on the transitions

Definition at line 10 of file Transition.h.

Public Member Functions

- [Transition](#) ([StateNode](#) *source, [StateNode](#) *destination)
constructor, specifies source and destination StateNode's
- [Transition](#) (const [Transition](#) &t)
copy constructor, just in case you need it
- virtual [~Transition](#) ()
destructor
- virtual void [enable](#) ()=0
called by [StateNode](#) when it becomes active - use this to request events (or whatever you need to do)
- virtual void [disable](#) ()=0
called by [StateNode](#) when it becomes inactive - undo whatever you did in Enable()
- virtual void [activate](#) ()
call this when the transition should be made, base class version simply calls StateNode::Leave() on [src](#) and StateNode::Enter() on [dst](#), but you can override.

- [Transition](#) & `operator=` (const [Transition](#) &t)
assignment operator (only does shallow copy)

Protected Attributes

- [StateNode](#) * `src`
the node being transitioned from
- [StateNode](#) * `dst`
the node being transitioned to

7.167.2 Constructor & Destructor Documentation

7.167.2.1 [Transition::Transition](#) ([StateNode](#) * *source*, [StateNode](#) * *destination*) [inline]

constructor, specifies source and destination [StateNode](#)'s

Definition at line 13 of file [Transition.h](#).

References [dst](#), and [src](#).

7.167.2.2 [Transition::Transition](#) (const [Transition](#) & *t*) [inline]

copy constructor, just in case you need it

Definition at line 15 of file [Transition.h](#).

References [dst](#), and [src](#).

7.167.2.3 [virtual Transition::~~Transition](#) () [inline, virtual]

destructor

Definition at line 17 of file [Transition.h](#).

7.167.3 Member Function Documentation

7.167.3.1 [void Transition::activate](#) () [virtual]

call this when the transition should be made, base class version simply calls [StateNode::Leave\(\)](#) on [src](#) and [StateNode::Enter\(\)](#) on [dst](#), but you can override.

Definition at line 4 of file Transition.cc.

References [StateNode::DoStart\(\)](#), [StateNode::DoStop\(\)](#), [dst](#), and [src](#).

7.167.3.2 **virtual void Transition::disable ()** [pure virtual]

called by [StateNode](#) when it becomes inactive - undo whatever you did in [Enable\(\)](#)

Implemented in [CompareTrans< T >](#), [TimeOutTrans](#), and [VisualTargetCloseTrans](#).

7.167.3.3 **virtual void Transition::enable ()** [pure virtual]

called by [StateNode](#) when it becomes active - use this to request events (or whatever you need to do)

Implemented in [CompareTrans< T >](#), [TimeOutTrans](#), and [VisualTargetCloseTrans](#).

7.167.3.4 **Transition& Transition::operator= (const Transition & t)** [inline]

assignment operator (only does shallow copy)

Definition at line 28 of file Transition.h.

References [dst](#), and [src](#).

7.167.4 Member Data Documentation

7.167.4.1 **StateNode* Transition::dst** [protected]

the node being transitioned to

Definition at line 31 of file Transition.h.

7.167.4.2 **StateNode* Transition::src** [protected]

the node being transitioned from

Definition at line 30 of file Transition.h.

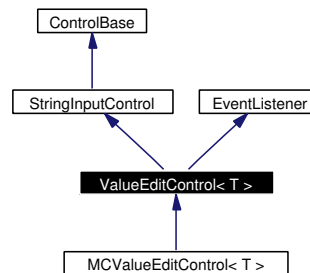
The documentation for this class was generated from the following files:

- [Transition.h](#)
- [Transition.cc](#)

7.168 ValueEditControl< T > Class Template Reference

```
#include <ValueEditControl.h>
```

Inheritance diagram for ValueEditControl< T >:



7.168.1 Detailed Description

```
template<class T> class ValueEditControl< T >
```

allows real-time modification of a value through a pointer

Todo

needs some work to really be useful again

Definition at line 16 of file ValueEditControl.h.

Public Member Functions

- [ValueEditControl](#) (const std::string &n, T *t)
constructor
- [ValueEditControl](#) (const std::string &n, const std::string &p, T *t)
constructor
- [ValueEditControl](#) (const std::string &n, const std::string &d, const std::string &p, T *t)
constructor
- [ValueEditControl](#) (const [ValueEditControl](#)< T > &vec)

copy constructor

- [ValueEditControl](#) [operator=](#) (const [ValueEditControl](#)< T > &vec)
assignment operator
- virtual [~ValueEditControl](#) ()
destructor
- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#) display, [Socket](#) *gui)
reads in current value from target
- virtual void [processEvent](#) (const [EventBase](#) &e)
will increment/decrement the current and then assign it to the target when head buttons pressed
- virtual void [pause](#) ()
request to continue receiving events so we can modify the value while running
- virtual [ControlBase](#) * [doSelect](#) ()
if the value of the [target](#)!=[cur](#), assigns the current value to the target and all the [copies](#)
- virtual [ControlBase](#) * [doNextItem](#) ()
adds one to the current value
- virtual [ControlBase](#) * [doPrevItem](#) ()
subtracts one from the current value
- virtual [ControlBase](#) * [takeInput](#) (const std::string &str)
called when the user has supplied a text string (may not have been prompted by [doReadStdIn\(\)](#))
- virtual std::string [getName](#) () const
shows current value

Target

accessors for the target pointer

- virtual T * [getTarget](#) () const
returns the target pointer
- virtual [ValueEditControl](#) & [setTarget](#) (T *t)
sets the target pointer - the object pointed to will be overwritten on [activate\(\)](#)
**this*

Copies

accessors for the copies vector, so you can assign the same value to several places if you need to

- virtual `std::vector< T * > & getCopies ()`
returns a reference to the vector `copies`
- virtual `ValueEditControl & addCopy (T *t)`
#copies.push_back(t)

Protected Attributes

- `T * target`
the main target
- `T cur`
the value to use when set
- `std::vector< T * > copies`
additional targets

7.168.2 Constructor & Destructor Documentation

7.168.2.1 `template<class T> ValueEditControl< T >::ValueEditControl (const std::string & n, T * t) [inline]`

constructor

Definition at line 19 of file ValueEditControl.h.

References ValueEditControl< T >::copies, ValueEditControl< T >::cur, and ValueEditControl< T >::target.

7.168.2.2 `template<class T> ValueEditControl< T >::ValueEditControl (const std::string & n, const std::string & p, T * t) [inline]`

constructor

Definition at line 21 of file ValueEditControl.h.

References ValueEditControl< T >::copies, ValueEditControl< T >::cur, and ValueEditControl< T >::target.

7.168.2.3 `template<class T> ValueEditControl< T >::ValueEditControl (const std::string & n, const std::string & d, const std::string & p, T * t) [inline]`

constructor

Definition at line 23 of file ValueEditControl.h.

References ValueEditControl< T >::copies, ValueEditControl< T >::cur, and ValueEditControl< T >::target.

7.168.2.4 `template<class T> ValueEditControl< T >::ValueEditControl (const ValueEditControl< T > & vec) [inline]`

copy constructor

Definition at line 25 of file ValueEditControl.h.

References ValueEditControl< T >::copies, ValueEditControl< T >::cur, and ValueEditControl< T >::target.

7.168.2.5 `template<class T> virtual ValueEditControl< T >::~~ValueEditControl () [inline, virtual]`

destructor

Definition at line 29 of file ValueEditControl.h.

7.168.3 Member Function Documentation

7.168.3.1 `template<class T> virtual ControlBase* ValueEditControl< T >::activate (MotionManager::MC_ID display, Socket * gui) [inline, virtual]`

reads in current value from target

Reimplemented from [StringInputControl](#).

Definition at line 32 of file ValueEditControl.h.

References [StringInputControl::activate\(\)](#), ValueEditControl< T >::cur, [erouter](#), [EventRouter::forgetListener\(\)](#), and ValueEditControl< T >::target.

7.168.3.2 `template<class T> virtual ValueEditControl& ValueEditControl< T >::addCopy (T * t) [inline, virtual]`

#copies.push_back(t)

Definition at line 105 of file ValueEditControl.h.

References ValueEditControl< T >::copies.

7.168.3.3 `template<class T> virtual ControlBase* ValueEditControl< T >::doNextItem () [inline, virtual]`

adds one to the current value

Reimplemented from [ControlBase](#).

Definition at line 79 of file ValueEditControl.h.

References ValueEditControl< T >::cur, and StringInputControl::refresh().

7.168.3.4 `template<class T> virtual ControlBase* ValueEditControl< T >::doPrevItem () [inline, virtual]`

subtracts one from the current value

Reimplemented from [ControlBase](#).

Definition at line 85 of file ValueEditControl.h.

References ValueEditControl< T >::cur, and StringInputControl::refresh().

7.168.3.5 `template<class T> virtual ControlBase* ValueEditControl< T >::doSelect () [inline, virtual]`

if the value of the [target](#)!=[cur](#), assigns the current value to the target and all the [copies](#)

Reimplemented from [ControlBase](#).

Reimplemented in [MCValueEditControl](#)< T >.

Definition at line 65 of file ValueEditControl.h.

References ValueEditControl< T >::copies, ValueEditControl< T >::cur, ValueEditControl< T >::getName(), and ValueEditControl< T >::target.

7.168.3.6 `template<class T> virtual std::vector<T*>& ValueEditControl< T >::getCopies () [inline, virtual]`

returns a reference to the vector [copies](#)

Definition at line 104 of file ValueEditControl.h.

References ValueEditControl< T >::copies.

7.168.3.7 `template<class T> virtual std::string ValueEditControl< T >::getName () const [inline, virtual]`

shows current value

Reimplemented from [ControlBase](#).

Definition at line 109 of file ValueEditControl.h.

References [ControlBase::getName\(\)](#), and [ValueEditControl< T >::target](#).

7.168.3.8 `template<class T> virtual T* ValueEditControl< T >::getTarget () const [inline, virtual]`

returns the target pointer

Definition at line 98 of file ValueEditControl.h.

References [ValueEditControl< T >::target](#).

7.168.3.9 `template<class T> ValueEditControl ValueEditControl< T >::operator= (const ValueEditControl< T > & vec) [inline]`

assignment operator

Definition at line 27 of file ValueEditControl.h.

References [ValueEditControl< T >::copies](#), [ValueEditControl< T >::cur](#), [ControlBase::operator=\(\)](#), and [ValueEditControl< T >::target](#).

7.168.3.10 `template<class T> virtual void ValueEditControl< T >::pause () [inline, virtual]`

request to continue receiving events so we can modify the value while running

Reimplemented from [ControlBase](#).

Definition at line 57 of file ValueEditControl.h.

References [EventRouter::addListener\(\)](#), [EventBase::buttonEGID](#), [EventBase::deactivateETID](#), [erouter](#), and [ControlBase::pause\(\)](#).

7.168.3.11 `template<class T> virtual void ValueEditControl< T >::processEvent (const EventBase & e) [inline, virtual]`

will increment/decrement the current and then assign it to the target when head buttons pressed

Implements [EventListener](#).

Definition at line 38 of file ValueEditControl.h.

References ValueEditControl< T >::doNextItem(), ValueEditControl< T >::doSelect(), and EventBase::getSourceID().

7.168.3.12 `template<class T> virtual ValueEditControl& ValueEditControl< T >::setTarget (T *t) [inline, virtual]`

sets the target pointer - the object pointed to will be overwritten on [activate\(\)](#) *this

Definition at line 99 of file ValueEditControl.h.

References ValueEditControl< T >::target.

7.168.3.13 `template<class T> virtual ControlBase* ValueEditControl< T >::takeInput (const std::string &str) [inline, virtual]`

called when the user has supplied a text string (may not have been prompted by [do-ReadStdIn\(\)](#)!)

Reimplemented from [StringInputControl](#).

Definition at line 91 of file ValueEditControl.h.

References ValueEditControl< T >::cur, and ValueEditControl< T >::doSelect().

7.168.4 Member Data Documentation

7.168.4.1 `template<class T> std::vector<T*> ValueEditControl< T >::copies [protected]`

additional targets

Definition at line 118 of file ValueEditControl.h.

7.168.4.2 `template<class T> T ValueEditControl< T >::cur [protected]`

the value to use when set

Definition at line 117 of file ValueEditControl.h.

7.168.4.3 `template<class T> T* ValueEditControl< T >::target [protected]`

the main target

Definition at line 116 of file ValueEditControl.h.

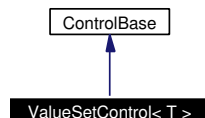
The documentation for this class was generated from the following file:

- [ValueEditControl.h](#)

7.169 ValueSetControl< T > Class Template Reference

```
#include <ValueSetControl.h>
```

Inheritance diagram for ValueSetControl< T >:



7.169.1 Detailed Description

```
template<class T> class ValueSetControl< T >
```

Upon activation, this control will set the target pointer to the specified value.

Definition at line 9 of file ValueSetControl.h.

Public Member Functions

- virtual [ControlBase](#) * [activate](#) ([MotionManager::MC_ID](#) display)
assigns *def* to object pointed to by *target*

Constructors/Destructors

- [ValueSetControl](#) ()
constructor
- [ValueSetControl](#) (const std::string &n, T *t)
constructor
- [ValueSetControl](#) (const std::string &n, T *t, const T &d)
constructor
- [ValueSetControl](#) (const [ValueSetControl](#) &vsc)
copy constructor
- [ValueSetControl](#) operator= (const [ValueSetControl](#) &vsc)
assignment operator
- virtual [~ValueSetControl](#) ()

destructor

Target

accessors for the target pointer

- virtual T * [getTarget](#) () const
returns the target pointer
- virtual [ValueSetControl](#) & [setTarget](#) (T *t)
sets the target pointer - the object pointed to will be overwritten on [activate\(\)](#)
*this

Value

accessors for the default value assigned when activated

- virtual T & [getDefault](#) ()
gets reference to default value
- virtual const T & [getDefault](#) () const
gets reference to default value
- virtual [ValueSetControl](#) & [setDefault](#) (const T &d)
*assigns d to the default value (not to the target, yet) *this*

Protected Attributes

- T * [target](#)
the target that will be set to the default value ([def](#))
- T [def](#)
the value that will be assigned to [target](#) upon a call to [activate\(\)](#)

7.169.2 Constructor & Destructor Documentation

7.169.2.1 `template<class T> ValueSetControl< T >::ValueSetControl ()` [inline]

constructor

Definition at line 13 of file ValueSetControl.h.

References [ValueSetControl](#)< T >::target.

7.169.2.2 `template<class T> ValueSetControl< T >::ValueSetControl (const std::string & n, T * t) [inline]`

constructor

Definition at line 14 of file ValueSetControl.h.

References `ValueSetControl< T >::target`.

7.169.2.3 `template<class T> ValueSetControl< T >::ValueSetControl (const std::string & n, T * t, const T & d) [inline]`

constructor

Definition at line 15 of file ValueSetControl.h.

References `ValueSetControl< T >::def`, and `ValueSetControl< T >::target`.

7.169.2.4 `template<class T> ValueSetControl< T >::ValueSetControl (const ValueSetControl< T > & vsc) [inline]`

copy constructor

Definition at line 16 of file ValueSetControl.h.

References `ValueSetControl< T >::def`, and `ValueSetControl< T >::target`.

7.169.2.5 `template<class T> virtual ValueSetControl< T >::~~ValueSetControl () [inline, virtual]`

destructor

Definition at line 18 of file ValueSetControl.h.

7.169.3 Member Function Documentation

7.169.3.1 `template<class T> virtual ControlBase* ValueSetControl< T >::activate (MotionManager::MC_ID display) [inline, virtual]`

assigns `def` to object pointed to by `target`

Todo

make the leds flash

Definition at line 22 of file ValueSetControl.h.

References ValueSetControl< T >::def, MotionManager::invalid_MC_ID, and ValueSetControl< T >::target.

7.169.3.2 `template<class T> virtual const T& ValueSetControl< T >::getDefault () const [inline, virtual]`

gets reference to default value

Definition at line 39 of file ValueSetControl.h.

References ValueSetControl< T >::def.

7.169.3.3 `template<class T> virtual T& ValueSetControl< T >::getDefault () [inline, virtual]`

gets reference to default value

Definition at line 38 of file ValueSetControl.h.

References ValueSetControl< T >::def.

7.169.3.4 `template<class T> virtual T* ValueSetControl< T >::getTarget () const [inline, virtual]`

returns the target pointer

Definition at line 32 of file ValueSetControl.h.

References ValueSetControl< T >::target.

7.169.3.5 `template<class T> ValueSetControl ValueSetControl< T >::operator= (const ValueSetControl< T > & vsc) [inline]`

assignment operator

Definition at line 17 of file ValueSetControl.h.

References ValueSetControl< T >::def, ControlBase::operator=(), and ValueSetControl< T >::target.

7.169.3.6 `template<class T> virtual ValueSetControl& ValueSetControl< T >::setDefault (const T & d) [inline, virtual]`

assigns d to the default value (not to the target, yet) *this

Definition at line 40 of file ValueSetControl.h.

References `ValueSetControl< T >::def`.

7.169.3.7 `template<class T> virtual ValueSetControl& ValueSetControl< T >::setTarget (T *t) [inline, virtual]`

sets the target pointer - the object pointed to will be overwritten on [activate\(\)](#) *this

Definition at line 33 of file `ValueSetControl.h`.

References `ValueSetControl< T >::target`.

7.169.4 Member Data Documentation

7.169.4.1 `template<class T> T ValueSetControl< T >::def [protected]`

the value that will be assigned to [target](#) upon a call to [activate\(\)](#)

Definition at line 45 of file `ValueSetControl.h`.

7.169.4.2 `template<class T> T* ValueSetControl< T >::target [protected]`

the target that will be set to the default value ([def](#))

Definition at line 44 of file `ValueSetControl.h`.

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

- [ValueSetControl.h](#)

7.170 GVector::vector2d< num > Class Template Reference

```
#include <gvector.h>
```

```
template<class num> class GVector::vector2d< num >
```

Public Member Functions

- [vector2d](#) ()
- [vector2d](#) (num nx, num ny)
- void [set](#) (num nx, num ny)
- void [set](#) ([vector2d](#)< num > p)
- [vector2d](#)< num > & [operator=](#) ([vector2d](#)< num > p)
- num [length](#) () const
- num [sqlength](#) () const
- num [angle](#) () const
- [vector2d](#)< num > [norm](#) () const
- void [normalize](#) ()
- num [dot](#) (const [vector2d](#)< num > p) const
- [vector2d](#)< num > [cross](#) (const [vector2d](#)< num > p) const
- [vector2d](#)< num > [operator+=](#) (const [vector2d](#)< num > p)
- [vector2d](#)< num > [operator-=](#) (const [vector2d](#)< num > p)
- [vector2d](#)< num > [operator *=](#) (const [vector2d](#)< num > p)
- [vector2d](#)< num > [operator/=](#) (const [vector2d](#)< num > p)
- [vector2d](#)< num > [operator+](#) (const [vector2d](#)< num > p) const
- [vector2d](#)< num > [operator-](#) (const [vector2d](#)< num > p) const
- [vector2d](#)< num > [operator *](#) (const [vector2d](#)< num > p) const
- [vector2d](#)< num > [operator/](#) (const [vector2d](#)< num > p) const
- [vector2d](#)< num > [operator *](#) (const num f) const
- [vector2d](#)< num > [operator/](#) (const num f) const
- [vector2d](#)< num > [operator *=](#) (num f)
- [vector2d](#)< num > [operator/=](#) (num f)
- [vector2d](#)< num > [operator-](#) () const
- bool [operator==](#) (const [vector2d](#)< num > p) const
- bool [operator!=](#) (const [vector2d](#)< num > p) const
- bool [operator<](#) (const [vector2d](#)< num > p) const
- bool [operator>](#) (const [vector2d](#)< num > p) const
- bool [operator<=](#) (const [vector2d](#)< num > p) const
- bool [operator>=](#) (const [vector2d](#)< num > p) const
- [vector2d](#)< num > [rotate](#) (const num a) const

Public Attributes

- num [x](#)
- num [y](#)

7.170.1 Constructor & Destructor Documentation

7.170.1.1 `template<class num> GVector::vector2d< num >::vector2d ()`
`[inline]`

Definition at line 355 of file `gvector.h`.

References `GVector::vector2d< num >::x`, and `GVector::vector2d< num >::y`.

7.170.1.2 `template<class num> GVector::vector2d< num >::vector2d (num`
`nx, num ny) [inline]`

Definition at line 357 of file `gvector.h`.

References `GVector::vector2d< num >::x`, and `GVector::vector2d< num >::y`.

7.170.2 Member Function Documentation

7.170.2.1 `template<class num> num GVector::vector2d< num >::angle ()`
`const [inline]`

Definition at line 369 of file `gvector.h`.

References `atan2a()`, `GVector::vector2d< num >::x`, and `GVector::vector2d< num >::y`.

7.170.2.2 `template<class num> vector2d<num> GVector::vector2d< num`
`>::cross (const vector2d< num > p) const`

7.170.2.3 `template<class num> num GVector::vector2d< num >::dot (const`
`vector2d< num > p) const`

Definition at line 441 of file `gvector.h`.

References `GVector::vector2d< num >::x`, and `GVector::vector2d< num >::y`.

7.170.2.4 `template<class num> num GVector::vector2d< num >::length ()
const`

Definition at line 406 of file gvector.h.

References GVector::vector2d< num >::x, and GVector::vector2d< num >::y.

7.170.2.5 `template<class num> vector2d< num > GVector::vector2d< num
>::norm () const`

Definition at line 418 of file gvector.h.

References GVector::vector2d< num >::x, and GVector::vector2d< num >::y.

7.170.2.6 `template<class num> void GVector::vector2d< num >::normalize ()`

Definition at line 431 of file gvector.h.

References GVector::vector2d< num >::x, and GVector::vector2d< num >::y.

7.170.2.7 `template<class num> vector2d<num> GVector::vector2d< num
>::operator * (const num f) const`**7.170.2.8** `template<class num> vector2d<num> GVector::vector2d< num
>::operator * (const vector2d< num > p) const`**7.170.2.9** `template<class num> vector2d<num> GVector::vector2d< num
>::operator *= (num f)`**7.170.2.10** `template<class num> vector2d<num> GVector::vector2d< num
>::operator *= (const vector2d< num > p)`**7.170.2.11** `template<class num> bool GVector::vector2d< num >::operator!=
(const vector2d< num > p) const`**7.170.2.12** `template<class num> vector2d<num> GVector::vector2d< num
>::operator+ (const vector2d< num > p) const`**7.170.2.13** `template<class num> vector2d<num> GVector::vector2d< num
>::operator+= (const vector2d< num > p)`**7.170.2.14** `template<class num> combine &&&vector2d< num >
GVector::vector2d< num >::operator- () const`

Definition at line 538 of file gvector.h.

References `GVector::vector2d< num >::x`, and `GVector::vector2d< num >::y`.

7.170.2.15 `template<class num> vector2d<num> GVector::vector2d< num >::operator- (const vector2d< num > p) const`

7.170.2.16 `template<class num> vector2d<num> GVector::vector2d< num >::operator-= (const vector2d< num > p)`

7.170.2.17 `template<class num> vector2d<num> GVector::vector2d< num >::operator/ (const num f) const`

7.170.2.18 `template<class num> vector2d<num> GVector::vector2d< num >::operator/ (const vector2d< num > p) const`

7.170.2.19 `template<class num> vector2d<num> GVector::vector2d< num >::operator/= (num f)`

7.170.2.20 `template<class num> vector2d<num> GVector::vector2d< num >::operator/= (const vector2d< num > p)`

7.170.2.21 `template<class num> bool GVector::vector2d< num >::operator< (const vector2d< num > p) const`

7.170.2.22 `template<class num> bool GVector::vector2d< num >::operator<= (const vector2d< num > p) const`

7.170.2.23 `template<class num> vector2d<num> & GVector::vector2d< num >::operator= (vector2d< num > p) [inline]`

Definition at line 364 of file `gvector.h`.

References `GVector::vector2d< num >::set()`.

7.170.2.24 `template<class num> bool GVector::vector2d< num >::operator==(const vector2d< num > p) const`

7.170.2.25 `template<class num> bool GVector::vector2d< num >::operator>(const vector2d< num > p) const`

7.170.2.26 `template<class num> bool GVector::vector2d< num >::operator>=(const vector2d< num > p) const`

7.170.2.27 `template<class num> vector2d< num > GVector::vector2d< num >::rotate (const num a) const`

Definition at line 547 of file gvector.h.

References GVector::vector2d< num >::x, and GVector::vector2d< num >::y.

7.170.2.28 `template<class num> void GVector::vector2d< num >::set (vector2d< num > p) [inline]`

Definition at line 362 of file gvector.h.

References GVector::vector2d< num >::x, and GVector::vector2d< num >::y.

7.170.2.29 `template<class num> void GVector::vector2d< num >::set (num nx, num ny) [inline]`

Definition at line 360 of file gvector.h.

References GVector::vector2d< num >::x, and GVector::vector2d< num >::y.

7.170.2.30 `template<class num> num GVector::vector2d< num >::slength () const`

Definition at line 412 of file gvector.h.

References GVector::vector2d< num >::x, and GVector::vector2d< num >::y.

7.170.3 Member Data Documentation

7.170.3.1 `template<class num> num GVector::vector2d< num >::x`

Definition at line 353 of file gvector.h.

7.170.3.2 template<class num> num [GVector::vector2d](#)< num >::y

Definition at line 353 of file gvector.h.

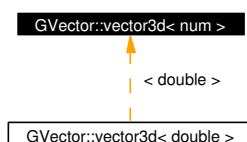
The documentation for this class was generated from the following file:

- [gvector.h](#)

7.171 GVector::vector3d< num > Class Template Reference

```
#include <gvector.h>
```

Inheritance diagram for GVector::vector3d< num >:



```
template<class num> class GVector::vector3d< num >
```

Public Member Functions

- [vector3d](#) ()
- [vector3d](#) (num nx, num ny, num nz)
- void [set](#) (num nx, num ny, num nz)
- void [set](#) ([vector3d](#)< num > p)
- [vector3d](#)< num > & [operator=](#) (const [vector3d](#)< num > p)
- num [length](#) () const
- num [sqlength](#) () const
- [vector3d](#)< num > [norm](#) () const
- void [normalize](#) ()
- num [dot](#) (const [vector3d](#)< num > p) const
- [vector3d](#)< num > [cross](#) (const [vector3d](#)< num > p) const
- [vector3d](#)< num > [operator+=](#) (const [vector3d](#)< num > p)
- [vector3d](#)< num > [operator-=](#) (const [vector3d](#)< num > p)
- [vector3d](#)< num > [operator *=](#) (const [vector3d](#)< num > p)
- [vector3d](#)< num > [operator/=](#) (const [vector3d](#)< num > p)
- [vector3d](#)< num > [operator+](#) (const [vector3d](#)< num > p) const
- [vector3d](#)< num > [operator-](#) (const [vector3d](#)< num > p) const
- [vector3d](#)< num > [operator *](#) (const [vector3d](#)< num > p) const
- [vector3d](#)< num > [operator/](#) (const [vector3d](#)< num > p) const
- [vector3d](#)< num > [operator *](#) (num f) const
- [vector3d](#)< num > [operator/](#) (num f) const
- [vector3d](#)< num > [operator *=](#) (num f)
- [vector3d](#)< num > [operator/=](#) (num f)
- [vector3d](#)< num > [operator-](#) () const

- bool `operator==` (const `vector3d`< num > p) const
- bool `operator!=` (const `vector3d`< num > p) const
- bool `operator<` (const `vector3d`< num > p) const
- bool `operator>` (const `vector3d`< num > p) const
- bool `operator<=` (const `vector3d`< num > p) const
- bool `operator>=` (const `vector3d`< num > p) const
- `vector3d`< num > `rotate_x` (const double a) const
- `vector3d`< num > `rotate_y` (const double a) const
- `vector3d`< num > `rotate_z` (const double a) const

Public Attributes

- num `x`
- num `y`
- num `z`

7.171.1 Constructor & Destructor Documentation

7.171.1.1 `template<class num> GVector::vector3d< num >::vector3d ()`
[inline]

Definition at line 45 of file gvector.h.

7.171.1.2 `template<class num> GVector::vector3d< num >::vector3d (num
nx, num ny, num nz) [inline]`

Definition at line 47 of file gvector.h.

7.171.2 Member Function Documentation

7.171.2.1 `template<class num> vector3d< num > GVector::vector3d< num
>::cross (const vector3d< num > p) const`

Definition at line 145 of file gvector.h.

References `GVector::vector3d< num >::x`, `GVector::vector3d< double >::x`, `GVector::vector3d< double >::y`, `GVector::vector3d< num >::y`, `GVector::vector3d< num >::z`, and `GVector::vector3d< double >::z`.

7.171.2.2 `template<class num> num GVector::vector3d< num >::dot (const
vector3d< num > p) const`

Definition at line 133 of file gvector.h.

References GVector::vector3d< double >::x, GVector::vector3d< num >::x, GVector::vector3d< double >::y, GVector::vector3d< num >::y, GVector::vector3d< double >::z, and GVector::vector3d< num >::z.

7.171.2.3 `template<class num> num GVector::vector3d< num >::length ()
const`

Definition at line 96 of file gvector.h.

References GVector::vector3d< num >::x, GVector::vector3d< num >::y, and GVector::vector3d< num >::z.

7.171.2.4 `template<class num> vector3d< num > GVector::vector3d< num
>::norm () const`

Definition at line 108 of file gvector.h.

References GVector::vector3d< double >::x, GVector::vector3d< num >::x, GVector::vector3d< double >::y, GVector::vector3d< num >::y, GVector::vector3d< double >::z, and GVector::vector3d< num >::z.

7.171.2.5 `template<class num> void GVector::vector3d< num >::normalize ()`

Definition at line 122 of file gvector.h.

References GVector::vector3d< num >::x, GVector::vector3d< num >::y, and GVector::vector3d< num >::z.

- 7.171.2.6 `template<class num> vector3d<num> GVector::vector3d< num >::operator * (num f) const`
- 7.171.2.7 `template<class num> vector3d<num> GVector::vector3d< num >::operator * (const vector3d< num > p) const`
- 7.171.2.8 `template<class num> vector3d<num> GVector::vector3d< num >::operator *= (num f)`
- 7.171.2.9 `template<class num> vector3d<num> GVector::vector3d< num >::operator *= (const vector3d< num > p)`
- 7.171.2.10 `template<class num> bool GVector::vector3d< num >::operator!= (const vector3d< num > p) const`
- 7.171.2.11 `template<class num> vector3d<num> GVector::vector3d< num >::operator+ (const vector3d< num > p) const`
- 7.171.2.12 `template<class num> vector3d<num> GVector::vector3d< num >::operator+= (const vector3d< num > p)`
- 7.171.2.13 `template<class num> combine &&&vector3d< num > GVector::vector3d< num >::operator- () const`

Definition at line 246 of file `gvector.h`.

References `GVector::vector3d< num >::x`, `GVector::vector3d< double >::x`, `GVector::vector3d< num >::y`, `GVector::vector3d< double >::y`, `GVector::vector3d< num >::z`, and `GVector::vector3d< double >::z`.

- 7.171.2.14 `template<class num> vector3d<num> GVector::vector3d< num >::operator- (const vector3d< num > p) const`
- 7.171.2.15 `template<class num> vector3d<num> GVector::vector3d< num >::operator-= (const vector3d< num > p)`
- 7.171.2.16 `template<class num> vector3d<num> GVector::vector3d< num >::operator/ (num f) const`
- 7.171.2.17 `template<class num> vector3d<num> GVector::vector3d< num >::operator/ (const vector3d< num > p) const`
- 7.171.2.18 `template<class num> vector3d<num> GVector::vector3d< num >::operator/= (num f)`
- 7.171.2.19 `template<class num> vector3d<num> GVector::vector3d< num >::operator/= (const vector3d< num > p)`
- 7.171.2.20 `template<class num> bool GVector::vector3d< num >::operator< (const vector3d< num > p) const`
- 7.171.2.21 `template<class num> bool GVector::vector3d< num >::operator<= (const vector3d< num > p) const`
- 7.171.2.22 `template<class num> vector3d<num>& GVector::vector3d< num >::operator= (const vector3d< num > p) [inline]`

Definition at line 55 of file `gvector.h`.

- 7.171.2.23 `template<class num> bool GVector::vector3d< num >::operator== (const vector3d< num > p) const`
- 7.171.2.24 `template<class num> bool GVector::vector3d< num >::operator> (const vector3d< num > p) const`
- 7.171.2.25 `template<class num> bool GVector::vector3d< num >::operator>= (const vector3d< num > p) const`
- 7.171.2.26 `template<class num> vector3d< num > GVector::vector3d< num >::rotate_x (const double a) const`

Definition at line 257 of file `gvector.h`.

References `GVector::vector3d< num >::x`, `GVector::vector3d< double >::x`, `GVector::vector3d< num >::y`, `GVector::vector3d< double >::y`, `GVector::vector3d<`

double >::z, and `GVector::vector3d< num >::z`.

7.171.2.27 `template<class num> vector3d< num > GVector::vector3d< num >::rotate_y (const double a) const`

Definition at line 274 of file `gvector.h`.

References `GVector::vector3d< num >::x`, `GVector::vector3d< double >::x`, `GVector::vector3d< num >::y`, `GVector::vector3d< double >::y`, `GVector::vector3d< double >::z`, and `GVector::vector3d< num >::z`.

7.171.2.28 `template<class num> vector3d< num > GVector::vector3d< num >::rotate_z (const double a) const`

Definition at line 291 of file `gvector.h`.

References `GVector::vector3d< num >::x`, `GVector::vector3d< double >::x`, `GVector::vector3d< double >::y`, `GVector::vector3d< num >::y`, `GVector::vector3d< num >::z`, and `GVector::vector3d< double >::z`.

7.171.2.29 `template<class num> void GVector::vector3d< num >::set (vector3d< num > p) [inline]`

Definition at line 52 of file `gvector.h`.

7.171.2.30 `template<class num> void GVector::vector3d< num >::set (num nx, num ny, num nz) [inline]`

Definition at line 50 of file `gvector.h`.

7.171.2.31 `template<class num> num GVector::vector3d< num >::sqlength () const`

Definition at line 102 of file `gvector.h`.

References `GVector::vector3d< num >::x`, `GVector::vector3d< num >::y`, and `GVector::vector3d< num >::z`.

7.171.3 Member Data Documentation

7.171.3.1 `template<class num> num GVector::vector3d< num >::x`

Definition at line 43 of file `gvector.h`.

7.171.3.2 template<class num> num [GVector::vector3d](#)< num >::y

Definition at line 43 of file gvector.h.

7.171.3.3 template<class num> num [GVector::vector3d](#)< num >::z

Definition at line 43 of file gvector.h.

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

- [gvector.h](#)

7.172 Vision Class Reference

```
#include <Vision.h>
```

Public Member Functions

- [Vision](#) ()
- [~Vision](#) ()
- int [setThreshold](#) (int threshold_id)
- const [vector3d](#) & [get_camera_loc](#) ()
- const [vector3d](#) & [get_camera_dir](#) ()
- int [getColor](#) (int x, int y)
- int [getWidth](#) ()
- int [getHeight](#) ()
- void [initialize](#) ()
- void [setCameraParam](#) ()
- void [initializeEventSpecs](#) ()
- void [enableEvents](#) (int vevent)
- void [enableEvents](#) (int vevent, int noise)
- void [disableEvents](#) (int vevent)
- void [setNoiseThreshold](#) (int vevent, int noise)
- bool [close](#) ()
- bool [processFrame](#) (const uchar *data_y, const uchar *data_u, const uchar *data_v, int [width](#), int [height](#))
- bool [saveThresholdImage](#) (char *filename)
- void [sendRawImage](#) ()
- void [sendRLEImage](#) ()
- void [sendColorArea](#) ()

Public Attributes

- unsigned long [frameTimestamp](#)
- int [frame_count](#)
- int [num_tmaps](#)
- int [cur_tmap](#)
- [cmap_t](#) * [cmap](#)
- [cmap_t](#) * [tmap](#) [MAX_TMAPS]
- [run](#) * [rmap](#)
- [run](#) * [rmap2](#)
- [region](#) * [reg](#)
- int [yindex](#) [144]
- [VisionObjectInfo](#) [vobj_info](#) [NUM_VISION_OBJECTS]

- [VisionEventSpec vevent_spec](#) [NUM_VEVENTS]
- [color_class_state color](#) [MAX_COLORS]
- int [width](#)
- int [height](#)
- int [max_width](#)
- int [max_height](#)
- int [max_runs](#)
- int [max_regions](#)
- int [num_colors](#)
- int [num_runs](#)
- int [num_regions](#)
- double [body_angle](#)
- double [body_height](#)
- double [head_angles](#) [3]
- [vector3d camera_loc](#)
- [vector3d camera_dir](#)
- [vector3d camera_up](#)
- [vector3d camera_right](#)
- double [sq_distort_coeff](#)
- double [lin_distort_coeff](#)
- bool [calcTotalArea](#)
- [Marker markers](#) [3]
- int [vis_markers](#)
- [ObjectInfo * obj_info](#)

Private Member Functions

- bool [thresholdImage](#) (CMVision::image_idx< rgb > &[img](#))
- bool [thresholdImage](#) (CMVision::image_yuv< const uchar > &[img](#))
- template<class image> bool [runLowLevelVision](#) (image &[img](#))
- int [getColorUnsafe](#) (int x, int y)
- int [getNearColor](#) (int x, int y)
- int [addToHistHorizStrip](#) (int y, int x1, int x2, int *color_cnt)
- int [addToHistVertStrip](#) (int x, int y1, int y2, int *color_cnt)
- void [createEvent](#) (unsigned int tid, unsigned int sid, float cenX, float cenY)
- [vector3d getPixelDirection](#) (double x, double y)
- void [findSpan](#) (double &left, double &right, double x1, double x2, double y1, double y2)
- int [calcEdgeMask](#) (double x1, double x2, double y1, double y2)
- int [calcEdgeMask](#) (int x1, int x2, int y1, int y2)
- int [calcEdgeMask](#) (region *tmpreg)
- int [isIn](#) (region *r1, region *r2)

- int [isAdjacent](#) ([region](#) *r1, [region](#) *r2)
- bool [findHand](#) (VObject *hand, [VisionObjectInfo](#) *hand_info)
- bool [findBall](#) (int ball_color, VObject *ball, [VisionObjectInfo](#) *ball_info)
- bool [findThing](#) (VObject *thing, [VisionObjectInfo](#) *thing_info)
- bool [findMarkers](#) ()
- bool [findGesture](#) ([VisionObjectInfo](#) *hand_info)
- int [identifyMarker](#) (int color1, int color2, int color3)
- bool [generateEvent](#) (int vevent, double conf, int cenX, int cenY)
- bool [runHighLevelVision](#) (ObjectInfo *obj_info)
- [Vision](#) (const [Vision](#) &)
don't copy
- [Vision](#) operator= (const [Vision](#) &)
don't assign

Private Attributes

- int [outCountAvgColor](#)
- int [outCountColorArea](#)
- int [outCountRaw](#)
- int [outCountRLE](#)
- [VisionSerializer](#) * vser
- CMVision::image_yuv< const uchar > [img](#)

Friends

- class [VisionSerializer](#)

7.172.1 Constructor & Destructor Documentation

7.172.1.1 [Vision::Vision](#) ()

Definition at line 23 of file [Vision.cc](#).

References [initialize\(\)](#), [setCameraParam\(\)](#), [VisionSerializer](#), and [vser](#).

7.172.1.2 [Vision::~~Vision](#) () [inline]

Definition at line 101 of file [Vision.h](#).

7.172.1.3 Vision::Vision (const Vision &) [private]

don't copy

7.172.2 Member Function Documentation**7.172.2.1 int Vision::addToHistHorizStrip (int y, int x1, int x2, int * color_cnt) [private]**

Definition at line 314 of file Vision.cc.

References bound(), getColorUnsafe(), height, and width.

7.172.2.2 int Vision::addToHistVertStrip (int x, int y1, int y2, int * color_cnt) [private]

Definition at line 329 of file Vision.cc.

References bound(), getColorUnsafe(), height, and width.

7.172.2.3 int Vision::calcEdgeMask (region * tmpreg) [inline, private]

Definition at line 199 of file Vision.h.

References calcEdgeMask().

7.172.2.4 int Vision::calcEdgeMask (int x1, int x2, int y1, int y2) [private]

Definition at line 299 of file Vision.cc.

References height, VisionInterface::OFF_EDGE_BOTTOM, VisionInterface::OFF_EDGE_LEFT, VisionInterface::OFF_EDGE_RIGHT, VisionInterface::OFF_EDGE_TOP, and width.

7.172.2.5 int Vision::calcEdgeMask (double x1, double x2, double y1, double y2) [private]**7.172.2.6 bool Vision::close ()**

Definition at line 267 of file Vision.cc.

References cmap, max_height, max_width, num_tmaps, reg, rmap, and tmap.

7.172.2.7 void Vision::createEvent (unsigned int *tid*, unsigned int *sid*, float *cenX*, float *cenY*) [private]

Definition at line 821 of file Vision.cc.

References `erouter`, `EventBase::EventTypeID_t`, and `EventRouter::postEvent()`.

7.172.2.8 void Vision::disableEvents (int *vevent*)

Definition at line 221 of file Vision.cc.

References `VisionEventSpec::listeners`, and `vevent_spec`.

7.172.2.9 void Vision::enableEvents (int *vevent*, int *noise*)

Definition at line 216 of file Vision.cc.

References `enableEvents()`, and `setNoiseThreshold()`.

7.172.2.10 void Vision::enableEvents (int *vevent*)

Definition at line 212 of file Vision.cc.

References `VisionEventSpec::listeners`, and `vevent_spec`.

7.172.2.11 bool Vision::findBall (int *ball_color*, VObject * *ball*, [VisionObjectInfo](#) * *ball_info*) [private]

Definition at line 380 of file Vision.cc.

References `addToHistHorizStrip()`, `addToHistVertStrip()`, `bound()`, `calcEdgeMask()`, `color`, `COLOR_BLUE`, `color_class_state`, `COLOR_GREEN`, `COLOR_PINK`, `COLOR_RED`, `gaussian_with_min()`, `generateEvent()`, `height`, `MAX_COLORS`, `pct_from_mean()`, `VisionEventNS::PinkBallSID`, `VisionEventNS::RedBallSID`, `VisionObjectInfo::reg`, `region`, `vevent_spec`, and `width`.

7.172.2.12 bool Vision::findGesture ([VisionObjectInfo](#) * *hand_info*)
[private]

Definition at line 779 of file Vision.cc.

References `generateEvent()`, `mathutils::limitRange()`, `num_runs`, `VisionObjectInfo::reg`, `rmap`, `VisionEventNS::ThumbsupSID`, and `vevent_spec`.

7.172.2.13 `bool Vision::findHand (VObject * hand, VisionObjectInfo * hand_info)` [private]

Definition at line 344 of file Vision.cc.

References `color`, `color_class_state`, `COLOR_SKIN`, `generateEvent()`, `VisionEventNS::HandSID`, `VisionObjectInfo::reg`, `region`, and `vevent_spec`.

7.172.2.14 `bool Vision::findMarkers ()` [private]

Definition at line 660 of file Vision.cc.

References `color`, `COLOR_BGREEN`, `COLOR_ORANGE`, `COLOR_PURPLE`, `erouter`, `identifyMarker()`, `isAdjacent()`, `markers`, `VisionEventNS::MarkersSID`, `EventManager::postEvent()`, `region`, `VisionEvent::setProperty()`, `EventBase::statusETID`, `vevent_spec`, and `vis_markers`.

7.172.2.15 `void Vision::findSpan (double & left, double & right, double x1, double x2, double y1, double y2)` [private]

7.172.2.16 `bool Vision::findThing (VObject * thing, VisionObjectInfo * thing_info)` [private]

Definition at line 600 of file Vision.cc.

References `generateEvent()`, `VisionObjectInfo::reg`, `VisionEventNS::ThingSID`, and `vevent_spec`.

7.172.2.17 `bool Vision::generateEvent (int vevent, double conf, int cenX, int cenY)` [private]

Definition at line 230 of file Vision.cc.

References `EventBase::activateETID`, `VisionEventSpec::confidence`, `createEvent()`, `EventBase::deactivateETID`, `EventBase::statusETID`, and `vevent_spec`.

7.172.2.18 `const vector3d& Vision::get_camera_dir ()` [inline]

Definition at line 134 of file Vision.h.

References `camera_dir`.

7.172.2.19 `const vector3d& Vision::get_camera_loc ()` [inline]

Definition at line 133 of file Vision.h.

References camera_loc.

7.172.2.20 `int Vision::getColor (int x, int y)` [inline]

Definition at line 135 of file Vision.h.

References cmap, COLOR_BACKGROUND, height, and width.

7.172.2.21 `int Vision::getColorUnsafe (int x, int y)` [inline, private]

Definition at line 183 of file Vision.h.

References cmap, and width.

7.172.2.22 `int Vision::getHeight ()` [inline]

Definition at line 139 of file Vision.h.

References height.

7.172.2.23 `int Vision::getNearColor (int x, int y)` [inline, private]

Definition at line 185 of file Vision.h.

References bound(), cmap, height, and width.

7.172.2.24 `vector3d Vision::getPixelDirection (double x, double y)` [private]

7.172.2.25 `int Vision::getWidth ()` [inline]

Definition at line 138 of file Vision.h.

References width.

7.172.2.26 `int Vision::identifyMarker (int color1, int color2, int color3)` [private]

Definition at line 627 of file Vision.cc.

References COLOR_ORANGE, COLOR_PURPLE, VisionInterface::MARKER_GOG, VisionInterface::MARKER_GOP, VisionInterface::MARKER_GPG, VisionInterface::MARKER_GPO, VisionInterface::MARKER_OGO,

VisionInterface::MARKER_OGP, VisionInterface::MARKER_OPG, VisionInterface::MARKER_OPO, VisionInterface::MARKER_PGO, VisionInterface::MARKER_PGP, VisionInterface::MARKER_POG, and VisionInterface::MARKER_POP.

7.172.2.27 void Vision::initialize ()

Definition at line 63 of file Vision.cc.

References bits_u, bits_v, bits_y, body_angle, body_height, cmap, cmap_t, color, Config::vision_config::colors, config, cur_tmap, frame_count, frameTimestamp, head_angles, height, initializeEventSpecs(), MAX_COLORS, max_height, max_regions, max_runs, max_width, MIN_EXP_REGION_SIZE, MIN_EXP_RUN_LENGTH, NewLarge(), num_colors, num_tmaps, obj_info, reg, Config::vision_config::resolution, rmap, rmap2, Config::vision_config::thresh, tmap, Config::vision, and width.

7.172.2.28 void Vision::initializeEventSpecs ()

Definition at line 195 of file Vision.cc.

References VisionEventSpec::count, VisionEventSpec::cx, VisionEventSpec::cy, VisionEventSpec::filter, VisionEventNS::HandSID, VisionEventSpec::listeners, VisionEventNS::MarkersSID, NUM_VEVENTS, VisionEventNS::PinkBallSID, VisionEventSpec::present, VisionEventNS::RedBallSID, VisionEventNS::ThingSID, VisionEventNS::ThumbsupSID, and vevent_spec.

7.172.2.29 int Vision::isAdjacent (region * r1, region * r2) [private]

Definition at line 618 of file Vision.cc.

7.172.2.30 int Vision::isIn (region * r1, region * r2) [private]

Definition at line 606 of file Vision.cc.

7.172.2.31 Vision Vision::operator= (const Vision &) [private]

don't assign

7.172.2.32 bool Vision::processFrame (const uchar * data_y, const uchar * data_u, const uchar * data_v, int width, int height)

Definition at line 982 of file Vision.cc.

References `frame_count`, `frameTimestamp`, `get_time()`, `img`, `obj_info`, `runHighLevelVision()`, `runLowLevelVision()`, `VisionSerializer::serialize()`, and `vser`.

7.172.2.33 **bool Vision::runHighLevelVision (ObjectInfo * *obj_info*)** [private]

Definition at line 828 of file `Vision.cc`.

References `COLOR_PINK`, `COLOR_RED`, `findBall()`, `findGesture()`, `findHand()`, `findMarkers()`, `VisionInterface::HAND`, `isIn()`, `VisionInterface::NUM_VISION_OBJECTS`, `VisionInterface::PBALL`, `VisionInterface::RBALL`, `reg`, `VisionObjectInfo::reg`, and `vobj_info`.

7.172.2.34 **template<class image> bool Vision::runLowLevelVision (*image* & *img*)** [private]

Definition at line 877 of file `Vision.cc`.

References `cmap`, `color`, `height`, `max_regions`, `max_runs`, `num_colors`, `num_regions`, `num_runs`, `reg`, `rmap`, `rmap2`, `thresholdImage()`, and `width`.

7.172.2.35 **bool Vision::saveThresholdImage (char * *filename*)**

Definition at line 1030 of file `Vision.cc`.

References `cmap`, `color`, `height`, `num_colors`, `width`, and `WritePPM()`.

7.172.2.36 **void Vision::sendColorArea ()**

7.172.2.37 **void Vision::sendRawImage ()**

7.172.2.38 **void Vision::sendRLEImage ()**

7.172.2.39 **void Vision::setCameraParam ()**

Definition at line 30 of file `Vision.cc`.

References `config`, `Config::vision_config::gain`, `Config::vision_config::shutter_speed`, `Config::vision`, and `Config::vision_config::white_balance`.

7.172.2.40 **void Vision::setNoiseThreshold (int *vevent*, int *noise*)**

Definition at line 226 of file `Vision.cc`.

References `VisionEventSpec::filter`, and `vevent_spec`.

7.172.2.41 `int Vision::setThreshold (int threshold_id)`

Definition at line 1019 of file Vision.cc.

References `cur_tmap`, and `num_tmaps`.

7.172.2.42 `bool Vision::thresholdImage (CMVision::image_yuv< const uchar > & img) [private]`

Definition at line 860 of file Vision.cc.

References `bits_u`, `bits_v`, `bits_y`, `cmap`, `cur_tmap`, and `tmap`.

7.172.2.43 `bool Vision::thresholdImage (CMVision::image_idx< rgb > & img) [private]`

Definition at line 845 of file Vision.cc.

References `cmap`, `color`, `height`, `MAX_COLORS`, and `width`.

7.172.3 Friends And Related Function Documentation**7.172.3.1** `friend class VisionSerializer [friend]`

Definition at line 98 of file Vision.h.

7.172.4 Member Data Documentation**7.172.4.1** `double Vision::body_angle`

Definition at line 123 of file Vision.h.

7.172.4.2 `double Vision::body_height`

Definition at line 123 of file Vision.h.

7.172.4.3 `bool Vision::calcTotalArea`

Definition at line 159 of file Vision.h.

7.172.4.4 [vector3d Vision::camera_dir](#)

Definition at line 126 of file Vision.h.

7.172.4.5 [vector3d Vision::camera_loc](#)

Definition at line 126 of file Vision.h.

7.172.4.6 [vector3d Vision::camera_right](#)

Definition at line 126 of file Vision.h.

7.172.4.7 [vector3d Vision::camera_up](#)

Definition at line 126 of file Vision.h.

7.172.4.8 [cmap_t* Vision::cmap](#)

Definition at line 108 of file Vision.h.

7.172.4.9 [color_class_state Vision::color](#)[MAX_COLORS]

Definition at line 116 of file Vision.h.

7.172.4.10 [int Vision::cur_tmap](#)

Definition at line 107 of file Vision.h.

7.172.4.11 [int Vision::frame_count](#)

Definition at line 104 of file Vision.h.

7.172.4.12 [unsigned long Vision::frameTimestamp](#)

Definition at line 103 of file Vision.h.

7.172.4.13 [double Vision::head_angles](#)[3]

Definition at line 124 of file Vision.h.

7.172.4.14 int [Vision::height](#)

Definition at line 118 of file Vision.h.

7.172.4.15 CMVision::image_yuv<const uchar> [Vision::img](#) [private]

Definition at line 175 of file Vision.h.

7.172.4.16 double [Vision::lin_distort_coeff](#)

Definition at line 129 of file Vision.h.

7.172.4.17 Marker [Vision::markers](#)[3]

Definition at line 161 of file Vision.h.

7.172.4.18 int [Vision::max_height](#)

Definition at line 119 of file Vision.h.

7.172.4.19 int [Vision::max_regions](#)

Definition at line 120 of file Vision.h.

7.172.4.20 int [Vision::max_runs](#)

Definition at line 120 of file Vision.h.

7.172.4.21 int [Vision::max_width](#)

Definition at line 119 of file Vision.h.

7.172.4.22 int [Vision::num_colors](#)

Definition at line 121 of file Vision.h.

7.172.4.23 int [Vision::num_regions](#)

Definition at line 121 of file Vision.h.

7.172.4.24 `int Vision::num_runs`

Definition at line 121 of file Vision.h.

7.172.4.25 `int Vision::num_tmaps`

Definition at line 106 of file Vision.h.

7.172.4.26 `ObjectInfo* Vision::obj_info`

Definition at line 163 of file Vision.h.

7.172.4.27 `int Vision::outCountAvgColor` `[private]`

Definition at line 167 of file Vision.h.

7.172.4.28 `int Vision::outCountColorArea` `[private]`

Definition at line 168 of file Vision.h.

7.172.4.29 `int Vision::outCountRaw` `[private]`

Definition at line 169 of file Vision.h.

7.172.4.30 `int Vision::outCountRLE` `[private]`

Definition at line 170 of file Vision.h.

7.172.4.31 `region* Vision::reg`

Definition at line 110 of file Vision.h.

7.172.4.32 `run* Vision::rmap`

Definition at line 109 of file Vision.h.

7.172.4.33 `run * Vision::rmap2`

Definition at line 109 of file Vision.h.

7.172.4.34 **double** [Vision::sq_distort_coeff](#)

Definition at line 129 of file Vision.h.

7.172.4.35 **cmap_t** * [Vision::tmap](#)[MAX_TMAPS]

Definition at line 108 of file Vision.h.

7.172.4.36 [VisionEventSpec](#) [Vision::vevent_spec](#)[NUM_VEVENTS]

Definition at line 114 of file Vision.h.

7.172.4.37 **int** [Vision::vis_markers](#)

Definition at line 162 of file Vision.h.

7.172.4.38 [VisionObjectInfo](#) [Vision::vobj_info](#)[NUM_VISION_OBJECTS]

Definition at line 113 of file Vision.h.

7.172.4.39 [VisionSerializer](#)* [Vision::vser](#) [private]

Definition at line 172 of file Vision.h.

7.172.4.40 **int** [Vision::width](#)

Definition at line 118 of file Vision.h.

7.172.4.41 **int** [Vision::yindex](#)[144]

Definition at line 111 of file Vision.h.

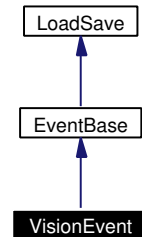
The documentation for this class was generated from the following files:

- [Vision.h](#)
- [Vision.cc](#)

7.173 VisionEvent Class Reference

```
#include <VisionEvent.h>
```

Inheritance diagram for VisionEvent:



7.173.1 Detailed Description

Extends [EventBase](#) to also include location in the visual field and distance (though distance is not implimented yet).

Definition at line 22 of file VisionEvent.h.

Public Member Functions

- [VisionEvent](#) ()
Constructor.
- [VisionEvent](#) ([EventTypeID_t](#) tid, unsigned int sid)
Constructor, pass a type id and source id.
- [VisionEvent](#) ([EventTypeID_t](#) tid, unsigned int sid, float cenX, float cenY)
Constructor, pass the type id, source id, center X and center Y.
- float [getCenterX](#) () const
returns the x coordinate
- [VisionEvent](#) & [setCenterX](#) (float cenX)
sets the x coordinate
- float [getCenterY](#) () const
returns the y coordinate

- [VisionEvent](#) & [setCenterY](#) (float cenY)
sets the y coordinate
- float [getDistance](#) () const
returns the distance (not implemented)
- [VisionEvent](#) & [setDistance](#) (float dist)
sets the distance
- int [getProperty](#) () const
returns the property
- [VisionEvent](#) & [setProperty](#) (int property)
sets the property
- virtual unsigned int [getBinSize](#) () const
calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.
- virtual unsigned int [LoadBuffer](#) (const char buf[], unsigned int len)
Load from a saved buffer.
- virtual unsigned int [SaveBuffer](#) (char buf[], unsigned int len) const
Save to a given buffer.

Protected Attributes

- float [_cenX](#)
a value representing location in visual field - from -1 if on the left edge to 1 if it's on the right edge
- float [_cenY](#)
a value representing location in visual field - from -1 if on the bottom edge to 1 if it's on the top edge
- float [_distance](#)
distance from snout to object in millimeters.
- int [_property](#)
some property, depending on the SID

7.173.2 Constructor & Destructor Documentation

7.173.2.1 VisionEvent::VisionEvent () [inline]

Constructor.

Definition at line 25 of file VisionEvent.h.

References `_cenX`, `_cenY`, `_distance`, `_property`, `EventBase::statusETID`, and `EventBase::visionEGID`.

7.173.2.2 VisionEvent::VisionEvent ([EventTypeID_t tid](#), unsigned int *sid*) [inline]

Constructor, pass a type id and source id.

Definition at line 27 of file VisionEvent.h.

References `_cenX`, `_cenY`, `_distance`, `_property`, and `EventBase::visionEGID`.

7.173.2.3 VisionEvent::VisionEvent ([EventTypeID_t tid](#), unsigned int *sid*, float *cenX*, float *cenY*) [inline]

Constructor, pass the type id, source id, center X and center Y.

Definition at line 29 of file VisionEvent.h.

References `_cenX`, `_cenY`, `_distance`, `_property`, and `EventBase::visionEGID`.

7.173.3 Member Function Documentation

7.173.3.1 virtual unsigned int VisionEvent::getBinSize () const [inline, virtual]

calculates space needed to save - if you can't precisely add up the size, overestimate and things will still work.

Returns:

number of bytes read/written, 0 if error (or empty)

Reimplemented from [EventBase](#).

Definition at line 43 of file VisionEvent.h.

References `_cenX`, `_cenY`, `_distance`, `_property`, `LoadSave::creatorSize()`, and `EventBase::getBinSize()`.

7.173.3.2 float VisionEvent::getCenterX () const [inline]

returns the x coordinate

Definition at line 31 of file VisionEvent.h.

References `_cenX`.

7.173.3.3 float VisionEvent::getCenterY () const [inline]

returns the y coordinate

Definition at line 34 of file VisionEvent.h.

References `_cenY`.

7.173.3.4 float VisionEvent::getDistance () const [inline]

returns the distance (not implemented)

Definition at line 37 of file VisionEvent.h.

References `_distance`.

7.173.3.5 int VisionEvent::getProperty () const [inline]

returns the property

Definition at line 40 of file VisionEvent.h.

References `_property`.

7.173.3.6 virtual unsigned int VisionEvent::LoadBuffer (const char *buf* [], unsigned int *len*) [inline, virtual]

Load from a saved buffer.

Parameters:

buf pointer to the memory where you should begin loading

len length of *buf* available (this isn't all yours, might be more stuff saved after yours)

Returns:

the number of bytes actually used

Reimplemented from [EventBase](#).

Definition at line 53 of file VisionEvent.h.

References `_cenX`, `_cenY`, `_distance`, `_property`, `LoadSave::checkCreator()`, `LoadSave::decode()`, and `EventBase::LoadBuffer()`.

7.173.3.7 virtual unsigned int VisionEvent::SaveBuffer (char *buf* [], unsigned int *len*) const `[inline, virtual]`

Save to a given buffer.

Parameters:

buf pointer to the memory where you should begin writing

len length of *buf* available. (this isn't all yours, constrain yourself to what you returned in `getBinSize()`)

Returns:

the number of bytes actually used

Reimplemented from [EventBase](#).

Definition at line 71 of file VisionEvent.h.

References `_cenX`, `_cenY`, `_distance`, `_property`, `LoadSave::encode()`, `EventBase::SaveBuffer()`, and `LoadSave::saveCreator()`.

7.173.3.8 [VisionEvent](#)& VisionEvent::setCenterX (float *cenX*) `[inline]`

sets the x coordinate

Definition at line 32 of file VisionEvent.h.

References `_cenX`.

7.173.3.9 [VisionEvent](#)& VisionEvent::setCenterY (float *cenY*) `[inline]`

sets the y coordinate

Definition at line 35 of file VisionEvent.h.

References `_cenY`.

7.173.3.10 [VisionEvent](#)& VisionEvent::setDistance (float *dist*) `[inline]`

sets the distance

Definition at line 38 of file VisionEvent.h.

References `_distance`.

7.173.3.11 [VisionEvent](#)& [VisionEvent::setProperty](#) (int *property*) [inline]

sets the property

Definition at line 41 of file [VisionEvent.h](#).

References [_property](#).

7.173.4 Member Data Documentation**7.173.4.1** float [VisionEvent::_cenX](#) [protected]

a value representing location in visual field - from -1 if on the left edge to 1 if it's on the right edge

Definition at line 90 of file [VisionEvent.h](#).

7.173.4.2 float [VisionEvent::_cenY](#) [protected]

a value representing location in visual field - from -1 if on the bottom edge to 1 if it's on the top edge

Definition at line 91 of file [VisionEvent.h](#).

7.173.4.3 float [VisionEvent::_distance](#) [protected]

distance from snout to object in millimeters.

Definition at line 92 of file [VisionEvent.h](#).

7.173.4.4 int [VisionEvent::_property](#) [protected]

some property, depending on the SID

Definition at line 93 of file [VisionEvent.h](#).

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

- [VisionEvent.h](#)

7.174 VisionEventSpec Struct Reference

```
#include <Vision.h>
```

Public Attributes

- int [listeners](#)
- int [filter](#)
- int [count](#)
- double [confidence](#)
- float [cx](#)
- float [cy](#)
- bool [present](#)

7.174.1 Member Data Documentation

7.174.1.1 double [VisionEventSpec::confidence](#)

Definition at line 90 of file Vision.h.

7.174.1.2 int [VisionEventSpec::count](#)

Definition at line 89 of file Vision.h.

7.174.1.3 float [VisionEventSpec::cx](#)

Definition at line 91 of file Vision.h.

7.174.1.4 float [VisionEventSpec::cy](#)

Definition at line 91 of file Vision.h.

7.174.1.5 int [VisionEventSpec::filter](#)

Definition at line 88 of file Vision.h.

7.174.1.6 int [VisionEventSpec::listeners](#)

Definition at line 87 of file Vision.h.

7.174.1.7 **bool** [VisionEventSpec::present](#)

Definition at line 92 of file Vision.h.

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

- [Vision.h](#)

7.175 VisionObjectInfo Struct Reference

```
#include <Vision.h>
```

Public Attributes

- [region](#) * [reg](#)
- [region](#) * [reg2](#)

7.175.1 Member Data Documentation

7.175.1.1 [region](#)* [VisionObjectInfo::reg](#)

Definition at line 76 of file Vision.h.

7.175.1.2 [region](#) * [VisionObjectInfo::reg2](#)

Definition at line 76 of file Vision.h.

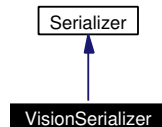
The documentation for this struct was generated from the following file:

- [Vision.h](#)

7.176 VisionSerializer Class Reference

```
#include <VisionSerializer.h>
```

Inheritance diagram for VisionSerializer:



7.176.1 Detailed Description

Encodes and transmits camera images.

Definition at line 9 of file VisionSerializer.h.

Public Member Functions

- [VisionSerializer](#) ()
constructor
- void [serialize](#) ()
encodes and sends current frame

Private Member Functions

- void [encodeVisionRaw](#) (char *buf, CMVision::image_yuv< const uchar > &img, int scale)
encodes the original camera image
- void [encodeVisionRLE](#) (char *buf, int num_runs, [run](#) *runs)
encodes the RLE processed image
- char * [encodeVisionRun](#) (char *buf, [run](#) *run)
helper function for encodeVisionRLE, encodes one run length
- [VisionSerializer](#) (const [VisionSerializer](#) &)
don't call

- [VisionSerializer](#) & `operator=` (const [VisionSerializer](#) &)
don't call

Private Attributes

- [Socket](#) * [visRaw](#)
socket to which raw vision images should be sent
- [Socket](#) * [visRLE](#)
socket to which RLE images should be sent

7.176.2 Constructor & Destructor Documentation

7.176.2.1 [VisionSerializer::VisionSerializer](#) ()

constructor

Definition at line 35 of file [VisionSerializer.cc](#).

References [config](#), [Wireless::listen\(\)](#), [Config::vision_config::raw_port](#), [Config::vision_config::rle_port](#), [Wireless::socket\(\)](#), [Config::vision](#), [visRaw](#), [visRLE](#), and [wireless](#).

7.176.2.2 [VisionSerializer::VisionSerializer](#) (const [VisionSerializer](#) &) [private]

don't call

7.176.3 Member Function Documentation

7.176.3.1 `void VisionSerializer::encodeVisionRaw (char * buf, CMVision::image_yuv< const uchar > & img, int scale)` [private]

encodes the original camera image

Definition at line 77 of file [VisionSerializer.cc](#).

7.176.3.2 `void VisionSerializer::encodeVisionRLE (char * buf, int num_runs, run * runs)` [private]

encodes the RLE processed image

Definition at line 99 of file VisionSerializer.cc.

References `Serializer::encode()`, and `encodeVisionRun()`.

7.176.3.3 `char * VisionSerializer::encodeVisionRun (char * buf, run * run)`
[inline, private]

helper function for `encodeVisionRLE`, encodes one run length

Definition at line 69 of file VisionSerializer.cc.

7.176.3.4 `VisionSerializer & VisionSerializer::operator= (const VisionSerializer &)` [private]

don't call

7.176.3.5 `void VisionSerializer::serialize ()`

encodes and sends current frame

Definition at line 43 of file VisionSerializer.cc.

References `encodeVisionRaw()`, `encodeVisionRLE()`, `Socket::getWriteBuffer()`, `Vision::img`, `Vision::num_runs`, `Vision::rmap`, `vision`, `visRaw`, `visRLE`, and `Socket::write()`.

7.176.4 Member Data Documentation

7.176.4.1 `Socket* VisionSerializer::visRaw` [private]

socket to which raw vision images should be sent

Definition at line 26 of file VisionSerializer.h.

7.176.4.2 `Socket* VisionSerializer::visRLE` [private]

socket to which RLE images should be sent

Definition at line 27 of file VisionSerializer.h.

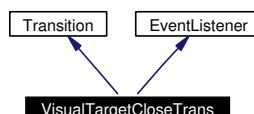
The documentation for this class was generated from the following files:

- [VisionSerializer.h](#)
- [VisionSerializer.cc](#)

7.177 VisualTargetCloseTrans Class Reference

```
#include <VisualTargetCloseTrans.h>
```

Inheritance diagram for VisualTargetCloseTrans:



7.177.1 Detailed Description

causes a transition when a visual object is "close"

Definition at line 10 of file VisualTargetCloseTrans.h.

Public Member Functions

- [VisualTargetCloseTrans](#) ([StateNode](#) *source, [StateNode](#) *destination, [VisionEventNS::VisionSourceID_t](#) source_id)
constructor
- virtual void [enable](#) ()
starts listening for the object specified by the source id in the constructor
- virtual void [disable](#) ()
called by [StateNode](#) when it becomes inactive - undo whatever you did in [Enable\(\)](#)
- virtual void [processEvent](#) (const [EventBase](#) &e)
if the object is "close", calls [activate\(\)](#)

Protected Attributes

- [VisionEventNS::VisionSourceID_t](#) sid
Source ID of object to track.

7.177.2 Constructor & Destructor Documentation

7.177.2.1 `VisualTargetCloseTrans::VisualTargetCloseTrans (StateNode * source, StateNode * destination, VisionEventNS::VisionSourceID_t source_id) [inline, explicit]`

constructor

Definition at line 13 of file VisualTargetCloseTrans.h.

References `sid`.

7.177.3 Member Function Documentation

7.177.3.1 `virtual void VisualTargetCloseTrans::disable () [inline, virtual]`

called by [StateNode](#) when it becomes inactive - undo whatever you did in `Enable()`

Implements [Transition](#).

Definition at line 19 of file VisualTargetCloseTrans.h.

References `erouter`, and `EventRouter::forgetListener()`.

7.177.3.2 `virtual void VisualTargetCloseTrans::enable () [inline, virtual]`

starts listening for the object specified by the source id in the constructor

Implements [Transition](#).

Definition at line 16 of file VisualTargetCloseTrans.h.

References `EventRouter::addListener()`, `erouter`, `sid`, and `EventBase::visionEGID`.

7.177.3.3 `virtual void VisualTargetCloseTrans::processEvent (const EventBase & e) [inline, virtual]`

if the object is "close", calls [activate\(\)](#)

Todo

need to activate if it's "close"

Implements [EventListener](#).

Definition at line 22 of file VisualTargetCloseTrans.h.

References `ASSERTRET`.

7.177.4 Member Data Documentation

7.177.4.1 [VisionEventNS::VisionSourceID.t](#) [VisualTargetCloseTrans::sid](#) [protected]

Source ID of object to track.

Definition at line 29 of file [VisualTargetCloseTrans.h](#).

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

- [VisualTargetCloseTrans.h](#)

7.178 VisionInterface::VObject Struct Reference

```
#include <VisionInterface.h>
```

Public Attributes

- double [confidence](#)
- [vector3d](#) [loc](#)
- double [left](#)
- double [right](#)
- double [distance](#)
- uchar [edge](#)

7.178.1 Member Data Documentation

7.178.1.1 double [VisionInterface::VObject::confidence](#)

Definition at line 49 of file VisionInterface.h.

7.178.1.2 double [VisionInterface::VObject::distance](#)

Definition at line 52 of file VisionInterface.h.

7.178.1.3 uchar [VisionInterface::VObject::edge](#)

Definition at line 53 of file VisionInterface.h.

7.178.1.4 double [VisionInterface::VObject::left](#)

Definition at line 51 of file VisionInterface.h.

7.178.1.5 [vector3d](#) [VisionInterface::VObject::loc](#)

Definition at line 50 of file VisionInterface.h.

7.178.1.6 double [VisionInterface::VObject::right](#)

Definition at line 51 of file VisionInterface.h.

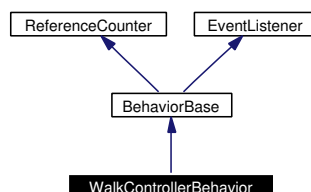
The documentation for this struct was generated from the following file:

- [VisionInterface.h](#)

7.179 WalkControllerBehavior Class Reference

```
#include <WalkControllerBehavior.h>
```

Inheritance diagram for WalkControllerBehavior:



7.179.1 Detailed Description

Listens to control commands coming in from the command port for remotely controlling the walk.

Definition at line 17 of file WalkControllerBehavior.h.

Public Member Functions

- [WalkControllerBehavior](#) ()
constructor
- virtual [~WalkControllerBehavior](#) ()
destructor
- virtual void [DoStart](#) ()
By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.
- virtual void [DoStop](#) ()
By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).
- virtual void [processEvent](#) (const [EventBase](#) &)
The only event we could possibly receive is the stop-if-no-heartbeat timer.
- virtual std::string [getName](#) () const
Identifies the behavior in menus and such.

Static Public Member Functions

- int [mechacmd_callback](#) (char *buf, int bytes)
called by wireless when there's new data
- std::string [getClassDescription](#) ()
Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Static Public Attributes

- [WalkControllerBehavior](#) * [theOne](#) = NULL

Protected Attributes

- [MotionManager::MC_ID](#) [walker_id](#)
the [WalkMC](#) to use

Private Member Functions

- void [runCommand](#) (unsigned char *command)
Executes a command. Called by [mechacmd_callback](#).
- [WalkControllerBehavior](#) (const [WalkControllerBehavior](#) &)
don't call
- [WalkControllerBehavior](#) operator= (const [WalkControllerBehavior](#) &)
don't call

Private Attributes

- float [dx](#)
Motion parameter.
- float [dy](#)
Motion parameter.
- float [da](#)

Motion parameter.

- [WalkControllerBehavior](#) * [theLastOne](#)
- [Socket](#) * [cmdsock](#)

The input command stream socket.

Static Private Attributes

Command Bytes

- const char [CMD_fwd](#) = 'f'
handy symbol for matching incoming communication
- const char [CMD_roto](#) = 'r'
handy symbol for matching incoming communication
- const char [CMD_side](#) = 's'
handy symbol for matching incoming communication
- const char [CMD_opt0](#) = '0'
handy symbol for matching incoming communication
- const char [CMD_opt1](#) = '1'
handy symbol for matching incoming communication
- const char [CMD_opt2](#) = '2'
handy symbol for matching incoming communication
- const char [CMD_opt3](#) = '3'
handy symbol for matching incoming communication
- const char [CMD_opt4](#) = '4'
handy symbol for matching incoming communication
- const char [CMD_opt5](#) = '5'
handy symbol for matching incoming communication
- const char [CMD_opt6](#) = '6'
handy symbol for matching incoming communication
- const char [CMD_opt7](#) = '7'
handy symbol for matching incoming communication
- const char [CMD_opt8](#) = '8'
handy symbol for matching incoming communication

- const char [CMD_opt9](#) = '9'
handy symbol for matching incoming communication

7.179.2 Constructor & Destructor Documentation

7.179.2.1 **WalkControllerBehavior::WalkControllerBehavior** (const [WalkControllerBehavior](#) &) [private]

don't call

7.179.2.2 **WalkControllerBehavior::WalkControllerBehavior** () [inline]

constructor

Definition at line 68 of file WalkControllerBehavior.h.

References [cmdsock](#), [da](#), [dx](#), [dy](#), [SocketNS::SOCK_STREAM](#), [theLastOne](#), [theOne](#), [walker_id](#), and [wireless](#).

7.179.2.3 **virtual WalkControllerBehavior::~~WalkControllerBehavior** () [inline, virtual]

destructor

Definition at line 76 of file WalkControllerBehavior.h.

References [theLastOne](#), and [theOne](#).

7.179.3 Member Function Documentation

7.179.3.1 **void WalkControllerBehavior::DoStart** () [virtual]

By default, merely adds to the reference counter (through [AddReference\(\)](#)) you should still call this from your overriding methods.

Reimplemented from [BehaviorBase](#).

Definition at line 81 of file WalkControllerBehavior.cc.

References [EventManager::addListener\(\)](#), [MotionManager::addMotion\(\)](#), [cmdsock](#), [config](#), [BehaviorBase::DoStart\(\)](#), [erouter](#), [Wireless::listen\(\)](#), [Controller::loadGUI\(\)](#), [Config::main](#), [mechacmd_callback\(\)](#), [motman](#), [Wireless::setReceiver\(\)](#), [Socket::sock](#), [EventBase::timerEGID](#), [Config::main_config::walkControl_port](#), [walker_id](#), and [wireless](#).

7.179.3.2 void WalkControllerBehavior::DoStop () [virtual]

By default, subtracts from the reference counter, and deletes if zero you should still call this when you override this call this at the end of your [DoStop\(\)](#), not beginning (it might delete this).

Reimplemented from [BehaviorBase](#).

Definition at line 95 of file WalkControllerBehavior.cc.

References [Wireless::close\(\)](#), [Controller::closeGUI\(\)](#), [cmdsock](#), [BehaviorBase::DoStop\(\)](#), [erouter](#), [EventRouter::forgetListener\(\)](#), [motman](#), [MotionManager::removeMotion\(\)](#), [walker_id](#), and [wireless](#).

7.179.3.3 std::string WalkControllerBehavior::getClassDescription () [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 89 of file WalkControllerBehavior.h.

References [config](#), [Config::main](#), and [Config::main_config::walkControl_port](#).

7.179.3.4 virtual std::string WalkControllerBehavior::getName () const [inline, virtual]

Identifies the behavior in menus and such.

Implements [BehaviorBase](#).

Definition at line 88 of file WalkControllerBehavior.h.

7.179.3.5 int WalkControllerBehavior::mechacmd_callback (char * buf, int bytes) [static]

called by wireless when there's new data

Definition at line 109 of file WalkControllerBehavior.cc.

References [runCommand\(\)](#), and [theOne](#).

7.179.3.6 [WalkControllerBehavior](#) WalkControllerBehavior::operator= (const [WalkControllerBehavior](#) &) [private]

don't call

7.179.3.7 **virtual void WalkControllerBehavior::processEvent (const [EventBase](#) &) [inline, virtual]**

The only event we could possibly receive is the stop-if-no-heartbeat timer.

Reimplemented from [BehaviorBase](#).

Definition at line 83 of file WalkControllerBehavior.h.

References `walker_id`.

7.179.3.8 **void WalkControllerBehavior::runCommand (unsigned char * *command*) [private]**

Executes a command. Called by `mechacmd_callback`.

Definition at line 6 of file WalkControllerBehavior.cc.

References `EventRouter::addTimer()`, `CMD_fwd`, `CMD_opt0`, `CMD_opt1`, `CMD_opt2`, `CMD_opt3`, `CMD_opt4`, `CMD_opt5`, `CMD_opt6`, `CMD_opt7`, `CMD_opt8`, `CMD_opt9`, `CMD_roto`, `CMD_side`, `da`, `dx`, `dy`, `erouter`, `SoundManager::PlayFile()`, `EventRouter::removeTimer()`, `sndman`, and `walker_id`.

7.179.4 Member Data Documentation

7.179.4.1 **const char [WalkControllerBehavior::CMD_fwd](#) = 'f' [static, private]**

handy symbol for matching incoming communication

Definition at line 31 of file WalkControllerBehavior.h.

7.179.4.2 **const char [WalkControllerBehavior::CMD_opt0](#) = '0' [static, private]**

handy symbol for matching incoming communication

Definition at line 34 of file WalkControllerBehavior.h.

7.179.4.3 **const char [WalkControllerBehavior::CMD_opt1](#) = '1' [static, private]**

handy symbol for matching incoming communication

Definition at line 35 of file WalkControllerBehavior.h.

7.179.4.4 `const char WalkControllerBehavior::CMD_opt2 = '2' [static,
private]`

handy symbol for matching incoming communication

Definition at line 36 of file WalkControllerBehavior.h.

7.179.4.5 `const char WalkControllerBehavior::CMD_opt3 = '3' [static,
private]`

handy symbol for matching incoming communication

Definition at line 37 of file WalkControllerBehavior.h.

7.179.4.6 `const char WalkControllerBehavior::CMD_opt4 = '4' [static,
private]`

handy symbol for matching incoming communication

Definition at line 38 of file WalkControllerBehavior.h.

7.179.4.7 `const char WalkControllerBehavior::CMD_opt5 = '5' [static,
private]`

handy symbol for matching incoming communication

Definition at line 39 of file WalkControllerBehavior.h.

7.179.4.8 `const char WalkControllerBehavior::CMD_opt6 = '6' [static,
private]`

handy symbol for matching incoming communication

Definition at line 40 of file WalkControllerBehavior.h.

7.179.4.9 `const char WalkControllerBehavior::CMD_opt7 = '7' [static,
private]`

handy symbol for matching incoming communication

Definition at line 41 of file WalkControllerBehavior.h.

7.179.4.10 `const char WalkControllerBehavior::CMD_opt8 = '8'` [static, private]

handy symbol for matching incoming communication

Definition at line 42 of file WalkControllerBehavior.h.

7.179.4.11 `const char WalkControllerBehavior::CMD_opt9 = '9'` [static, private]

handy symbol for matching incoming communication

Definition at line 43 of file WalkControllerBehavior.h.

7.179.4.12 `const char WalkControllerBehavior::CMD_roto = 'r'` [static, private]

handy symbol for matching incoming communication

Definition at line 32 of file WalkControllerBehavior.h.

7.179.4.13 `const char WalkControllerBehavior::CMD_side = 's'` [static, private]

handy symbol for matching incoming communication

Definition at line 33 of file WalkControllerBehavior.h.

7.179.4.14 `Socket* WalkControllerBehavior::cmdsock` [private]

The input command stream socket.

Definition at line 58 of file WalkControllerBehavior.h.

7.179.4.15 `float WalkControllerBehavior::da` [private]

Motion parameter.

Definition at line 48 of file WalkControllerBehavior.h.

7.179.4.16 `float WalkControllerBehavior::dx` [private]

Motion parameter.

Definition at line 46 of file WalkControllerBehavior.h.

7.179.4.17 float [WalkControllerBehavior::dy](#) [private]

Motion parameter.

Definition at line 47 of file WalkControllerBehavior.h.

7.179.4.18 [WalkControllerBehavior*](#) [WalkControllerBehavior::theLastOne](#) [private]

The last WCB object that was theOne, so we can restore it to prominence when we die. This is a nice gesture, but it doesn't really make sense since we're all using the same port. But just in case something changes and we don't do that, this mechanism is in place.

Definition at line 55 of file WalkControllerBehavior.h.

7.179.4.19 [WalkControllerBehavior*](#) [WalkControllerBehavior::theOne](#) = NULL [static]

Points to the one WalkControllerBehavior object that the input command stream is talking to. A kludge. Dunno how you're gonna make sure you're not using this uninitialized.

Definition at line 4 of file WalkControllerBehavior.cc.

7.179.4.20 [MotionManager::MC_ID](#) [WalkControllerBehavior::walker_id](#) [protected]

the [WalkMC](#) to use

Definition at line 27 of file WalkControllerBehavior.h.

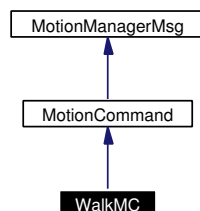
The documentation for this class was generated from the following files:

- [WalkControllerBehavior.h](#)
- [WalkControllerBehavior.cc](#)

7.180 WalkMC Class Reference

```
#include <WalkMC.h>
```

Inheritance diagram for WalkMC:



7.180.1 Detailed Description

A nice walking class from Carnegie Mellon University's 2001 Robosoccer team, modified to fit this framework, see their [license](#).

Moves the feet through a looping path in order to walk - default parameters use a walk low to the ground so you don't walk over the ball.

This portion of the code falls under CMPack's license:

```

=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
-----
This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
-----
Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed. In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
  
```

Definition at line 49 of file WalkMC.h.

Public Types

- typedef [SplinePath](#)< [vector3d](#), double > [splinepath](#)
for convenience
- typedef [HermiteSplineSegment](#)< [vector3d](#), double > [spline](#)
for convenience

Public Member Functions

- [WalkMC](#) (const char *pfile=NULL)
constructor
- virtual void [DoStart](#) ()
sends an activate [LocomotionEvent](#)
- virtual void [DoStop](#) ()
sends a deactivate [LocomotionEvent](#)
- virtual int [updateOutputs](#) ()
calculates current positions of the paws
- virtual int [isDirty](#) ()
returns true if we are walking
- virtual int [isAlive](#) ()
always true - never autoprunes
- void [load](#) (const char *pfile)
loads parameters from a file (
- void [save](#) (const char *pfile) const
saves parameters to a file (
- void [setTargetVelocity](#) (double [dx](#), double [dy](#), double [da](#))
set the direction to walk - can specify x (forward), y (left), and angular (counterclockwise) velocities
- const [vector3d](#) & [getTargetVelocity](#) ()

returns current velocity we're trying to go

- const [vector3d](#) & [getCurVelocity](#) () const
returns the velocity we're actually moving (subject to clipping at max_accel_xya), doesn't reflect value of [getPaused\(\)](#)...
- unsigned int [getTravelTime](#) ()
returns the time we've been traveling along the current vector
- void [setPaused](#) (bool p)
if set to true, will stop moving
- bool [getPaused](#) () const
if is true, we aren't moving
- void [setHeight](#) (double h)
sets [WalkParam::body_height](#) of wp
- double [getHeight](#) ()
gets [WalkParam::body_height](#) of wp
- void [setAngle](#) (double a)
sets [WalkParam::body_angle](#) of wp
- double [getAngle](#) ()
gets [WalkParam::body_angle](#) of wp
- void [setHop](#) (double h)
sets [WalkParam::hop](#) of wp
- double [getHop](#) ()
gets [WalkParam::hop](#) of wp
- void [setSway](#) (double h)
sets [WalkParam::sway](#) of wp
- double [getSway](#) ()
gets [WalkParam::sway](#) of wp
- void [setPeriod](#) (long p)
sets [WalkParam::period](#) of wp
- long [getPeriod](#) ()

gets [WalkParam::period](#) of *wp*

- void [resetLegPos](#) ()
takes current leg positions from [WorldState](#) and tries to match the point in the cycle most like it

Static Public Attributes

- const float [MAX_DX](#) = 180
==180 mm/sec
- const float [MAX_DY](#) = 140
==140 mm/sec
- const float [MAX_DA](#) = 1.8
==1.8 rad/sec
- const [vector3d](#) [max_accel_xya](#)
vector version of MAX_DX,MAX_DY,MAX_DA

Protected Member Functions

- void [init](#) (const char *pfile)
does some setup stuff, calls load(pfile)

Protected Attributes

- [OutputCmd](#) [cmds](#) [NumOutputs][NumFrames]
holds current joint commands
- bool [isPaused](#)
true if we are paused
- [WalkParam](#) [wp](#)
current walking parameters (note that it's not static - different WalkMC's can have different setting, handy...
- [LegWalkState](#) [legw](#) [NumLegs]
current state of each leg

- [vector3d legpos](#) [NumLegs]
current position of each leg
- [splinepath body_loc](#)
the path the body goes through while walking (?)
- [splinepath body_angle](#)
the path the body goes through while walking (?)
- [vector3d pos_delta](#)
how much we've moved
- [double angle_delta](#)
how much we've turned
- [unsigned int travelTime](#)
the time since the last call to setTargetVelocity - handy to check the time we've been traveling current vector
- [int time](#)
time of last call to updateJointCmds()
- [int TimeStep](#)
time to pretend passes between each call to updateJointCmds() - usually Robot-Info::FrameTime, unless you want to make it walk in slow motion (handy sometimes for debugging)
- [vector3d vel_xya](#)
the current velocity we're moving
- [vector3d target_vel_xya](#)
the velocity that was requested

7.180.2 Member Typedef Documentation

7.180.2.1 typedef [HermiteSplineSegment](#)<[vector3d](#),double> [WalkMC::spline](#)

for convenience

Definition at line 52 of file WalkMC.h.

7.180.2.2 typedef [SplinePath](#)<[vector3d](#),double> [WalkMC::splinepath](#)

for convenience

Definition at line 51 of file WalkMC.h.

7.180.3 Constructor & Destructor Documentation

7.180.3.1 [WalkMC::WalkMC](#) (const char * *pfile* = NULL)

constructor

Definition at line 59 of file WalkMC.cc.

References [ERS210Info::FrameTime](#), [get_time\(\)](#), [init\(\)](#), and [time](#).

7.180.4 Member Function Documentation

7.180.4.1 void [WalkMC::DoStart](#) () [virtual]

sends an activate [LocomotionEvent](#)

Reimplemented from [MotionCommand](#).

Definition at line 68 of file WalkMC.cc.

References [EventBase::activateETID](#), [MotionCommand::DoStart\(\)](#), [get_time\(\)](#), [MotionManagerMsg::getID\(\)](#), [EventBase::locomotionEGID](#), [MotionCommand::post-Event\(\)](#), [LocomotionEvent::setXYA\(\)](#), [target_vel_xya](#), [travelTime](#), [GVector::vector3d< double >::x](#), [GVector::vector3d< double >::y](#), and [GVector::vector3d< double >::z](#).

7.180.4.2 void [WalkMC::DoStop](#) () [virtual]

sends a deactivate [LocomotionEvent](#)

Reimplemented from [MotionCommand](#).

Definition at line 76 of file WalkMC.cc.

References [EventBase::deactivateETID](#), [MotionCommand::DoStop\(\)](#), [get_time\(\)](#), [MotionManagerMsg::getID\(\)](#), [EventBase::locomotionEGID](#), [MotionCommand::post-Event\(\)](#), [LocomotionEvent::setXYA\(\)](#), [target_vel_xya](#), [travelTime](#), [GVector::vector3d< double >::x](#), [GVector::vector3d< double >::y](#), and [GVector::vector3d< double >::z](#).

7.180.4.3 double [WalkMC::getAngle](#) () [inline]

gets [WalkParam::body_angle](#) of [wp](#)

Definition at line 111 of file WalkMC.h.

References WalkMC::WalkParam::body_angle, and wp.

7.180.4.4 **const [vector3d](#)& WalkMC::getCurVelocity () const** [[inline](#)]

returns the velocity we're actually moving (subject to clipping at max_accel_xya), doesn't reflect value of [getPaused\(\)](#)...

Definition at line 102 of file WalkMC.h.

References vel_xya.

7.180.4.5 **double WalkMC::getHeight ()** [[inline](#)]

gets [WalkParam::body_height](#) of wp

Definition at line 109 of file WalkMC.h.

References WalkMC::WalkParam::body_height, and wp.

7.180.4.6 **double WalkMC::getHop ()** [[inline](#)]

gets [WalkParam::hop](#) of wp

Definition at line 113 of file WalkMC.h.

References WalkMC::WalkParam::hop, and wp.

7.180.4.7 **bool WalkMC::getPaused () const** [[inline](#)]

if is true, we aren't moving

Definition at line 107 of file WalkMC.h.

References isPaused.

7.180.4.8 **long WalkMC::getPeriod ()** [[inline](#)]

gets [WalkParam::period](#) of wp

Definition at line 117 of file WalkMC.h.

References WalkMC::WalkParam::period, and wp.

7.180.4.9 `double WalkMC::getSway ()` [inline]

gets [WalkParam::sway](#) of [wp](#)

Definition at line 115 of file WalkMC.h.

References [WalkMC::WalkParam::sway](#), and [wp](#).

7.180.4.10 `const vector3d& WalkMC::getTargetVelocity ()` [inline]

returns current velocity we're trying to go

Definition at line 100 of file WalkMC.h.

References [target_vel_xya](#).

7.180.4.11 `unsigned int WalkMC::getTravelTime ()` [inline]

returns the time we've been traveling along the current vector

Definition at line 104 of file WalkMC.h.

References [get.time\(\)](#), and [travelTime](#).

7.180.4.12 `void WalkMC::init (const char * pfile)` [protected]

does some setup stuff, calls [load\(pfile\)](#)

Definition at line 85 of file WalkMC.cc.

References [WalkMC::LegWalkState::air](#), [WalkMC::WalkParam::body_angle](#), [body_-angle](#), [WalkMC::WalkParam::body_height](#), [body_loc](#), [GetLegPosition\(\)](#), [Spline-Path< vector3d, double >::init\(\)](#), [ERS210Info::JointsPerLeg](#), [legpos](#), [legw](#), [load\(\)](#), [ERS210Info::NumLegs](#), [vector3d](#), and [wp](#).

7.180.4.13 `virtual int WalkMC::isAlive ()` [inline, virtual]

always true - never autoprunes

Implements [MotionCommand](#).

Definition at line 91 of file WalkMC.h.

7.180.4.14 `virtual int WalkMC::isDirty ()` [inline, virtual]

returns true if we are walking

Implements [MotionCommand](#).

Definition at line 90 of file WalkMC.h.

References `isPaused`, `target_vel_xya`, `GVector::vector3d< double >::x`, `GVector::vector3d< double >::y`, and `GVector::vector3d< double >::z`.

7.180.4.15 `void WalkMC::load (const char * pfile)`

loads parameters from a file (

Todo

use [LoadSave](#))

Definition at line 109 of file WalkMC.cc.

References `checksum()`, `mzero()`, `read_file()`, and `wp`.

7.180.4.16 `void WalkMC::resetLegPos ()`

takes current leg positions from [WorldState](#) and tries to match the point in the cycle most like it

Definition at line 228 of file WalkMC.cc.

References `GetLegPosition()`, `ERS210Info::JointsPerLeg`, `ERS210Info::LegOffset`, `legpos`, `ERS210Info::NumLegs`, `WorldState::outputs`, and `state`.

7.180.4.17 `void WalkMC::save (const char * pfile) const`

saves parameters to a file (

Todo

use [LoadSave](#))

Definition at line 117 of file WalkMC.cc.

References `checksum()`, `save_file()`, and `wp`.

7.180.4.18 `void WalkMC::setAngle (double a) [inline]`

sets [WalkParam::body_angle](#) of `wp`

Definition at line 110 of file WalkMC.h.

References `WalkMC::WalkParam::body_angle`, and `wp`.

7.180.4.19 void WalkMC::setHeight (double *h*) [inline]

sets [WalkParam::body_height](#) of [wp](#)

Definition at line 108 of file WalkMC.h.

References [WalkMC::WalkParam::body_height](#), and [wp](#).

7.180.4.20 void WalkMC::setHop (double *h*) [inline]

sets [WalkParam::hop](#) of [wp](#)

Definition at line 112 of file WalkMC.h.

References [WalkMC::WalkParam::hop](#), and [wp](#).

7.180.4.21 void WalkMC::setPaused (bool *p*) [inline]

if set to true, will stop moving

Definition at line 106 of file WalkMC.h.

References [isPaused](#).

7.180.4.22 void WalkMC::setPeriod (long *p*) [inline]

sets [WalkParam::period](#) of [wp](#)

Definition at line 116 of file WalkMC.h.

References [WalkMC::WalkParam::period](#), and [wp](#).

7.180.4.23 void WalkMC::setSway (double *h*) [inline]

sets [WalkParam::sway](#) of [wp](#)

Definition at line 114 of file WalkMC.h.

References [WalkMC::WalkParam::sway](#), and [wp](#).

7.180.4.24 void WalkMC::setTargetVelocity (double *dx*, double *dy*, double *da*)

set the direction to walk - can specify x (forward), y (left), and angular (counterclockwise) velocities

Definition at line 123 of file WalkMC.cc.

References `bound()`, `da`, `dx`, `get_time()`, `MotionManagerMsg::getID()`, `MotionCommand::isActive()`, `GVector::vector2d< num >::length()`, `EventBase::locomotionEGID`, `MAX_DA`, `MAX_DX`, `MAX_DY`, `MotionCommand::postEvent()`, `GVector::vector3d< double >::set()`, `LocomotionEvent::setXYA()`, `EventBase::statusETID`, `target_vel_xya`, `travelTime`, `GVector::vector2d< num >::x`, and `GVector::vector2d< num >::y`.

7.180.4.25 `int WalkMC::updateOutputs ()` [virtual]

calculates current positions of the paws

Implements [MotionCommand](#).

Definition at line 148 of file `WalkMC.cc`.

References `WalkMC::LegWalkState::air`, `WalkMC::LegWalkState::airpath`, `BodyPosition::angle`, `angle_delta`, `WalkMC::WalkParam::body_angle`, `WalkMC::WalkParam::body_height`, `bound()`, `cmds`, `HermiteSplineSegment< vector3d, double >::create()`, `WalkMC::LegParam::down_time`, `WalkMC::LegParam::down_vel`, `HermiteSplineSegment< vector3d, double >::eval()`, `get_time()`, `GetLegAngles()`, `WalkMC::WalkParam::hop`, `isDirty()`, `ERS210Info::JointsPerLeg`, `WalkMC::WalkParam::leg`, `ERS210Info::LegOffset`, `legpos`, `legw`, `WalkMC::LegParam::lift_time`, `WalkMC::LegParam::lift_vel`, `BodyPosition::loc`, `max_accel_xya`, `motman`, `WalkMC::LegParam::neutral`, `ERS210Info::NumFrames`, `ERS210Info::NumLegJoints`, `ERS210Info::NumLegs`, `WalkMC::WalkParam::period`, `pos_delta`, `GVector::vector3d< double >::rotate_z()`, `GVector::vector3d< double >::set()`, `MotionManager::setOutput()`, `WalkMC::WalkParam::sway`, `target_vel_xya`, `time`, `TimeStep`, `vel_xya`, `wp`, `GVector::vector3d< double >::x`, `GVector::vector3d< double >::y`, and `GVector::vector3d< double >::z`.

7.180.5 Member Data Documentation

7.180.5.1 `double WalkMC::angle_delta` [protected]

how much we've turned

Definition at line 144 of file `WalkMC.h`.

7.180.5.2 `splinepath WalkMC::body_angle` [protected]

the path the body goes through while walking (?)

Definition at line 141 of file `WalkMC.h`.

7.180.5.3 [splinepath WalkMC::body_loc](#) [protected]

the path the body goes through while walking (?)

Definition at line 140 of file WalkMC.h.

7.180.5.4 [OutputCmd WalkMC::cmds\[NumOutputs\]\[NumFrames\]](#)
[protected]

holds current joint commands

Definition at line 129 of file WalkMC.h.

7.180.5.5 [bool WalkMC::isPaused](#) [protected]

true if we are paused

Definition at line 135 of file WalkMC.h.

7.180.5.6 [vector3d WalkMC::legpos\[NumLegs\]](#) [protected]

current position of each leg

Definition at line 139 of file WalkMC.h.

7.180.5.7 [LegWalkState WalkMC::legw\[NumLegs\]](#) [protected]

current state of each leg

Definition at line 138 of file WalkMC.h.

7.180.5.8 [const vector3d WalkMC::max_accel_xya](#) [static]

vector version of MAX_DX,MAX_DY,MAX_DA

7.180.5.9 [const float WalkMC::MAX_DA = 1.8](#) [static]

==1.8 rad/sec

Definition at line 52 of file WalkMC.cc.

7.180.5.10 [const float WalkMC::MAX_DX = 180](#) [static]

==180 mm/sec

Definition at line 50 of file WalkMC.cc.

7.180.5.11 `const float WalkMC::MAX_DY = 140` [static]

==140 mm/sec

Definition at line 51 of file WalkMC.cc.

7.180.5.12 `vector3d WalkMC::pos_delta` [protected]

how much we've moved

Definition at line 143 of file WalkMC.h.

7.180.5.13 `vector3d WalkMC::target_vel_xya` [protected]

the velocity that was requested

Definition at line 151 of file WalkMC.h.

7.180.5.14 `int WalkMC::time` [protected]

time of last call to updateJointCmds()

Definition at line 147 of file WalkMC.h.

7.180.5.15 `int WalkMC::TimeStep` [protected]

time to pretend passes between each call to updateJointCmds() - usually Robot-Info::FrameTime, unless you want to make it walk in slow motion (handy sometimes for debugging)

Definition at line 148 of file WalkMC.h.

7.180.5.16 `unsigned int WalkMC::travelTime` [protected]

the time since the last call to setTargetVelocity - handy to check the time we've been traveling current vector

Definition at line 146 of file WalkMC.h.

7.180.5.17 `vector3d WalkMC::vel_xya` [protected]

the current velocity we're moving

Definition at line 150 of file WalkMC.h.

7.180.5.18 [WalkParam WalkMC::wp](#) [protected]

current walking parameters (note that it's not static - different WalkMC's can have different setting, handy...

Definition at line 137 of file WalkMC.h.

The documentation for this class was generated from the following files:

- [WalkMC.h](#)
- [WalkMC.cc](#)

7.181 WalkMC::LegParam Struct Reference

```
#include <WalkMC.h>
```

7.181.1 Detailed Description

holds parameters about how to move each leg

Definition at line 62 of file WalkMC.h.

Public Member Functions

- [LegParam \(\)](#)
constructor

Public Attributes

- [vector3d neutral](#)
defines the "neutral" point of each leg - where it is in midstep
- [vector3d lift_vel](#)
give the velocities to use when raising the paw
- [vector3d down_vel](#)
give the velocities to use when lowering the paw
- double [lift_time](#)
the time (as percentage of [WalkParam::period](#)) in the cycle to lift (so you can set different offsets between the paws)
- double [down_time](#)
the time (as percentage of [WalkParam::period](#)) in the cycle to put down (so you can set different offsets between the paws)

7.181.2 Constructor & Destructor Documentation

7.181.2.1 WalkMC::LegParam::LegParam () [inline]

constructor

Definition at line 63 of file WalkMC.h.

References `down_time`, `down_vel`, `lift_time`, `lift_vel`, and `neutral`.

7.181.3 Member Data Documentation

7.181.3.1 `double WalkMC::LegParam::down_time`

the time (as percentage of `WalkParam::period`) in the cycle to put down (so you can set different offsets between the paws)

Definition at line 68 of file WalkMC.h.

7.181.3.2 `vector3d WalkMC::LegParam::down_vel`

give the velocities to use when lowering the paw

Definition at line 66 of file WalkMC.h.

7.181.3.3 `double WalkMC::LegParam::lift_time`

the time (as percentage of `WalkParam::period`) in the cycle to lift (so you can set different offsets between the paws)

Definition at line 67 of file WalkMC.h.

7.181.3.4 `vector3d WalkMC::LegParam::lift_vel`

give the velocities to use when raising the paw

Definition at line 65 of file WalkMC.h.

7.181.3.5 `vector3d WalkMC::LegParam::neutral`

defines the "neutral" point of each leg - where it is in midstep

Definition at line 64 of file WalkMC.h.

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

- [WalkMC.h](#)

7.182 WalkMC::LegWalkState Struct Reference

```
#include <WalkMC.h>
```

7.182.1 Detailed Description

holds state about each leg's path

Definition at line 55 of file WalkMC.h.

Public Member Functions

- [LegWalkState \(\)](#)
constructor

Public Attributes

- [spline airpath](#)
the path to follow
- [bool air](#)
true if in the air

7.182.2 Constructor & Destructor Documentation

7.182.2.1 WalkMC::LegWalkState::LegWalkState () [inline]

constructor

Definition at line 56 of file WalkMC.h.

References [air](#), and [airpath](#).

7.182.3 Member Data Documentation

7.182.3.1 [bool WalkMC::LegWalkState::air](#)

true if in the air

Definition at line 58 of file WalkMC.h.

7.182.3.2 [spline WalkMC::LegWalkState::airpath](#)

the path to follow

Definition at line 57 of file WalkMC.h.

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

- [WalkMC.h](#)

7.183 WalkMC::WalkParam Struct Reference

```
#include <WalkMC.h>
```

7.183.1 Detailed Description

holds more general parameters about the walk

Definition at line 72 of file WalkMC.h.

Public Member Functions

- [WalkParam](#) ()

constructor

Public Attributes

- [LegParam](#) leg [4]
a set of LegParam's, one for each leg
- double [body_height](#)
the height to hold the body (mm)
- double [body_angle](#)
the angle to hold the body (rad - 0 is level)
- double [hop](#)
sinusoidal hop amplitude
- double [sway](#)
sinusoidal sway in y direction
- long [period](#)
the time between steps
- long [reserved](#)
live with it

7.183.2 Constructor & Destructor Documentation

7.183.2.1 WalkMC::WalkParam::WalkParam () [inline]

constructor

Definition at line 73 of file WalkMC.h.

References `body_angle`, `body_height`, `hop`, `period`, `reserved`, and `sway`.

7.183.3 Member Data Documentation

7.183.3.1 double WalkMC::WalkParam::body_angle

the angle to hold the body (rad - 0 is level)

Definition at line 76 of file WalkMC.h.

7.183.3.2 double WalkMC::WalkParam::body_height

the height to hold the body (mm)

Definition at line 75 of file WalkMC.h.

7.183.3.3 double WalkMC::WalkParam::hop

sinusoidal hop amplitude

Definition at line 77 of file WalkMC.h.

7.183.3.4 LegParam WalkMC::WalkParam::leg[4]

a set of LegParam's, one for each leg

Definition at line 74 of file WalkMC.h.

7.183.3.5 long WalkMC::WalkParam::period

the time between steps

Definition at line 79 of file WalkMC.h.

7.183.3.6 long WalkMC::WalkParam::reserved

live with it

Definition at line 80 of file WalkMC.h.

7.183.3.7 **double** [WalkMC::WalkParam::sway](#)

sinusoidal sway in y direction

Definition at line 78 of file WalkMC.h.

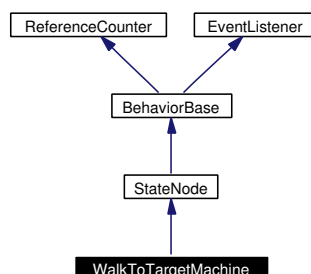
The documentation for this struct was generated from the following file:

- [WalkMC.h](#)

7.184 WalkToTargetMachine Class Reference

```
#include <WalkToTargetMachine.h>
```

Inheritance diagram for WalkToTargetMachine:



7.184.1 Detailed Description

a state machine for walking towards a visual target

Definition at line 11 of file WalkToTargetMachine.h.

Public Member Functions

- **WalkToTargetMachine** ([VisionEventNS::VisionSourceID_t](#) obj, [StateNode](#) *c=NULL, [StateNode](#) *l=NULL, [StateNode](#) *p=NULL)
constructor, pass parent, success and failure nodes, and VisionSourceID_t
- void **setup** ()
This is called by [DoStart\(\)](#) when you should setup the network.
- virtual void **DoStart** ()
Transitions should call this when you are entering the state, so it can enable its transitions.
- virtual void **DoStop** ()
Transitions should call this when you are leaving the state, so it can disable its transitions.
- virtual void **processEvent** (const [EventBase](#) &)
Doesn't do anything, supplied here so you don't have to (since events will probably be going to Transition's, not here).

Static Public Member Functions

- `std::string getClassDescription ()`

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Protected Attributes

- `VisionEventNS::VisionSourceID_t tracking`

the object being tracked

- `TimeOutTrans * timeout`

transition in case we lose the ball

- `StateNode * close`

dest if we get close to the object

- `StateNode * lost`

dest if we get lost

- `MotionManager::MC_ID walker_id`

so we can walk

- `MotionManager::MC_ID headpointer_id`

so we can point the head at the object

Private Member Functions

- `WalkToTargetMachine (const WalkToTargetMachine &)`

don't call this

- `WalkToTargetMachine operator= (const WalkToTargetMachine &)`

don't call this

7.184.2 Constructor & Destructor Documentation

7.184.2.1 WalkToTargetMachine::WalkToTargetMachine ([VisionEventNS::VisionSourceID_t](#) *obj*, [StateNode](#) * *c* = NULL, [StateNode](#) * *l* = NULL, [StateNode](#) * *p* = NULL) [inline]

constructor, pass parent, success and failure nodes, and VisionSourceID_t

Definition at line 14 of file WalkToTargetMachine.h.

References `close`, `headpointer_id`, `lost`, `timeout`, `tracking`, and `walker_id`.

7.184.2.2 WalkToTargetMachine::WalkToTargetMachine (const [WalkToTargetMachine](#) &) [private]

don't call this

7.184.3 Member Function Documentation

7.184.3.1 void WalkToTargetMachine::DoStart () [virtual]

Transitions should call this when you are entering the state, so it can enable its transitions.

Reimplemented from [StateNode](#).

Definition at line 18 of file WalkToTargetMachine.cc.

References `EventRouter::addListener()`, `MotionManager::addMotion()`, `StateNode::DoStart()`, `Vision::enableEvents()`, `erouter`, `headpointer_id`, `motman`, `tracking`, `vision`, `EventBase::visionEGID`, and `walker_id`.

7.184.3.2 void WalkToTargetMachine::DoStop () [virtual]

Transitions should call this when you are leaving the state, so it can disable its transitions.

Reimplemented from [StateNode](#).

Definition at line 26 of file WalkToTargetMachine.cc.

References `Vision::disableEvents()`, `StateNode::DoStop()`, `erouter`, `EventRouter::forgetListener()`, `headpointer_id`, `motman`, `MotionManager::removeMotion()`, `tracking`, `vision`, and `walker_id`.

7.184.3.3 `std::string WalkToTargetMachine::getClassDescription ()` [inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 24 of file WalkToTargetMachine.h.

7.184.3.4 [WalkToTargetMachine](#) `WalkToTargetMachine::operator= (const WalkToTargetMachine &)` [private]

don't call this

7.184.3.5 `void WalkToTargetMachine::processEvent (const EventBase & event)` [virtual]

Doesn't do anything, supplied here so you don't have to (since events will probably be going to Transition's, not here).

Reimplemented from [StateNode](#).

Definition at line 35 of file WalkToTargetMachine.cc.

References [MotionManager::checkinMotion\(\)](#), [MotionManager::checkoutMotion\(\)](#), [DtoR\(\)](#), [VisionEvent::getCenterX\(\)](#), [VisionEvent::getCenterY\(\)](#), [EventBase::getTypeID\(\)](#), [ERS210Info::HeadOffset](#), [headpointer_id](#), [motman](#), [WorldState::outputs](#), [ERS210Info::PanOffset](#), [TimeOutTrans::resetTimer\(\)](#), [HeadPointerMC::setJoints\(\)](#), [WalkMC::setTargetVelocity\(\)](#), [state](#), [EventBase::statusETID](#), [ERS210Info::TiltOffset](#), [timeout](#), and [walker_id](#).

7.184.3.6 `void WalkToTargetMachine::setup ()` [virtual]

This is called by [DoStart\(\)](#) when you should setup the network.

Reimplemented from [StateNode](#).

Definition at line 10 of file WalkToTargetMachine.cc.

References [StateNode::addTransition\(\)](#), [close](#), [lost](#), [timeout](#), and [tracking](#).

7.184.4 Member Data Documentation

7.184.4.1 [StateNode*](#) [WalkToTargetMachine::close](#) [protected]

dest if we get close to the object

Definition at line 33 of file WalkToTargetMachine.h.

7.184.4.2 [MotionManager::MC_ID WalkToTargetMachine::headpointer_id](#)
[protected]

so we can point the head at the object

Definition at line 36 of file WalkToTargetMachine.h.

7.184.4.3 [StateNode* WalkToTargetMachine::lost](#) [protected]

dest if we get lost

Definition at line 34 of file WalkToTargetMachine.h.

7.184.4.4 [TimeOutTrans* WalkToTargetMachine::timeout](#) [protected]

transition in case we lose the ball

Definition at line 32 of file WalkToTargetMachine.h.

7.184.4.5 [VisionEventNS::VisionSourceID_t WalkToTargetMachine::tracking](#)
[protected]

the object being tracked

Definition at line 31 of file WalkToTargetMachine.h.

7.184.4.6 [MotionManager::MC_ID WalkToTargetMachine::walker_id](#)
[protected]

so we can walk

Definition at line 35 of file WalkToTargetMachine.h.

The documentation for this class was generated from the following files:

- [WalkToTargetMachine.h](#)
- [WalkToTargetMachine.cc](#)

7.185 WAV Class Reference

```
#include <WAV.h>
```

Public Member Functions

- [WAV \(\)](#)
- [WAV \(byte *addr\)](#)
- [~WAV \(\)](#)
- [WAVErrror Set \(byte *addr\)](#)
- [WAVErrror CopyTo \(OSoundVectorData *data\)](#)
- [WAVErrror Rewind \(\)](#)
- unsigned int [GetSamplingRate \(\)](#)
- unsigned int [GetBitsPerSample \(\)](#)
- size_t [GetSoundUnitSize \(\)](#)
- byte * [GetDataStart \(\)](#)
- byte * [GetDataEnd \(\)](#)

Private Member Functions

- longword [get_longword](#) (byte *addr)
- word [get_word](#) (byte *addr)

Private Attributes

- OSoundInfo [soundInfo](#)
- size_t [soundUnitSize](#)
- byte * [dataStart](#)
- byte * [dataEnd](#)
- byte * [dataCurrent](#)

Static Private Attributes

- const size_t [MONO8K8B_UNIT_SIZE](#) = 256
- const size_t [MONO16K16B_UNIT_SIZE](#) = 1024
- const size_t [FMTSIZE_WITHOUT_EXTINFO](#) = 16

7.185.1 Constructor & Destructor Documentation

7.185.1.1 WAV::WAV ()

Definition at line 28 of file WAV.cc.

7.185.1.2 WAV::WAV (byte * *addr*)

Definition at line 32 of file WAV.cc.

References Set().

7.185.1.3 WAV::~~WAV () [inline]

Definition at line 45 of file WAV.h.

7.185.2 Member Function Documentation

7.185.2.1 **WAVEError** WAV::CopyTo (OSoundVectorData * *data*)

Definition at line 169 of file WAV.cc.

References dataCurrent, dataEnd, MONO8K8B_UNIT_SIZE, soundInfo, soundUnitSize, WAV_FAIL, WAV_SIZE_NOT_ENOUGH, and WAV_SUCCESS.

7.185.2.2 longword WAV::get_longword (byte * *addr*) [private]

Definition at line 232 of file WAV.cc.

7.185.2.3 word WAV::get_word (byte * *addr*) [private]

Definition at line 242 of file WAV.cc.

7.185.2.4 unsigned int WAV::GetBitsPerSample () [inline]

Definition at line 52 of file WAV.h.

References soundInfo.

7.185.2.5 byte* WAV::GetDataEnd () [inline]

Definition at line 56 of file WAV.h.

References dataEnd.

7.185.2.6 byte* WAV::GetDataStart () [inline]

Definition at line 55 of file WAV.h.

References `dataStart`.

7.185.2.7 `unsigned int WAV::GetSamplingRate ()` [inline]

Definition at line 51 of file `WAV.h`.

References `soundInfo`.

7.185.2.8 `size_t WAV::GetSoundUnitSize ()` [inline]

Definition at line 53 of file `WAV.h`.

References `soundUnitSize`.

7.185.2.9 `WAVEError WAV::Rewind ()`

Definition at line 225 of file `WAV.cc`.

References `dataCurrent`, `dataStart`, and `WAV_SUCCESS`.

7.185.2.10 `WAVEError WAV::Set (byte * addr)`

Definition at line 38 of file `WAV.cc`.

References `dataCurrent`, `dataEnd`, `dataStart`, `FMTSIZE_WITHOUT_EXTINFO`, `get_longword()`, `get_word()`, `MONO16K16B_UNIT_SIZE`, `MONO8K8B_UNIT_SIZE`, `soundInfo`, `soundUnitSize`, `WAV_BITSPERSAMPLE_NOT_SUPPORTED`, `WAV_CHANNEL_NOT_SUPPORTED`, `WAV_FORMAT_NOT_SUPPORTED`, `WAV_NOT_RIFF`, `WAV_NOT_WAV`, `WAV_SAMPLINGRATE_NOT_SUPPORTED`, and `WAV_SUCCESS`.

7.185.3 Member Data Documentation

7.185.3.1 `byte* WAV::dataCurrent` [private]

Definition at line 74 of file `WAV.h`.

7.185.3.2 `byte* WAV::dataEnd` [private]

Definition at line 73 of file `WAV.h`.

7.185.3.3 `byte* WAV::dataStart` [private]

Definition at line 72 of file WAV.h.

7.185.3.4 `const size_t WAV::FMTSIZE_WITHOUT_EXTINFO = 16`
[static, private]

Definition at line 68 of file WAV.h.

7.185.3.5 `const size_t WAV::MONO16K16B_UNIT_SIZE = 1024` [static,
private]

Definition at line 66 of file WAV.h.

7.185.3.6 `const size_t WAV::MONO8K8B_UNIT_SIZE = 256` [static,
private]

Definition at line 63 of file WAV.h.

7.185.3.7 `OSoundInfo WAV::soundInfo` [private]

Definition at line 70 of file WAV.h.

7.185.3.8 `size_t WAV::soundUnitSize` [private]

Definition at line 71 of file WAV.h.

The documentation for this class was generated from the following files:

- [WAV.h](#)
- [WAV.cc](#)

7.186 Wireless Class Reference

```
#include <Wireless.h>
```

7.186.1 Detailed Description

Tekkotsu wireless class.

For more information on using wireless, please read the following tutorials:

- [TekkotsuMon](#)
- [TCP/IP](#)
- [Remote Processing OPENR](#) Tekkotsu Wireless and Remote Processing OPENR provide different interfaces to comparable wireless functionality.

Definition at line 23 of file Wireless.h.

Public Member Functions

- [Wireless](#) ()
constructor - only one wireless object is required per Aperios process.
- [~Wireless](#) ()
destructor
- int [listen](#) (int sock, int port)
The socket waits for incoming connections.
- int [connect](#) (int sock, const char *ipaddr, int port)
The socket tries to connect to a specific.
- void [setReceiver](#) (int sock, int(*rcvcbckfn)(char *, int))
sets receiver callback for a socket
- void [close](#) (int sock)
closes a socket
- [Socket](#) * [socket](#) (TransportType_t ttype)
Creates a new socket.

- `Socket * socket (TransportType_t ttype, int recvsize, int sendsize)`
- `bool isConnected (int sock)`
utility function that you can use if you're curious about the state of the socket.
- `bool isReady (int sock)`
utility function that you can use if you're curious about the state of the socket.
- `bool hasData (int sock)`
utility function that you can use if you're curious about the state of the socket.
- `void setReceiver (Socket &sobj, int(*rcvcbckfn)(char *, int))`
helper function for the function with the same name that takes a socket descriptor (int)
- `void setReceiver (Socket *sobj, int(*rcvcbckfn)(char *, int))`
helper function for the function with the same name that takes a socket descriptor (int)
- `int listen (Socket &sobj, int port)`
helper function for the function with the same name that takes a socket descriptor (int)
- `int listen (Socket *sobj, int port)`
helper function for the function with the same name that takes a socket descriptor (int)
- `int connect (Socket &sobj, const char *ipaddr, int port)`
helper function for the function with the same name that takes a socket descriptor (int)
- `int connect (Socket *sobj, const char *ipaddr, int port)`
helper function for the function with the same name that takes a socket descriptor (int)
- `void close (Socket &sobj)`
helper function for the function with the same name that takes a socket descriptor (int)
- `void close (Socket *sobj)`
helper function for the function with the same name that takes a socket descriptor (int)
- `void receive (int sock, int(*rcvcbckfn)(char *, int))`
function for internal and `Socket` use. You should not call this
- `void receive (int sock)`
function for internal and `Socket` use. You should not call this

- void [send](#) (int sock)
- void [blockingSend](#) (int sock)

- void [ListenCont](#) (void *msg)
- void [BindCont](#) (void *msg)
- void [ConnectCont](#) (void *msg)
- void [SendCont](#) (void *msg)
- void [ReceiveCont](#) (void *msg)
- void [CloseCont](#) (void *msg)

Static Public Attributes

- const int [WIRELESS_MAX_SOCKETS](#) = 100
Maximum number of sockets which can be created.

- const int [WIRELESS_DEF_RECV_SIZE](#) = 1024
Default number of bytes to use for receive buffers (overridden by value passed to [socket\(\)](#)).

- const int [WIRELESS_DEF_SEND_SIZE](#) = 1024
Default number of bytes to use for send buffers (overridden by value passed to [socket\(\)](#)).

Private Member Functions

- [Wireless](#) (const [Wireless](#) &)
don't call

- [Wireless](#) & operator= (const [Wireless](#) &)
don't call

Private Attributes

- antStackRef [ipstackRef](#)
private ALOKL_TODO

- OID [myOID](#)

private ALOKL_TODO

- [Socket](#) * [sockets](#) [[WIRELESS_MAX_SOCKETS](#)]

private ALOKL_TODO

- int [sock_num](#)

private ALOKL_TODO

7.186.2 Constructor & Destructor Documentation

7.186.2.1 Wireless::Wireless ()

constructor - only one wireless object is required per Aperios process.

[MMCombo](#) already creates one. The (global) instance is called wireless, and you can access it by including Wireless/Wireless.h (this file) in your code

Definition at line 17 of file Wireless.cc.

References [ipstackRef](#), [myOID](#), [sockets](#), and [WIRELESS_MAX_SOCKETS](#).

7.186.2.2 Wireless::~~Wireless ()

destructor

Definition at line 28 of file Wireless.cc.

7.186.2.3 Wireless::Wireless (const [Wireless](#) &) [private]

don't call

7.186.3 Member Function Documentation

7.186.3.1 void Wireless::BindCont (void * *msg*)

callback function for communicating with Aperios Networking Toolkit. You should not call this.

Definition at line 191 of file Wireless.cc.

References [SocketNS::CONNECTION_CONNECTED](#), [SocketNS::CONNECTION_ERROR](#), [sockets](#), and [Socket::state](#).

7.186.3.2 void Wireless::blockingSend (int sock)

function called by the [Socket](#) objects to actually write data to the network. You should not call this.

Definition at line 239 of file Wireless.cc.

References [SocketNS::CONNECTION_CONNECTED](#), [ipstackRef](#), [Socket::sendSize](#), [sock_num](#), [sockets](#), [Socket::state](#), and [Socket::tx](#).

7.186.3.3 void Wireless::close (Socket * subj) [inline]

helper function for the function with the same name that takes a socket descriptor (int)

Definition at line 92 of file Wireless.h.

References [close\(\)](#), and [Socket::sock](#).

7.186.3.4 void Wireless::close (Socket & subj) [inline]

helper function for the function with the same name that takes a socket descriptor (int)

Definition at line 91 of file Wireless.h.

References [close\(\)](#), and [Socket::sock](#).

7.186.3.5 void Wireless::close (int sock)

closes a socket

Definition at line 319 of file Wireless.cc.

References [SocketNS::CONNECTION_CLOSED](#), [SocketNS::CONNECTION_CLOSING](#), [ipstackRef](#), [myOID](#), [sockets](#), and [Socket::state](#).

7.186.3.6 void Wireless::CloseCont (void * msg)

callback function for communicating with Aperios Networking Toolkit. You should not call this.

Definition at line 334 of file Wireless.cc.

References [SocketNS::CONNECTION_CLOSED](#), [listen\(\)](#), [Socket::server_port](#), [sockets](#), and [Socket::state](#).

7.186.3.7 `int Wireless::connect (Socket * sobj, const char * ipaddr, int port)`
[inline]

helper function for the function with the same name that takes a socket descriptor (int)

Definition at line 89 of file Wireless.h.

References connect(), and Socket::sock.

7.186.3.8 `int Wireless::connect (Socket & sobj, const char * ipaddr, int port)`
[inline]

helper function for the function with the same name that takes a socket descriptor (int)

Definition at line 87 of file Wireless.h.

References connect(), and Socket::sock.

7.186.3.9 `int Wireless::connect (int sock, const char * ipaddr, int port)`

The socket tries to connect to a specific.

Definition at line 126 of file Wireless.cc.

References SocketNS::CONNECTION_CLOSED, SocketNS::CONNECTION_CONNECTING, Socket::endpoint, Socket::init(), ipstackRef, myOID, SocketNS::SOCK_DGRAM, sock_num, SocketNS::SOCK_STREAM, sockets, Socket::state, and Socket::trType.

7.186.3.10 `void Wireless::ConnectCont (void * msg)`

callback function for communicating with Aperios Networking Toolkit. You should not call this.

Definition at line 177 of file Wireless.cc.

References SocketNS::CONNECTION_CONNECTED, SocketNS::CONNECTION_ERROR, sockets, and Socket::state.

7.186.3.11 `bool Wireless::hasData (int sock)` [inline]

utility function that you can use if you're curious about the state of the socket.

You shouldn't need to use it, since asking sockets for write and read buffers does the necessary sanity checks

Definition at line 76 of file Wireless.h.

References Socket::rx, and sockets.

7.186.3.12 bool Wireless::isConnected (int *sock*) [inline]

utility function that you can use if you're curious about the state of the socket.

You shouldn't need to use it, since asking sockets for write and read buffers does the necessary sanity checks

Definition at line 73 of file Wireless.h.

References SocketNS::CONNECTION_CONNECTED, sockets, and Socket::state.

7.186.3.13 bool Wireless::isReady (int *sock*) [inline]

utility function that you can use if you're curious about the state of the socket.

You shouldn't need to use it, since asking sockets for write and read buffers does the necessary sanity checks

Definition at line 75 of file Wireless.h.

References sockets, and Socket::tx.

7.186.3.14 int Wireless::listen (Socket * *subj*, int *port*) [inline]

helper function for the function with the same name that takes a socket descriptor (int)

Definition at line 86 of file Wireless.h.

References listen(), and Socket::sock.

7.186.3.15 int Wireless::listen (Socket & *subj*, int *port*) [inline]

helper function for the function with the same name that takes a socket descriptor (int)

Definition at line 85 of file Wireless.h.

References listen(), and Socket::sock.

7.186.3.16 int Wireless::listen (int *sock*, int *port*)

The socket waits for incoming connections.

That is, it acts like a server. If a connection is established and later broken, it resumes waiting for new connections.

Definition at line 74 of file Wireless.cc.

References SocketNS::CONNECTION_CLOSED, SocketNS::CONNECTION_LISTENING, Socket::endpoint, Socket::init(), ipstackRef, myOID, Socket::server-

port, SocketNS::SOCK_DGRAM, sock_num, SocketNS::SOCK_STREAM, sockets, Socket::state, and Socket::trType.

7.186.3.17 void Wireless::ListenCont (void * msg)

callback function for communicating with Aperios Networking Toolkit. You should not call this.

Definition at line 161 of file Wireless.cc.

References SocketNS::CONNECTION_CONNECTED, SocketNS::CONNECTION_ERROR, Socket::rcvcbckfn, receive(), sockets, and Socket::state.

7.186.3.18 Wireless& Wireless::operator= (const Wireless &) [private]

don't call

7.186.3.19 void Wireless::receive (int sock)

function for internal and Socket use. You should not call this

Definition at line 266 of file Wireless.cc.

References SocketNS::CONNECTION_CONNECTED, ipstackRef, myOID, Socket::rx, sock_num, sockets, and Socket::state.

7.186.3.20 void Wireless::receive (int sock, int(* rcvcbckfn)(char *, int))

function for internal and Socket use. You should not call this

Definition at line 283 of file Wireless.cc.

References SocketNS::CONNECTION_CONNECTED, ipstackRef, myOID, Socket::rcvcbckfn, Socket::rx, sock_num, sockets, and Socket::state.

7.186.3.21 void Wireless::ReceiveCont (void * msg)

callback function for communicating with Aperios Networking Toolkit. You should not call this.

Definition at line 301 of file Wireless.cc.

References close(), SocketNS::CONNECTION_ERROR, Socket::rcvcbckfn, receive(), Socket::recvSize, sockets, and Socket::state.

7.186.3.22 void Wireless::send (int sock)

function called by the [Socket](#) objects to actually write data to the network. You should not call this.

Definition at line 205 of file Wireless.cc.

References [SocketNS::CONNECTION_CONNECTED](#), [ipstackRef](#), [myOID](#), [Socket::sendSize](#), [sock_num](#), [sockets](#), [Socket::state](#), and [Socket::tx](#).

7.186.3.23 void Wireless::SendCont (void * msg)

callback function for communicating with Aperios Networking Toolkit. You should not call this.

Definition at line 224 of file Wireless.cc.

References [close\(\)](#), [SocketNS::CONNECTION_ERROR](#), [Socket::flush\(\)](#), [sockets](#), [Socket::state](#), and [Socket::tx](#).

7.186.3.24 void Wireless::setReceiver ([Socket](#) * sobj, int(* rcvcbckfn)(char *, int)) [inline]

helper function for the function with the same name that takes a socket descriptor (int)

Definition at line 83 of file Wireless.h.

References [setReceiver\(\)](#), and [Socket::sock](#).

7.186.3.25 void Wireless::setReceiver ([Socket](#) & sobj, int(* rcvcbckfn)(char *, int)) [inline]

helper function for the function with the same name that takes a socket descriptor (int)

Definition at line 81 of file Wireless.h.

References [setReceiver\(\)](#), and [Socket::sock](#).

7.186.3.26 void Wireless::setReceiver (int sock, int(* rcvcbckfn)(char *, int))

sets receiver callback for a socket

Definition at line 258 of file Wireless.cc.

References [Socket::rcvcbckfn](#), [sock_num](#), and [sockets](#).

7.186.3.27 [Socket](#) * [Wireless::socket](#) ([TransportType.t ttype](#), int *recvsize*, int *sendsize*)

Parameters:

ttype selects between TCP and UDP

recvsize size of input buffer

sendsize size of output buffer

Definition at line 38 of file [Wireless.cc](#).

References [ipstackRef](#), [sock_num](#), [sockets](#), and [WIRELESS_MAX_SOCKETS](#).

7.186.3.28 [Socket](#) * [Wireless::socket](#) ([TransportType.t ttype](#))

Creates a new socket.

Returns:

pointer to [Socket](#) object created

Parameters:

ttype selects between TCP and UDP

See also:

[WIRELESS_DEF_RECV_SIZE](#), [WIRELESS_DEF_SEND_SIZE](#)

Definition at line 33 of file [Wireless.cc](#).

References [WIRELESS_DEF_RECV_SIZE](#), and [WIRELESS_DEF_SEND_SIZE](#).

7.186.4 Member Data Documentation

7.186.4.1 antStackRef [Wireless::ipstackRef](#) [private]

private ALOKL_TODO

Definition at line 123 of file [Wireless.h](#).

7.186.4.2 OID [Wireless::myOID](#) [private]

private ALOKL_TODO

Definition at line 124 of file [Wireless.h](#).

7.186.4.3 `int Wireless::sock_num` `[private]`

private ALOKL_TODO

Definition at line 126 of file Wireless.h.

7.186.4.4 `Socket* Wireless::sockets[WIRELESS_MAX_SOCKETS]`
`[private]`

private ALOKL_TODO

Definition at line 125 of file Wireless.h.

7.186.4.5 `const int Wireless::WIRELESS_DEF_RECV_SIZE = 1024`
`[static]`

Default number of bytes to use for receive buffers (overridden by value passed to [socket\(\)](#)).

Definition at line 29 of file Wireless.h.

7.186.4.6 `const int Wireless::WIRELESS_DEF_SEND_SIZE = 1024`
`[static]`

Default number of bytes to use for send buffers (overridden by value passed to [socket\(\)](#)).

Definition at line 32 of file Wireless.h.

7.186.4.7 `const int Wireless::WIRELESS_MAX_SOCKETS = 100` `[static]`

Maximum number of sockets which can be created.

Definition at line 26 of file Wireless.h.

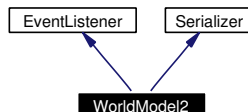
The documentation for this class was generated from the following files:

- [Wireless.h](#)
- [Wireless.cc](#)

7.187 WorldModel2 Class Reference

```
#include <WorldModel2.h>
```

Inheritance diagram for WorldModel2:



7.187.1 Detailed Description

Tekkotsu's localization and mapping system (**IN ACTIVE DEVELOPMENT**).

WorldModel2 is a system of egocentric and allocentric maps designed to represent the world around the AIBO. Presently, WorldModel2 is composed of **beta** and **alpha** software, with most egocentric functionality in beta and everything else in alpha.

The bulk of the alpha software in WorldModel2 is involved in global mapping and localization. This software is disabled by default and therefore escapes documentation by Doxygen. For more information on these routines, read [WorldModel2.h](#); to give them a try (hardly advisable) set the environment variable GLOBAL_MAP to -DWANT_-GLOBAL_MAP before compiling.

Definition at line 120 of file WorldModel2.h.

Public Member Functions

- [WorldModel2](#) ()
Constructor.
- virtual [~WorldModel2](#) ()
Destructor.
- void [resetWorldModel](#) ()
Resets the world model to the start state.
- void [enableKludge](#) (unsigned int kludge)
Enable a kludge (see the [WM2Kludge](#) namespace for details).
- void [disableKludge](#) (unsigned int kludge)
Disable a kludge (see the [WM2Kludge](#) namespace).

- virtual void `processEvent` (const `EventBase` &event)

Process vision and motion events.

- void `getRequests` (`MRvector` &requests)

Get suggestions about what to look at next (ALPHA).

Sensor toggles

Enable or disable certain sensing mechanisms, which are off by default. Disabling sensor mechanisms is useful when you have other CPU intensive things to do and don't want to spend time updating the maps. Note that there is no way to disable processing of `WalkMC` movement—this would lead to gross inconsistencies in the map. Note that the mechanisms all start off blind—you have to turn them on yourself before they'll work.

- void `enableIR` ()

Enable infrared data registration.

- void `disableIR` ()

Disable infrared data registration.

- void `enableGPA` ()

Enable ground plane assumption mapping (ALPHA).

- void `disableGPA` ()

Disable ground plane assumption mapping (ALPHA).

Map data access routines

*Access routines for the SDM and HHM data. You have your choice of "safer" and "faster". First are the "safer" methods, which copies the requested data into arrays furnished by the user. They're not **that** safe, though—you're passing a pointer to your array, and you'd better make certain that your array has enough space for the data.*

You'll probably want to include the header file `WorldModel2/Maps/Configuration.h` to get the defines `DM_CELL_COUNT`, `HM_CELL_COUNT`, and `GM_CELL_COUNT`, which describe the number of cells in the maps. The same file also contains plenty of other constants describing the maps, which may be of use to you.

Safe access routines choose the data you want using "Pickers", functors described and documented in `WorldModel2/Maps/almStructures.h` that you specify as the first argument. Unfortunately, all the data is encoded as float, but this shouldn't create many problems.

*Fast access routines simply return pointers to the active map data itself. **Please do not change anything unless you know what you are doing!***

- void [pickDMData](#) ([dmPicker](#) &p, float *dest)
The safe access method for spherical depth map data.
- [dm_cell](#) * [invadeDMData](#) (void)
The fast access method for spherical depth map data. Use at own risk.
- void [pickHMData](#) ([hmPicker](#) &p, float *dest)
Safe access method for horizontal height map. Do see long description.
- [hm_cell](#) * [invadeHMData](#) (void)
Risky fast access method for horiz. height map. See pickHMData note.

Private Member Functions

- void [processSensors](#) ()
process sensors, incorporating IRs into local maps
- void [processLocomotion](#) (const [EventBase](#) &event)
process locomotion events—movements
- void [processGround](#) ()
Try to do ground plane assumption.
- void [serialize](#) ()
Perform serialization – send data down the line, if needed.
- [WorldModel2](#) (const [WorldModel2](#) &)
dummy functions prevent warnings
- [WorldModel2](#) & [operator=](#) (const [WorldModel2](#) &)
dummy functions prevent warnings

Private Attributes

- unsigned int [kludges](#)
Which kludges are enabled? See [WM2Kludge](#) documentation for info.
- bool [moving](#)
- bool [enabledIR](#)
True if IR sensing is enabled (c.f. [enableIR](#)).

- bool [enabledGPA](#)

True if GPA mapping is enabled (c.f. [enableGPA](#)).

- double [dx](#)

The current velocities of the AIBO.

- double [dy](#)

The current velocities of the AIBO.

- double [da](#)

The current velocities of the AIBO.

- long [movingSince](#)

- [Socket](#) * [sockDM](#)

Here are sockets for serialization of the egocentric map data.

- [Socket](#) * [sockHM](#)

Here are sockets for serialization of the egocentric map data.

Static Private Attributes

- const int [GPAdelay](#) = 100
- const int [SRLdelay](#) = 1000
- bool [locked](#) = false

7.187.2 Constructor & Destructor Documentation

7.187.2.1 WorldModel2::WorldModel2 ()

Constructor.

Definition at line 65 of file WorldModel2.cc.

References [EventRouter::addListener\(\)](#), [EventRouter::addTimer\(\)](#), [AFS_NUM_-LANDMARKS](#), [AFS_NUM_PARTICLES](#), [config](#), [da](#), [DM_CELL_COUNT](#), [Config::worldmodel2_config::dm_port](#), [dx](#), [erouter](#), [Config::worldmodel2_config::fs_port](#), [GM_CELL_COUNT](#), [Config::worldmodel2_config::gm_port](#), [HM_CELL_COUNT](#), [Config::worldmodel2_config::hm_port](#), [Wireless::listen\(\)](#), [locked](#), [EventBase::locomotionEGID](#), [resetWorldModel\(\)](#), [Socket::sock](#), [SocketNS::SOCK_-STREAM](#), [sockDM](#), [sockHM](#), [SRLdelay](#), [TIMER_SID_SRL](#), [wireless](#), and [Config::worldmodel2](#).

7.187.2.2 WorldModel2::~~WorldModel2 () [virtual]

Destructor.

Definition at line 125 of file WorldModel2.cc.

References Wireless::close(), disableGPA(), disableIR(), erouter, EventRouter::forgetListener(), locked, sockDM, sockHM, and wireless.

7.187.2.3 WorldModel2::WorldModel2 (const WorldModel2 &) [private]

dummy functions prevent warnings

7.187.3 Member Function Documentation**7.187.3.1 void WorldModel2::disableGPA ()**

Disable ground plane assumption mapping (ALPHA).

Definition at line 179 of file WorldModel2.cc.

References enabledGPA, erouter, EventRouter::removeTimer(), and TIMER_SID_-GPA.

7.187.3.2 void WorldModel2::disableIR ()

Disable infrared data registration.

Definition at line 157 of file WorldModel2.cc.

References enabledIR, erouter, EventRouter::removeListener(), and EventBase::sensorEGID.

7.187.3.3 void WorldModel2::disableKludge (unsigned int *kludge*)

Disable a kludge (see the [WM2Kludge](#) namespace).

Definition at line 196 of file WorldModel2.cc.

References kludges.

7.187.3.4 void WorldModel2::enableGPA ()

Enable ground plane assumption mapping (ALPHA).

Our AIBO environment has a green floor. We can assume that all green items in the color segmented image from the vision system are parts of the floor. Using triangulation and the assumption that the floor is a flat plane at $z=0$, we can fill in parts of the map instantaneously. Note that ground plane assumption mapping only occurs when the AIBO is standing still in the erect posture from `ms/data/motion/stand.pos` and staring straight ahead (i.e. all head positioning attributes are 0).

Definition at line 169 of file `WorldModel2.cc`.

References `EventRouter::addTimer()`, `enabledGPA`, `erouter`, `GPAdelay`, `moving`, `movingSince`, and `TIMER_SID_GPA`.

7.187.3.5 `void WorldModel2::enableIR ()`

Enable infrared data registration.

The AIBO continually takes depth measurements with the IR range sensor in its nose. Calling this function will cause `WorldModel2` to integrate these range measurements into its maps, along with segmented color data from the vision system for the measured point. Note that IR measurements are only integrated when the AIBO is standing still in the erect posture from `ms/data/motion/stand.pos`.

Definition at line 151 of file `WorldModel2.cc`.

References `EventRouter::addListener()`, `enabledIR`, `erouter`, and `EventBase::sensorEGID`.

7.187.3.6 `void WorldModel2::enableKludge (unsigned int kludge)`

Enable a kludge (see the [WM2Kludge](#) namespace for details).

Definition at line 190 of file `WorldModel2.cc`.

References `kludges`.

7.187.3.7 `void WorldModel2::getRequests (MRvector & requests)`

Get suggestions about what to look at next (ALPHA).

All maps within `WorldModel2` give each of their grid cells 'confidence' scores: measures of how certain the mapping system is about the data they contain. At the moment, these scores are mostly ad-hoc, but later they may be based on more statistically rigorous techniques. In any case, `getRequests` performs a simple k-means cluster analysis on the certainty measurements of the `WorldModel2` maps (where K and the number of iterations are specified in `WorldModel2's Configuration.h`) and places the centers of these clusters in appropriate [MotionRequest](#) structures, which are inserted into the requests vector. Your code may choose to honor these suggestions, or it may not.

Definition at line 284 of file WorldModel2.cc.

References AGM::genRequests(), and ALM::genRequests().

7.187.3.8 **dm_cell** * WorldModel2::invadeDMData (void)

The fast access method for spherical depth map data. Use at own risk.

Definition at line 301 of file WorldModel2.cc.

References ALM::getDM().

7.187.3.9 **hm_cell** * WorldModel2::invadeHMData (void)

Risky fast access method for horiz. height map. See pickHMData note.

Definition at line 311 of file WorldModel2.cc.

References ALM::getHM().

7.187.3.10 **WorldModel2**& WorldModel2::operator= (const **WorldModel2** &) [private]

dummy functions prevent warnings

7.187.3.11 void WorldModel2::pickDMData (**dmPicker** & *p*, float * *dest*)

The safe access method for spherical depth map data.

Definition at line 295 of file WorldModel2.cc.

References DM_CELL_COUNT, and ALM::getDM().

7.187.3.12 void WorldModel2::pickHMData (**hmPicker** & *p*, float * *dest*)

Safe access method for horizontal height map. Do see long description.

You'll want to use the HM_CELL_COUNT define in World-Model2/Maps/Configuration.h for sizing. Do remember that the HHM is **round** and is not guaranteed to maintain all the data in its array—only that within the circle of cells within ALM_HM_SIZE (Euclidean) of the central cell of the array.

Definition at line 305 of file WorldModel2.cc.

References ALM::getHM(), and HM_CELL_COUNT.

7.187.3.13 **void WorldModel2::processEvent (const [EventBase](#) & event)** [virtual]

Process vision and motion events.

Implements [EventListener](#).

Definition at line 259 of file WorldModel2.cc.

References [EventBase::getGeneratorID\(\)](#), [EventBase::getSourceID\(\)](#), [EventBase::locomotionEGID](#), [processGround\(\)](#), [processLocomotion\(\)](#), [processSensors\(\)](#), [EventBase::sensorEGID](#), [serialize\(\)](#), [TIMER_SID_GPA](#), [TIMER_SID_SRL](#), [EventBase::timerEGID](#), and [EventBase::visionEGID](#).

7.187.3.14 **void WorldModel2::processGround ()** [private]

Try to do ground plane assumption.

Definition at line 556 of file WorldModel2.cc.

References [aiboIsErect\(\)](#), [aiboStaresDeadAhead\(\)](#), [enabledGPA](#), [erouter](#), [ALM::registerGround\(\)](#), [EventRouter::removeTimer\(\)](#), and [UNLESS](#).

7.187.3.15 **void WorldModel2::processLocomotion (const [EventBase](#) & event)** [private]

process locomotion events—movements

Definition at line 430 of file WorldModel2.cc.

References [EventRouter::addListener\(\)](#), [EventRouter::addTimer\(\)](#), [AFS_NUM_-LANDMARKS](#), [afsMotion\(\)](#), [afsResample\(\)](#), [afsWhatsUp\(\)](#), [_FastSLAM_update::da](#), [_FastSLAM_update::dx](#), [dx](#), [_FastSLAM_update::dy](#), [dy](#), [enabledGPA](#), [enabledIR](#), [erouter](#), [get_time\(\)](#), [GPAdelay](#), [kludges](#), [ALM::move\(\)](#), [moving](#), [movingSince](#), [EventRouter::removeListener\(\)](#), [EventRouter::removeTimer\(\)](#), [EventBase::sensorEGID](#), [_FastSLAM_update::time](#), [_FastSLAM_update::type](#), and [EventBase::visionEGID](#).

7.187.3.16 **void WorldModel2::processSensors ()** [private]

process sensors, incorporating IRs into local maps

Definition at line 329 of file WorldModel2.cc.

References [enabledIR](#), [ERS210Info::HeadOffset](#), [ERS210Info::IRDistOffset](#), [kludges](#), [WorldState::outputs](#), [ERS210Info::PanOffset](#), [ALM::registerDepth\(\)](#), [WorldState::sensors](#), [state](#), [ERS210Info::TiltOffset](#), and [UNLESS](#).

7.187.3.17 void WorldModel2::resetWorldModel ()

Resets the world model to the start state.

Clears all maps ("empty" maps will contain an assumed ground plane at elevation 0) and, if allocentric functionality is enabled, resets the localization system to its initial zero-knowledge state.

Definition at line 227 of file WorldModel2.cc.

References `afsInit()`, `AGM::init()`, and `ALM::init()`.

7.187.3.18 void WorldModel2::serialize () [private]

Perform serialization – send data down the line, if needed.

Definition at line 584 of file WorldModel2.cc.

References `AFS_NUM_LANDMARKS`, `AFS_NUM_PARTICLES`, `afsInvadeFSDData()`, `afsWhatsUp()`, `_hm_cell::color`, `_dm_cell::color`, `colortype`, `_hm_cell::confidence`, `_dm_cell::confidence`, `_dm_cell::depth`, `DM_CELL_COUNT`, `Serializer::encode()`, `ALM::getDM()`, `AGM::getGM()`, `ALM::getHM()`, `Socket::getWriteBuffer()`, `GM_CELL_COUNT`, `_hm_cell::height`, `HM_CELL_COUNT`, `_afsParticle::landmarks`, `_afsLandmarkLoc::mean`, `Particles`, `_afsParticle::pose`, `sockDM`, `sockHM`, `_afsLandmarkLoc::state`, `_afsPose::theta`, `_hm_cell::trav`, `_afsLandmarkLoc::variance`, `Socket::write()`, `_afsPose::x`, and `_afsPose::y`.

7.187.4 Member Data Documentation

7.187.4.1 double WorldModel2::da [private]

The current velocities of the AIBO.

Definition at line 306 of file WorldModel2.h.

7.187.4.2 double WorldModel2::dx [private]

The current velocities of the AIBO.

Definition at line 306 of file WorldModel2.h.

7.187.4.3 double WorldModel2::dy [private]

The current velocities of the AIBO.

Definition at line 306 of file WorldModel2.h.

7.187.4.4 **bool** [WorldModel2::enabledGPA](#) [private]

True if GPA mapping is enabled (c.f. `enableGPA`).

Definition at line 303 of file `WorldModel2.h`.

7.187.4.5 **bool** [WorldModel2::enabledIR](#) [private]

True if IR sensing is enabled (c.f. `enableIR`).

Definition at line 302 of file `WorldModel2.h`.

7.187.4.6 **const int** [WorldModel2::GPAdelay](#) = 100 [static, private]

Constant indicates how frequently head position sampling should take place for the ground plane assumption. Units are milliseconds.

Definition at line 281 of file `WorldModel2.h`.

7.187.4.7 **unsigned int** [WorldModel2::kludges](#) [private]

Which kludges are enabled? See [WM2Kludge](#) documentation for info.

Definition at line 293 of file `WorldModel2.h`.

7.187.4.8 **bool** [WorldModel2::locked](#) = false [static, private]

The AIBO's memory management system is borked. Consequently, all memory space is allocated statically and can only be used by one `WorldModel2` at a time. This variable indicates whether there's already someone using the data.

Definition at line 61 of file `WorldModel2.cc`.

7.187.4.9 **bool** [WorldModel2::moving](#) [private]

This boolean is set to true if the system isn't certain that AIBO is stopped. In other words, in spite of its name, it may be true even when the AIBO isn't moving, e.g. during emergency stop. If it's false, though, the AIBO should surely be at rest.

Definition at line 299 of file `WorldModel2.h`.

7.187.4.10 **long** [WorldModel2::movingSince](#) [private]

The time when we started going at current velocities. 0 is a special value for moving-Since, indicating that no information on movement has been given to this `WorldModel`

object yet. Routines that use this motion information should check for the 0 value and abend if they find it.

Definition at line 312 of file WorldModel2.h.

7.187.4.11 `Socket* WorldModel2::sockDM` [private]

Here are sockets for serialization of the egocentric map data.

Definition at line 315 of file WorldModel2.h.

7.187.4.12 `Socket * WorldModel2::sockHM` [private]

Here are sockets for serialization of the egocentric map data.

Definition at line 315 of file WorldModel2.h.

7.187.4.13 `const int WorldModel2::SRLdelay = 1000` [static, private]

Constant indicates how frequently serialization of map data should be performed. Units are milliseconds.

Definition at line 284 of file WorldModel2.h.

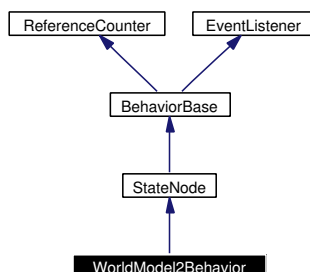
The documentation for this class was generated from the following files:

- [WorldModel2.h](#)
- [WorldModel2.cc](#)

7.188 WorldModel2Behavior Class Reference

```
#include <WorldModel2Behavior.h>
```

Inheritance diagram for WorldModel2Behavior:



7.188.1 Detailed Description

Implements a stateful behavior specifically designed to feed data to the second World-Model.

Definition at line 17 of file WorldModel2Behavior.h.

Public Member Functions

- [WorldModel2Behavior \(\)](#)
Constructor.
- virtual [~WorldModel2Behavior \(\)](#)
- virtual void [setup \(\)](#)
Initial setup.
- virtual void [DoStart \(\)](#)
Behavior startup.
- virtual void [DoStop \(\)](#)
Behavior stop.

Static Public Member Functions

- std::string [getClassDescription \(\)](#)

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Private Member Functions

- [WorldModel2Behavior](#) (const [WorldModel2Behavior](#) &)
don't call
- [WorldModel2Behavior](#) operator= (const [WorldModel2Behavior](#) &)
don't call

Private Attributes

- [StateNode](#) * start
Pointer to the start state.
- [WorldModel2](#) WM2
WorldModel2 controller.
- [SharedObject](#)< [MotionSequenceMC](#)< [MotionSequence::SizeSmall](#) > > standUp
Crap related to standing up.
- [MotionManager::MC_ID](#) standUp_id
id for the standup

7.188.2 Constructor & Destructor Documentation

7.188.2.1 WorldModel2Behavior::WorldModel2Behavior ()

Constructor.

Definition at line 36 of file WorldModel2Behavior.cc.

References [MotionManager::addMotion\(\)](#), [ERS210Info::ElevatorOffset](#), [WorldModel2::enableKludge\(\)](#), [ERS210Info::KneeOffset](#), [ERS210Info::LBkLegOffset](#), [ERS210Info::LFrLegOffset](#), [motman](#), [ERS210Info::RBkLegOffset](#), [ERS210Info::RFRLegOffset](#), [ERS210Info::RotatorOffset](#), [SP_LBK_JOINT](#), [SP_LBK_KNEE](#), [SP_LBK_SHLDR](#), [SP_LFR_JOINT](#), [SP_LFR_KNEE](#), [SP_LFR_SHLDR](#), [SP_RBK_JOINT](#), [SP_RBK_KNEE](#), [SP_RBK_SHLDR](#), [SP_RFR_JOINT](#), [SP_RFR_KNEE](#), [SP_RFR_SHLDR](#), [standUp](#), [standUp_id](#), and [WM2](#).

7.188.2.2 `WorldModel2Behavior::~~WorldModel2Behavior ()` [virtual]

Definition at line 81 of file WorldModel2Behavior.cc.

7.188.2.3 `WorldModel2Behavior::WorldModel2Behavior (const WorldModel2Behavior &)` [private]

don't call

7.188.3 Member Function Documentation**7.188.3.1** `void WorldModel2Behavior::DoStart ()` [virtual]

Behavior startup.

Reimplemented from [StateNode](#).

Definition at line 105 of file WorldModel2Behavior.cc.

References [StateNode::DoStart\(\)](#), [start](#), and [WM2](#).

7.188.3.2 `void WorldModel2Behavior::DoStop ()` [virtual]

Behavior stop.

Reimplemented from [StateNode](#).

Definition at line 114 of file WorldModel2Behavior.cc.

References [StateNode::DoStop\(\)](#), and [WM2](#).

7.188.3.3 `std::string WorldModel2Behavior::getClassDescription ()`
[inline, static]

Gives a short description of what this class of behaviors does... you should override this (but don't have to).

Reimplemented from [BehaviorBase](#).

Definition at line 26 of file WorldModel2Behavior.h.

7.188.3.4 `WorldModel2Behavior WorldModel2Behavior::operator= (const WorldModel2Behavior &)` [private]

don't call

7.188.3.5 void WorldModel2Behavior::setup () [virtual]

Initial setup.

Reimplemented from [StateNode](#).

Definition at line 86 of file WorldModel2Behavior.cc.

References [StateNode::addNode\(\)](#), [StateNode::addTransition\(\)](#), [StateNode::setup\(\)](#), [standUp_id](#), [start](#), and [WM2](#).

7.188.4 Member Data Documentation

7.188.4.1 [SharedObject](#)< [MotionSequenceMC](#)<[MotionSequence::SizeSmall](#)> > [WorldModel2Behavior::standUp](#) [private]

Crap related to standing up.

Definition at line 33 of file WorldModel2Behavior.h.

7.188.4.2 [MotionManager::MC_ID](#) [WorldModel2Behavior::standUp_id](#) [private]

id for the standup

Definition at line 34 of file WorldModel2Behavior.h.

7.188.4.3 [StateNode*](#) [WorldModel2Behavior::start](#) [private]

Pointer to the start state.

Definition at line 29 of file WorldModel2Behavior.h.

7.188.4.4 [WorldModel2](#) [WorldModel2Behavior::WM2](#) [private]

[WorldModel2](#) controller.

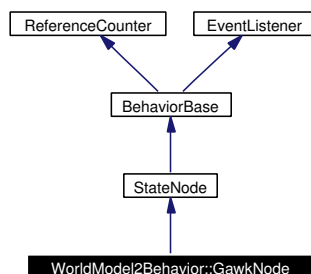
Definition at line 30 of file WorldModel2Behavior.h.

The documentation for this class was generated from the following files:

- [WorldModel2Behavior.h](#)
- [WorldModel2Behavior.cc](#)

7.189 WorldModel2Behavior::GawkNode Struct Reference

Inheritance diagram for WorldModel2Behavior::GawkNode:



7.189.1 Detailed Description

This one wags the head around.

Definition at line 57 of file WorldModel2Behavior.h.

Public Member Functions

- [GawkNode](#) (const char *title, [StateNode](#) *poppa, [WorldModel2](#) *WM, [MotionManager::MC_ID](#) su)
Constructor.
- virtual void [processEvent](#) (const [EventBase](#) &)
Head wag poll timer.
- virtual void [DoStart](#) ()
Behavior startup.
- virtual void [DoStop](#) ()
Behavior stop.
- [GawkNode](#) (const [WorldModel2Behavior::GawkNode](#) &)
don't call
- [GawkNode](#) operator= (const [WorldModel2Behavior::GawkNode](#) &)
don't call

Public Attributes

- [WorldModel2 * WM2ref](#)
Reference to WM2B's WM2 object.
- [MotionManager::MC_ID head_id](#)
A motion command ID for the head.
- [MotionManager::MC_ID stand_id](#)
A motion command ID for standing up.

7.189.2 Constructor & Destructor Documentation

7.189.2.1 WorldModel2Behavior::GawkNode::GawkNode (const char * *title*, [StateNode * poppa](#), [WorldModel2 * WM](#), [MotionManager::MC_ID su](#))

Constructor.

Definition at line 217 of file WorldModel2Behavior.cc.

7.189.2.2 WorldModel2Behavior::GawkNode::GawkNode (const [WorldModel2Behavior::GawkNode &](#))

don't call

7.189.3 Member Function Documentation

7.189.3.1 void WorldModel2Behavior::GawkNode::DoStart () [virtual]

Behavior startup.

Reimplemented from [StateNode](#).

Definition at line 242 of file WorldModel2Behavior.cc.

References [MotionManager::addMotion\(\)](#), [EventRouter::addTimer\(\)](#), [StateNode::DoStart\(\)](#), [erouter](#), [head_id](#), and [motman](#).

7.189.3.2 void WorldModel2Behavior::GawkNode::DoStop () [virtual]

Behavior stop.

Reimplemented from [StateNode](#).

Definition at line 261 of file WorldModel2Behavior.cc.

References AFS_NUM_LANDMARKS, StateNode::DoStop(), erouter, head_id, _afsParticle::landmarks, _afsLandmarkLoc::mean, motman, _afsParticle::pose, MotionManager::removeMotion(), EventRouter::removeTimer(), _afsLandmarkLoc::state, _afsLandmarkLoc::variance, WM2ref, _afsPose::x, and _afsPose::y.

7.189.3.3 [GawkNode](#) WorldModel2Behavior::GawkNode::operator= (const [WorldModel2Behavior::GawkNode](#) &)

don't call

7.189.3.4 void WorldModel2Behavior::GawkNode::processEvent (const [EventBase](#) &) [virtual]

Head wag poll timer.

Reimplemented from [StateNode](#).

Definition at line 229 of file WorldModel2Behavior.cc.

References MotionManager::checkinMotion(), MotionManager::checkoutMotion(), get_time(), head_id, motman, and HeadPointerMC::setJoints().

7.189.4 Member Data Documentation

7.189.4.1 [MotionManager::MC_ID](#) WorldModel2Behavior::GawkNode::head_id

A motion command ID for the head.

Definition at line 69 of file WorldModel2Behavior.h.

7.189.4.2 [MotionManager::MC_ID](#) WorldModel2Behavior::GawkNode::stand_id

A motion command ID for standing up.

Definition at line 70 of file WorldModel2Behavior.h.

7.189.4.3 [WorldModel2*](#) WorldModel2Behavior::GawkNode::WM2ref

Reference to WM2B's WM2 object.

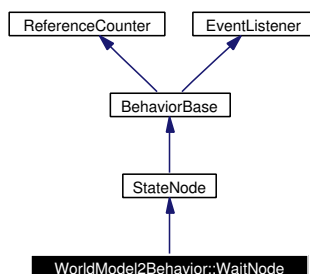
Definition at line 58 of file WorldModel2Behavior.h.

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

- [WorldModel2Behavior.h](#)
- [WorldModel2Behavior.cc](#)

7.190 WorldModel2Behavior::WaitNode Struct Reference

Inheritance diagram for WorldModel2Behavior::WaitNode:



7.190.1 Detailed Description

This one lets the AIBO have a reflective pause.

Definition at line 77 of file WorldModel2Behavior.h.

Public Member Functions

- [WaitNode](#) (const char *title, [StateNode](#) *poppa, [MotionManager::MC_ID](#) su)
Constructor.
- virtual void [DoStart](#) ()
Behavior startup.
- virtual void [DoStop](#) ()
Behavior stop.

Public Attributes

- [MotionManager::MC_ID](#) head_id
A motion command ID for the head.
- [MotionManager::MC_ID](#) stand_id
A motion command ID for standing up.

7.190.2 Constructor & Destructor Documentation

7.190.2.1 WorldModel2Behavior::WaitNode::WaitNode (const char * *title*, [StateNode](#) * *poppa*, [MotionManager::MC_ID](#) *su*)

Constructor.

Definition at line 308 of file WorldModel2Behavior.cc.

7.190.3 Member Function Documentation

7.190.3.1 void WorldModel2Behavior::WaitNode::DoStart () [virtual]

Behavior startup.

Reimplemented from [StateNode](#).

Definition at line 317 of file WorldModel2Behavior.cc.

References [MotionManager::addMotion\(\)](#), [MotionManager::checkinMotion\(\)](#), [MotionManager::checkoutMotion\(\)](#), [StateNode::DoStart\(\)](#), [head_id](#), [motman](#), and [HeadPointerMC::setJoints\(\)](#).

7.190.3.2 void WorldModel2Behavior::WaitNode::DoStop () [virtual]

Behavior stop.

Reimplemented from [StateNode](#).

Definition at line 332 of file WorldModel2Behavior.cc.

References [StateNode::DoStop\(\)](#), [head_id](#), [motman](#), and [MotionManager::removeMotion\(\)](#).

7.190.4 Member Data Documentation

7.190.4.1 [MotionManager::MC_ID](#) WorldModel2Behavior::WaitNode::head_id

A motion command ID for the head.

Definition at line 84 of file WorldModel2Behavior.h.

7.190.4.2 [MotionManager::MC_ID](#) WorldModel2Behavior::WaitNode::stand_id

A motion command ID for standing up.

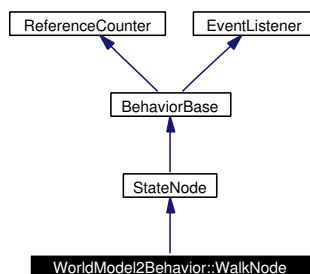
Definition at line 85 of file WorldModel2Behavior.h.

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

- [WorldModel2Behavior.h](#)
- [WorldModel2Behavior.cc](#)

7.191 WorldModel2Behavior::WalkNode Struct Reference

Inheritance diagram for WorldModel2Behavior::WalkNode:



7.191.1 Detailed Description

This one does a random walk.

Definition at line 38 of file `WorldModel2Behavior.h`.

Public Types

- enum { [TURN](#), [AHEAD](#) }

Public Member Functions

- [WalkNode](#) (const char *title, [StateNode](#) *popppa, [MotionManager::MC.ID](#) su)
Constructor.
- virtual [~WalkNode](#) ()
Destructor.
- virtual void [processEvent](#) (const [EventBase](#) &)
IR obstacle avoidance.
- virtual void [DoStart](#) ()
Behavior startup.
- virtual void [DoStop](#) ()
Behavior stop.

Public Attributes

- [MotionManager::MC_ID](#) `walker_id`
A motion command ID for [WalkMC](#).
- [MotionManager::MC_ID](#) `head_id`
A motion command ID for the head.
- [MotionManager::MC_ID](#) `stand_id`
A motion command ID for standing up.
- `enum WorldModel2Behavior::WalkNode:: { ... } walkstate`
are we walking forward or turning?

Static Public Attributes

- `const double THR_TURN = 300`
obstacle threshold for turning away
- `const double THR_AHEAD = 600`
obstacle threshild for stopping turn

7.191.2 Member Enumeration Documentation

7.191.2.1 anonymous enum

Enumeration values:

TURN
AHEAD

Definition at line 51 of file WorldModel2Behavior.h.

7.191.3 Constructor & Destructor Documentation

7.191.3.1 `WorldModel2Behavior::WalkNode::WalkNode (const char * title, StateNode * poppa, MotionManager::MC_ID su)`

Constructor.

Definition at line 128 of file WorldModel2Behavior.cc.

References [MotionManager::addMotion\(\)](#), `head_id`, `motman`, and `walker_id`.

7.191.3.2 WorldModel2Behavior::WalkNode::~WalkNode () [virtual]

Destructor.

Definition at line 141 of file WorldModel2Behavior.cc.

References head_id, motman, MotionManager::removeMotion(), and walker_id.

7.191.4 Member Function Documentation

7.191.4.1 void WorldModel2Behavior::WalkNode::DoStart () [virtual]

Behavior startup.

Reimplemented from [StateNode](#).

Definition at line 176 of file WorldModel2Behavior.cc.

References EventRouter::addListener(), AHEAD, MotionManager::checkinMotion(), MotionManager::checkoutMotion(), StateNode::DoStart(), erouter, get_time(), head_id, motman, EventBase::sensorEGID, HeadPointerMC::setJoints(), WalkMC::setTargetVelocity(), walker_id, and walkstate.

7.191.4.2 void WorldModel2Behavior::WalkNode::DoStop () [virtual]

Behavior stop.

Reimplemented from [StateNode](#).

Definition at line 196 of file WorldModel2Behavior.cc.

References MotionManager::checkinMotion(), MotionManager::checkoutMotion(), StateNode::DoStop(), erouter, get_time(), motman, EventRouter::removeListener(), EventBase::sensorEGID, WalkMC::setTargetVelocity(), and walker_id.

7.191.4.3 void WorldModel2Behavior::WalkNode::processEvent (const [EventBase](#) &) [virtual]

IR obstacle avoidance.

Reimplemented from [StateNode](#).

Definition at line 150 of file WorldModel2Behavior.cc.

References AHEAD, MotionManager::checkinMotion(), MotionManager::checkoutMotion(), ERS210Info::IRDistOffset, motman, WorldState::sensors, WalkMC::setTargetVelocity(), state, THR_AHEAD, THR_TURN, TURN, walker_id, and walkstate.

7.191.5 Member Data Documentation

7.191.5.1 [MotionManager::MC_ID](#) [WorldModel2Behavior::WalkNode::head_id](#)

A motion command ID for the head.

Definition at line 49 of file [WorldModel2Behavior.h](#).

7.191.5.2 [MotionManager::MC_ID](#) [WorldModel2Behavior::WalkNode::stand_id](#)

A motion command ID for standing up.

Definition at line 50 of file [WorldModel2Behavior.h](#).

7.191.5.3 `const double` [WorldModel2Behavior::WalkNode::THR_AHEAD](#) = 600 [static]

obstacle threshild for stopping turn

Definition at line 54 of file [WorldModel2Behavior.h](#).

7.191.5.4 `const double` [WorldModel2Behavior::WalkNode::THR_TURN](#) = 300 [static]

obstacle threshold for turning away

Definition at line 53 of file [WorldModel2Behavior.h](#).

7.191.5.5 [MotionManager::MC_ID](#) [WorldModel2Behavior::WalkNode::walker_id](#)

A motion command ID for [WalkMC](#).

Definition at line 48 of file [WorldModel2Behavior.h](#).

7.191.5.6 `enum { ... }` [WorldModel2Behavior::WalkNode::walkstate](#)

are we walking forward or turning?

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

- [WorldModel2Behavior.h](#)
- [WorldModel2Behavior.cc](#)

7.192 WorldState Class Reference

```
#include <WorldState.h>
```

7.192.1 Detailed Description

The state of the robot and its environment.

Definition at line 126 of file WorldState.h.

Public Member Functions

- [WorldState](#) ()
constructor - sets everything to zeros
- void [read](#) (OSensorFrameVectorData &sensor, [EventRouter](#) *er)
will process a sensor reading as given by OPEN-R
- void [read](#) (const OPowerStatus &power, [EventRouter](#) *er)
will process a power status update as given by OPEN-R

Public Attributes

- bool [alwaysGenerateStatus](#)
controls whether status events are generated for the boolean buttons
- float [outputs](#) [NumOutputs]
same format as Motion stuff, for ears, $x < .5$ is up, $x \geq .5$ is down
- float [buttons](#) [NumButtons]
magnitude is pressure for some, 0/1 for others
- float [sensors](#) [NumSensors]
IR, Accel, Thermo, Power stuff.
- float [pids](#) [NumPIDJoints][3]
current PID settings
- float [pidduties](#) [NumPIDJoints]

duty cycles - -1 means the motor is trying to move full power in negative direction, 1 means full power in positive direction, in practice, these values stay rather small - 0.15 is significant force.

- unsigned int [robotStatus](#)
bitmask, see OPENR/OPower.h
- unsigned int [batteryStatus](#)
bitmask, see OPENR/OPower.h
- unsigned int [powerFlags](#) [PowerSourceID::NumPowerSIDs]
bitmasks of similarly-grouped items from previous two masks, corresponds to the PowerSourceID_t's
- unsigned int [button.times](#) [NumButtons]
value is time of current press, 0 if not down
- unsigned int [lastSensorUpdateTime](#)
primarily so calcDers can determine the time difference between updates, but others might want to know this too...
- [ProfilerOfSize< 20 > mainProfile](#)
holds code profiling information for MainObject
- [ProfilerOfSize< 06 > motionProfile](#)
holds code profiling information for MotionObject
- [WorldStateSerializer * wsser](#)
sends current worldstate to TekkotsuMon, if it's connected
- unsigned int [robotDesign](#)
bitmask corresponding to OPENR::GetRobotDesign()

Static Public Attributes

- const double [g](#) = 9.80665
the gravitational acceleration of objects on earth
- const double [IROORDist](#) = 900.0
If IR returns this, we're out of range.
- const unsigned int [ERS210Mask](#) = 1<<0

use this to test for ERS210 features

- const unsigned int [ERS220Mask](#) = 1<<1

use this to test for ERS220 features

Protected Member Functions

- void [chkEvent](#) ([EventRouter](#) *er, unsigned int off, float newval, const char *name)

Tests to see if the button status has changed and post events as needed.

- void [chkPowerEvent](#) (unsigned int sid, unsigned int cur, unsigned int mask, const char *name, std::string actname[PowerSourceID::NumPowerSIDs], std::string dename[PowerSourceID::NumPowerSIDs], unsigned int summask[PowerSourceID::NumPowerSIDs])

sets the names of the flags that will be generating events

- void [calcDers](#) (double next, double &cur, double &vel, double &acc, double invtimediff)

given the next value, calculates and stores the next current, the velocity, and the acceleration

Protected Attributes

- unsigned int [curtime](#)

set by [read\(OSensorFrameVectorData& sensor, EventRouter er\)](#) for [chkEvent\(\)](#) so each call doesn't have to*

Private Member Functions

- [WorldState](#) (const [WorldState](#) &)

don't use

- [WorldState](#) operator= (const [WorldState](#) &)

don't use

7.192.2 Constructor & Destructor Documentation

7.192.2.1 WorldState::WorldState ()

constructor - sets everything to zeros

Definition at line 26 of file WorldState.cc.

References `button_times`, `buttons`, `ERS210Mask`, `ERS220Mask`, `get_time()`, `ERS210Info::NumButtons`, `ERS210Info::NumOutputs`, `ERS210Info::NumPIDJoints`, `ERS210Info::NumSensors`, `outputs`, `pidduties`, `pids`, `powerFlags`, `robotDesign`, `sensors`, and `wssr`.

7.192.2.2 WorldState::WorldState (const WorldState &) [private]

don't use

7.192.3 Member Function Documentation

7.192.3.1 void WorldState::calcDers (double *next*, double & *cur*, double & *vel*, double & *acc*, double *invtimediff*) [inline, protected]

given the next value, calculates and stores the next current, the velocity, and the acceleration

Parameters:

next the new value that's about to be set

cur the previous value

vel the previous 1st derivative

acc the previous 2nd derivative

invtimediff $1/(\text{curtime} - \text{prevtime})$ in seconds

Definition at line 196 of file WorldState.h.

7.192.3.2 void WorldState::chkEvent (EventRouter * *er*, unsigned int *off*, float *newval*, const char * *name*) [protected]

Tests to see if the button status has changed and post events as needed.

Definition at line 287 of file WorldState.cc.

References `EventBase::activateETID`, `alwaysGenerateStatus`, `button_times`, `EventBase::buttonEGID`, `buttons`, `curtime`, `EventBase::deactivateETID`, `EventRouter::postEvent()`, and `EventBase::statusETID`.

7.192.3.3 `void WorldState::chkPowerEvent (unsigned int sid, unsigned int cur, unsigned int mask, const char * name, std::string actname[PowerSourceID::NumPowerSIDs], std::string dename[PowerSourceID::NumPowerSIDs], unsigned int summask[PowerSourceID::NumPowerSIDs])` [inline, protected]

sets the names of the flags that will be generating events

note that this function does not actually do the event posting

Definition at line 179 of file WorldState.h.

References powerFlags.

7.192.3.4 `WorldState WorldState::operator= (const WorldState &)` [private]

don't use

7.192.3.5 `void WorldState::read (const OPowerStatus & power, EventRouter * er)`

will process a power status update as given by OPEN-R

This will cause events to be posted

Definition at line 213 of file WorldState.cc.

References EventBase::activateETID, chkPowerEvent(), EventBase::deactivateETID, EventRouter::postEvent(), ERS210Info::PowerCapacityOffset, ERS210Info::PowerCurrentOffset, EventBase::powerEGID, powerFlags, ERS210Info::PowerRemainOffset, ERS210Info::PowerThermoOffset, ERS210Info::PowerVoltageOffset, sensors, and EventBase::statusETID.

7.192.3.6 `void WorldState::read (OSensorFrameVectorData & sensor, EventRouter * er)`

will process a sensor reading as given by OPEN-R

This will cause events to be posted

Todo

change to use most recent instead of oldest - is a buffer!

Definition at line 71 of file WorldState.cc.

References ERS210Info::BAccelOffset, ERS210Info::BackButOffset,
 ERS210Info::ChinButOffset, chkEvent(), curtime, ERS210Info::DAccelOffset,
 ERS210Info::ElevatorOffset, ERS210Mask, ERS220Mask, get_time(), GETB,
 GETD, GETDUTY, ERS210Info::HeadBkButOffset, ERS210Info::HeadFrButOffset,
 ERS210Info::HeadOffset, ERS210Info::IRDistOffset, ERS210Info::KneeOffset,
 ERS210Info::LAccelOffset, lastSensorUpdateTime, ERS210Info::LBkLegOffset,
 ERS210Info::LBkPawOffset, ERS210Info::LFrLegOffset, ERS210Info::LFrPaw-
 Offset, outputs, ERS210Info::PanOffset, pidduties, EventRouter::postEvent(),
 ERS210Info::RBkLegOffset, ERS210Info::RBkPawOffset, ERS210Info::RFr-
 LegOffset, ERS210Info::RFrPawOffset, robotDesign, ERS210Info::RollOffset,
 ERS210Info::RotatorOffset, EventBase::sensorEGID, sensors, EventBase::status-
 ETID, ERS210Info::ThermoOffset, and ERS210Info::TiltOffset.

7.192.4 Member Data Documentation

7.192.4.1 bool [WorldState::alwaysGenerateStatus](#)

controls whether status events are generated for the boolean buttons

Definition at line 131 of file WorldState.h.

7.192.4.2 unsigned int [WorldState::batteryStatus](#)

bitmask, see OPENR/OPower.h

Definition at line 140 of file WorldState.h.

7.192.4.3 unsigned int [WorldState::button.times](#)[NumButtons]

value is time of current press, 0 if not down

Definition at line 143 of file WorldState.h.

7.192.4.4 float [WorldState::buttons](#)[NumButtons]

magnitude is pressure for some, 0/1 for others

Definition at line 134 of file WorldState.h.

7.192.4.5 unsigned int [WorldState::curtime](#) [protected]

set by [read\(OSensorFrameVectorData& sensor, EventRouter* er\)](#) for [chkEvent\(\)](#) so
 each call doesn't have to

Definition at line 172 of file WorldState.h.

7.192.4.6 `const unsigned int WorldState::ERS210Mask = 1<<0` [static]

use this to test for ERS210 features

Definition at line 168 of file WorldState.h.

7.192.4.7 `const unsigned int WorldState::ERS220Mask = 1<<1` [static]

use this to test for ERS220 features

Definition at line 169 of file WorldState.h.

7.192.4.8 `const double WorldState::g = 9.80665` [static]

the gravitational acceleration of objects on earth

Definition at line 21 of file WorldState.cc.

7.192.4.9 `const double WorldState::IROORDist = 900.0` [static]

If IR returns this, we're out of range.

Definition at line 22 of file WorldState.cc.

7.192.4.10 `unsigned int WorldState::lastSensorUpdateTime`

primarily so calcDers can determine the time difference between updates, but others might want to know this too...

Definition at line 145 of file WorldState.h.

7.192.4.11 `ProfilerOfSize<20> WorldState::mainProfile`

holds code profiling information for MainObject

Definition at line 150 of file WorldState.h.

7.192.4.12 `ProfilerOfSize<06> WorldState::motionProfile`

holds code profiling information for MotionObject

Definition at line 151 of file WorldState.h.

7.192.4.13 float [WorldState::outputs](#)[NumOutputs]

same format as Motion stuff, for ears, $x < .5$ is up, $x \geq .5$ is down

Definition at line 133 of file WorldState.h.

7.192.4.14 float [WorldState::pidduties](#)[NumPIDJoints]

duty cycles - -1 means the motor is trying to move full power in negative direction, 1 means full power in positive direction, in practice, these values stay rather small - 0.15 is significant force.

Definition at line 137 of file WorldState.h.

7.192.4.15 float [WorldState::pids](#)[NumPIDJoints][3]

current PID settings

Definition at line 136 of file WorldState.h.

7.192.4.16 unsigned int [WorldState::powerFlags](#)[PowerSourceID::NumPower-SIDs]

bitmasks of similarly-grouped items from previous two masks, corresponds to the PowerSourceID_t's

Definition at line 141 of file WorldState.h.

7.192.4.17 unsigned int [WorldState::robotDesign](#)

bitmask corresponding to OPENR::GetRobotDesign()

This allows you to efficiently test different combinations, like any 2x0 model vs. any 3xx model (when/if the 3xx's are supported).

Testing this will give more accurate feedback as to whether features exist than checking [RobotInfo](#) values - to achieve dual booting, [RobotInfo](#) may, for instance, tell you there are two ears, but if you're running on a 220 the value you set them to will be ignored

Definition at line 167 of file WorldState.h.

7.192.4.18 unsigned int [WorldState::robotStatus](#)

bitmask, see OPENR/OPower.h

Definition at line 139 of file WorldState.h.

7.192.4.19 float [WorldState::sensors](#)[NumSensors]

IR, Accel, Thermo, Power stuff.

Definition at line 135 of file WorldState.h.

7.192.4.20 [WorldStateSerializer](#)* [WorldState::ws](#)

sends current worldstate to TekkotsuMon, if it's connected

Definition at line 157 of file WorldState.h.

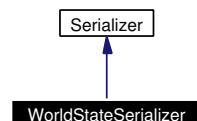
The documentation for this class was generated from the following files:

- [WorldState.h](#)
- [WorldState.cc](#)

7.193 WorldStateSerializer Class Reference

```
#include <WorldStateSerializer.h>
```

Inheritance diagram for WorldStateSerializer:



Public Member Functions

- [WorldStateSerializer](#) ()
- void [serialize](#) ()

Private Member Functions

- [WorldStateSerializer](#) (const [WorldStateSerializer](#) &)
- [WorldStateSerializer](#) & [operator=](#) (const [WorldStateSerializer](#) &)

Private Attributes

- [Socket](#) * [wsJoints](#)
- [Socket](#) * [wsPIDs](#)

7.193.1 Constructor & Destructor Documentation

7.193.1.1 WorldStateSerializer::WorldStateSerializer ()

Definition at line 8 of file `WorldStateSerializer.cc`.

References `config`, `Wireless::listen()`, `Config::main`, `Wireless::socket()`, `wireless`, `wsJoints`, `Config::main_config::wsjoints_port`, `wsPIDs`, and `Config::main_config::wspids_port`.

7.193.1.2 [WorldStateSerializer::WorldStateSerializer](#) (const [WorldStateSerializer](#) &) [private]

7.193.2 Member Function Documentation

7.193.2.1 [WorldStateSerializer](#) & [WorldStateSerializer::operator=](#) (const [WorldStateSerializer](#) &) [private]

7.193.2.2 void [WorldStateSerializer::serialize](#) ()

Definition at line 16 of file [WorldStateSerializer.cc](#).

References [WorldState::buttons](#), [Serializer::encode\(\)](#), [Socket::getWriteBuffer\(\)](#), [WorldState::lastSensorUpdateTime](#), [ERS210Info::NumPIDJoints](#), [WorldState::outputs](#), [WorldState::pidduties](#), [ERS210Info::PIDJointOffset](#), [WorldState::pids](#), [WorldState::sensors](#), [state](#), [Socket::write\(\)](#), [wsJoints](#), and [wsPIDs](#).

7.193.3 Member Data Documentation

7.193.3.1 [Socket*](#) [WorldStateSerializer::wsJoints](#) [private]

Definition at line 17 of file [WorldStateSerializer.h](#).

7.193.3.2 [Socket*](#) [WorldStateSerializer::wsPIDs](#) [private]

Definition at line 18 of file [WorldStateSerializer.h](#).

The documentation for this class was generated from the following files:

- [WorldStateSerializer.h](#)
- [WorldStateSerializer.cc](#)

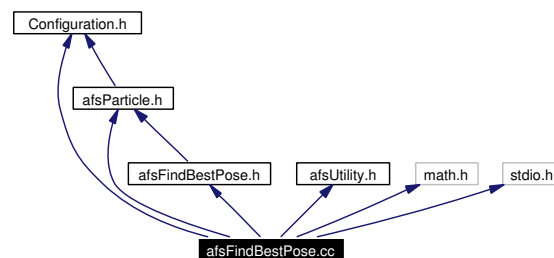
Chapter 8

Tekkotsu File Documentation

8.1 afsFindBestPose.cc File Reference

```
#include "afsParticle.h"
#include "afsUtility.h"
#include "afsFindBestPose.h"
#include "Configuration.h"
#include <math.h>
#include <stdio.h>
```

Include dependency graph for afsFindBestPose.cc:



Functions

- `afsPose afsFindBestPose (afsParticle *p, afsPose pose, afsXY *real_landmarks)`
- `void afsApplyBestPose (afsParticle *p, afsParticle *new_p, afsPose best_pose)`

- double `afsParticleError` (`afsParticle` *p, `afsPose` pose, `afsXY` *real_landmarks)
- void `afsGuessState` (`afsParticle` *P, `afsPose` *g_pose, `afsXY` *g_landmarks)

8.1.1 Function Documentation

8.1.1.1 void `afsApplyBestPose` (`afsParticle` *p, `afsParticle` *new_p, `afsPose` best_pose)

This routine takes the location and orientation in the particle p and places the same information in new_p after applying the translation and rotation recommended in best_pose (which you can get from `afsFindBestPose`). You can make p and new_p point to the same thing safely, but why would you want to?

Definition at line 117 of file `afsFindBestPose.cc`.

References `AFS_NUM_LANDMARKS`, `find_dtheta()`, `_afsParticle::landmarks`, `_afsLandmarkLoc::mean`, `_afsParticle::pose`, `_afsLandmarkLoc::state`, `_afsPose::theta`, `_afsPose::x`, and `_afsPose::y`.

8.1.1.2 `afsPose` `afsFindBestPose` (`afsParticle` *p, `afsPose` pose, `afsXY` *real_landmarks)

This routine determines the best orientation of the particle p given the supplied robot pose and landmark positions. The latter are supplied in an array pointed to by the last argument (note: the landmark ID numbers will be used to index the array).

Definition at line 23 of file `afsFindBestPose.cc`.

References `AFS_NUM_LANDMARKS`, `find_dtheta()`, `_afsParticle::landmarks`, `_afsLandmarkLoc::mean`, `_afsParticle::pose`, `_afsLandmarkLoc::state`, `_afsPose::theta`, `_afsXY::x`, `_afsPose::x`, `_afsXY::y`, and `_afsPose::y`.

8.1.1.3 void `afsGuessState` (`afsParticle` *P, `afsPose` *g_pose, `afsXY` *g_landmarks)

This routine makes an estimated map of landmarks in the environment and comes up with an estimated robot pose for situations where ground truth is not known. It does all of this by averaging. The first argument is a collection of all the particles in the filter. g in the arguments stands for "guess".

Definition at line 193 of file `afsFindBestPose.cc`.

References `AFS_NUM_LANDMARKS`, `AFS_NUM_PARTICLES`, `find_dtheta()`, `_afsParticle::landmarks`, `_afsLandmarkLoc::mean`, `_afsParticle::pose`, `_afsLandmarkLoc::state`, `_afsPose::theta`, `_afsPose::x`, `_afsLandmarkLoc::x`, `_afsXY::x`, `_afsPose::y`, `_afsLandmarkLoc::y`, and `_afsXY::y`.

8.1.1.4 double afsParticleError (afsParticle * *p*, afsPose *pose*, afsXY * *realLandmarks*)

This routine finds the error value for a particle given real map information. The function we use to calculate the error here is the same one we minimized in afsFindBestPose, only there we were just concerned with its derivative. NOTE: The more unprimed landmarks a particle has, the lower its error is likely to be. This does not affect the performance of the algorithm

Definition at line 161 of file afsFindBestPose.cc.

References AFS_NUM_LANDMARKS, _afsParticle::landmarks, _afsLandmarkLoc::mean, _afsParticle::pose, _afsLandmarkLoc::state, _afsXY::x, _afsPose::x, and _afsPose::y.

8.2 afsFindBestPose.h File Reference

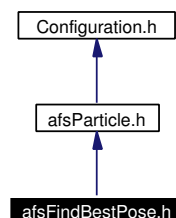
8.2.1 Detailed Description

Finds the proper orientation of the landmark and position information within a particle based on the known locations of landmarks.

Definition in file [afsFindBestPose.h](#).

```
#include "afsParticle.h"
```

Include dependency graph for afsFindBestPose.h:



This graph shows which files directly or indirectly include this file:



Compounds

- struct [_afsRRP](#)
- struct [_afsXY](#)

Typedefs

- typedef [_afsRRP](#) afsRRP
- typedef [_afsXY](#) afsXY

Functions

- [afsPose](#) afsFindBestPose ([afsParticle](#) *p, [afsPose](#) pose, [afsXY](#) *real.landmarks)

- void `afsApplyBestPose` (`afsParticle` *p, `afsParticle` *new_p, `afsPose` best_pose)
- double `afsParticleError` (`afsParticle` *p, `afsPose` pose, `afsXY` *real_landmarks)
- void `afsGuessState` (`afsParticle` *P, `afsPose` *g_pose, `afsXY` *g_landmarks)

8.2.2 Typedef Documentation

8.2.2.1 typedef struct `_afsRRP` `afsRRP`

A structure that contains the relative radial position of a landmark (the robot is at the origin).

8.2.2.2 typedef struct `_afsXY` `afsXY`

A structure that simply contains the x,y coordinates of landmarks.

8.2.3 Function Documentation

8.2.3.1 void `afsApplyBestPose` (`afsParticle` *p, `afsParticle` *new_p, `afsPose` best_pose)

This routine takes the location and orientation in the particle p and places the same information in new_p after applying the translation and rotation recommended in best_pose (which you can get from `afsFindBestPose`). You can make p and new_p point to the same thing safely, but why would you want to?

Definition at line 117 of file `afsFindBestPose.cc`.

References `AFS_NUM_LANDMARKS`, `find_dtheta()`, `_afsParticle::landmarks`, `_afsLandmarkLoc::mean`, `_afsParticle::pose`, `_afsLandmarkLoc::state`, `_afsPose::theta`, `_afsPose::x`, and `_afsPose::y`.

8.2.3.2 `afsPose` `afsFindBestPose` (`afsParticle` *p, `afsPose` pose, `afsXY` *real_landmarks)

This routine determines the best orientation of the particle p given the supplied robot pose and landmark positions. The latter are supplied in an array pointed to by the last argument (note: the landmark ID numbers will be used to index the array).

Definition at line 23 of file `afsFindBestPose.cc`.

References `AFS_NUM_LANDMARKS`, `find_dtheta()`, `_afsParticle::landmarks`, `_afsLandmarkLoc::mean`, `_afsParticle::pose`, `_afsLandmarkLoc::state`, `_afsPose::theta`, `_afsPose::x`, `_afsXY::x`, `_afsPose::y`, and `_afsXY::y`.

8.2.3.3 void afsGuessState (afsParticle * *P*, afsPose * *g_pose*, afsXY * *g_landmarks*)

This routine makes an estimated map of landmarks in the environment and comes up with an estimated robot pose for situations where ground truth is not known. It does all of this by averaging. The first argument is a collection of all the particles in the filter. *g* in the arguments stands for "guess".

Definition at line 193 of file afsFindBestPose.cc.

References AFS_NUM_LANDMARKS, AFS_NUM_PARTICLES, find_dtheta(), _afsParticle::landmarks, _afsLandmarkLoc::mean, _afsParticle::pose, _afsLandmarkLoc::state, _afsPose::theta, _afsXY::x, _afsLandmarkLoc::x, _afsPose::x, _afsXY::y, _afsLandmarkLoc::y, and _afsPose::y.

8.2.3.4 double afsParticleError (afsParticle * *p*, afsPose *pose*, afsXY * *real_landmarks*)

This routine finds the error value for a particle given real map information. The function we use to calculate the error here is the same one we minimized in afsFindBestPose, only there we were just concerned with its derivative. NOTE: The more unprimed landmarks a particle has, the lower its error is likely to be. This does not affect the performance of the algorithm

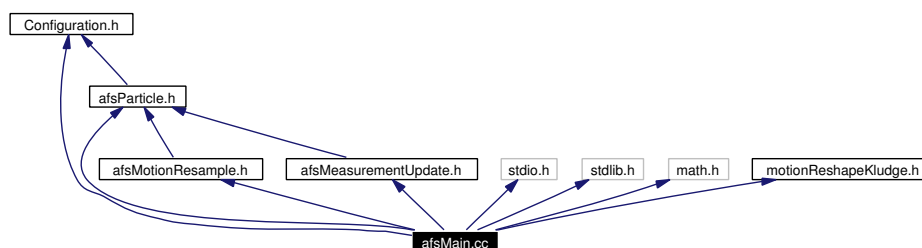
Definition at line 161 of file afsFindBestPose.cc.

References AFS_NUM_LANDMARKS, _afsParticle::landmarks, _afsLandmarkLoc::mean, _afsParticle::pose, _afsLandmarkLoc::state, _afsPose::x, _afsXY::x, and _afsPose::y.

8.3 afsMain.cc File Reference

```
#include "afsParticle.h"
#include "afsMotionResample.h"
#include "afsMeasurementUpdate.h"
#include "Configuration.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "motionReshapeKludge.h"
```

Include dependency graph for afsMain.cc:



Functions

- void [afsInit](#) ()
- void [afsMotion](#) (double [dx](#), double [dy](#), double [da](#), unsigned int [time](#))
- void [afsSetLandmark](#) (int landmark, double x, double y, double covariance)
- void [afsDistribute](#) (double lx, double ty, double rx, double by)
- void [afsMeasurement](#) (int landmark, double theta)
- void [afsResample](#) ()
- [afsParticle](#) * [afsWhatsUp](#) (double *error)
- double [afsCertainty](#) ()
- [afsParticle](#) * [afsInvadeFSData](#) ()

Variables

- [afsParticle](#) [ParticleSets](#) [2][AFS_NUM_PARTICLES]
- [afsParticle](#) * [Particles](#)
- [afsParticle](#) * [newParticles](#)
- double [Weights](#) [AFS_NUM_PARTICLES]

8.3.1 Function Documentation

8.3.1.1 `double afsCertainty ()`

Definition at line 390 of file afsMain.cc.

References AFS_NUM_LANDMARKS, AFS_NUM_PARTICLES, afsApplyBestPose(), afsFindBestPose(), afsGuessState(), afsParticleError(), afsWhatsUp(), and Particles.

8.3.1.2 `void afsDistribute (double lx, double ty, double rx, double by)`

Definition at line 77 of file afsMain.cc.

References AFS_NUM_PARTICLES, dx, Particles, _afsParticle::pose, _afsPose::theta, _afsPose::x, and _afsPose::y.

8.3.1.3 `void afsInit ()`

Definition at line 26 of file afsMain.cc.

References AFS_NUM_PARTICLES, afsParticleInit(), newParticles, Particles, and ParticleSets.

8.3.1.4 `afsParticle* afsInvadeFSDData ()`

Definition at line 428 of file afsMain.cc.

References Particles.

8.3.1.5 `void afsMeasurement (int landmark, double theta)`

Definition at line 91 of file afsMain.cc.

References AFS_NUM_PARTICLES, afsMeasurementUpdate(), and Particles.

8.3.1.6 `void afsMotion (double dx, double dy, double da, unsigned int time)`

Definition at line 36 of file afsMain.cc.

References AFS_NUM_PARTICLES, afsMotionResample(), da, dx, Particles, and time.

8.3.1.7 void afsResample ()

Definition at line 100 of file afsMain.cc.

References AFS_NUM_PARTICLES, afsParticleCopy(), _afsParticle::gotweight, newParticles, Particles, _afsParticle::weight, and Weights.

8.3.1.8 void afsSetLandmark (int *landmark*, double *x*, double *y*, double *covariance*)

Definition at line 56 of file afsMain.cc.

References AFS_NUM_PARTICLES, _afsParticle::landmarks, _afsLandmarkLoc::mean, Particles, _afsLandmarkLoc::state, and _afsLandmarkLoc::variance.

8.3.1.9 [afsParticle](#)* afsWhatsUp (double * *error*)

Definition at line 188 of file afsMain.cc.

References AFS_NUM_LANDMARKS, AFS_NUM_PARTICLES, afsApplyBestPose(), afsFindBestPose(), afsGuessState(), afsParticleError(), afsParticleInit(), dx, _afsParticle::landmarks, _afsLandmarkLoc::mean, Particles, _afsParticle::pose, _afsLandmarkLoc::state, _afsPose::theta, _afsLandmarkLoc::variance, _afsPose::x, _afsXY::x, _afsPose::y, and _afsXY::y.

8.3.2 Variable Documentation

8.3.2.1 [afsParticle](#) * [newParticles](#)

Definition at line 22 of file afsMain.cc.

8.3.2.2 [afsParticle](#)* [Particles](#)

Definition at line 22 of file afsMain.cc.

8.3.2.3 [afsParticle](#) [ParticleSets](#)[2][AFS_NUM_PARTICLES]

Definition at line 21 of file afsMain.cc.

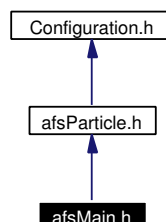
8.3.2.4 double [Weights](#)[AFS_NUM_PARTICLES]

Definition at line 23 of file afsMain.cc.

8.4 afsMain.h File Reference

```
#include "afsParticle.h"
```

Include dependency graph for afsMain.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [afsInit](#) ()
- void [afsSetLandmark](#) (int landmark, double x, double y, double covariance)
- void [afsDistribute](#) (double lx, double ty, double rx, double by)
- void [afsMotion](#) (double dx, double dy, double da, unsigned int time)
- void [afsMeasurement](#) (int landmark, double theta)
- void [afsResample](#) ()
- [afsParticle](#) * [afsWhatsUp](#) (double *error)
- double [afsCertainty](#) ()
- [afsParticle](#) * [afsInvadeFSDData](#) ()

8.4.1 Function Documentation

8.4.1.1 double afsCertainty ()

Definition at line 390 of file afsMain.cc.

References AFS_NUM_LANDMARKS, AFS_NUM_PARTICLES, afsApplyBestPose(), afsFindBestPose(), afsGuessState(), afsParticleError(), afsWhatsUp(), and Particles.

8.4.1.2 void afsDistribute (double *lx*, double *ty*, double *rx*, double *by*)

Definition at line 77 of file afsMain.cc.

References AFS_NUM_PARTICLES, dx, Particles, _afsParticle::pose, _afsPose::theta, _afsPose::x, and _afsPose::y.

8.4.1.3 void afsInit ()

Definition at line 26 of file afsMain.cc.

References AFS_NUM_PARTICLES, afsParticleInit(), newParticles, Particles, and ParticleSets.

8.4.1.4 [afsParticle*](#) afsInvadeFSData ()

Definition at line 428 of file afsMain.cc.

References Particles.

8.4.1.5 void afsMeasurement (int *landmark*, double *theta*)

Definition at line 91 of file afsMain.cc.

References AFS_NUM_PARTICLES, afsMeasurementUpdate(), and Particles.

8.4.1.6 void afsMotion (double *dx*, double *dy*, double *da*, unsigned int *time*)

Definition at line 36 of file afsMain.cc.

References AFS_NUM_PARTICLES, afsMotionResample(), da, dx, Particles, and time.

8.4.1.7 void afsResample ()

Definition at line 100 of file afsMain.cc.

References AFS_NUM_PARTICLES, afsParticleCopy(), _afsParticle::gotweight, newParticles, Particles, _afsParticle::weight, and Weights.

8.4.1.8 void afsSetLandmark (int *landmark*, double *x*, double *y*, double *covariance*)

Definition at line 56 of file afsMain.cc.

References AFS_NUM_PARTICLES, _afsParticle::landmarks, _afsLandmarkLoc::mean, Particles, _afsLandmarkLoc::state, and _afsLandmarkLoc::variance.

8.4.1.9 [afsParticle](#)* afsWhatsUp (double * *error*)

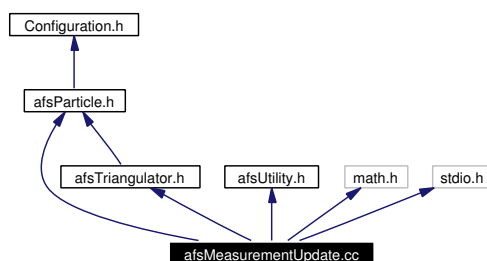
Definition at line 188 of file afsMain.cc.

References AFS_NUM_LANDMARKS, AFS_NUM_PARTICLES, afsApplyBestPose(), afsFindBestPose(), afsGuessState(), afsParticleError(), afsParticleInit(), dx, _afsParticle::landmarks, _afsLandmarkLoc::mean, Particles, _afsParticle::pose, _afsLandmarkLoc::state, _afsPose::theta, _afsLandmarkLoc::variance, _afsXY::x, _afsPose::x, _afsXY::y, and _afsPose::y.

8.5 afsMeasurementUpdate.cc File Reference

```
#include "afsParticle.h"
#include "afsTriangulator.h"
#include "afsUtility.h"
#include <math.h>
#include <stdio.h>
```

Include dependency graph for afsMeasurementUpdate.cc:



Functions

- void [doPriming](#) ([afsParticle](#) *p, int landmark, double theta)
- void [doKalmanUpdate](#) ([afsParticle](#) *p, int landmark, double theta)
- void [afsMeasurementUpdate](#) ([afsParticle](#) *p, int landmark, double theta)

8.5.1 Function Documentation

8.5.1.1 void afsMeasurementUpdate ([afsParticle](#) *p, int landmark, double theta)

Definition at line 23 of file afsMeasurementUpdate.cc.

References [doKalmanUpdate\(\)](#), [doPriming\(\)](#), [_afsParticle::landmarks](#), [_afsParticle::pose](#), [_afsLandmarkLoc::state](#), and [_afsPose::theta](#).

8.5.1.2 void doKalmanUpdate ([afsParticle](#) *p, int landmark, double theta)

Definition at line 80 of file afsMeasurementUpdate.cc.

References [AFS_MEASURE_VARIANCE](#), [AFS_VARIANCE_MULTIPLIER](#), [find_dtheta\(\)](#), [_afsParticle::gotweight](#), [_afsParticle::landmarks](#), [_afsLandmarkLoc::mean](#), [_-](#)

afsParticle::pose, R, _afsLandmarkLoc::variance, _afsParticle::weight, _afsPose::x, and _afsPose::y.

8.5.1.3 void doPriming ([afsParticle](#) * *p*, int *landmark*, double *theta*)

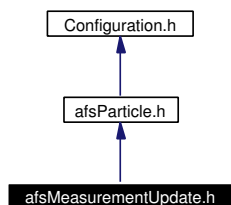
Definition at line 39 of file afsMeasurementUpdate.cc.

References [afsTriangulator\(\)](#), [_afsLastObservation::empty](#), [_afsParticle::gotweight](#), [_afsParticle::landmarks](#), [_afsParticle::pose](#), [_afsLandmarkLoc::priming](#), [_afsLandmarkLoc::state](#), [_afsLastObservation::theta](#), [_afsPose::x](#), [_afsLastObservation::x](#), [_afsPose::y](#), and [_afsLastObservation::y](#).

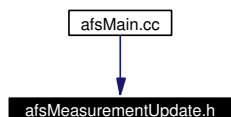
8.6 afsMeasurementUpdate.h File Reference

```
#include "afsParticle.h"
```

Include dependency graph for afsMeasurementUpdate.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [afsMeasurementUpdate](#) ([afsParticle](#) *p, int landmark, double theta)

8.6.1 Function Documentation

8.6.1.1 void afsMeasurementUpdate ([afsParticle](#) *p, int *landmark*, double *theta*)

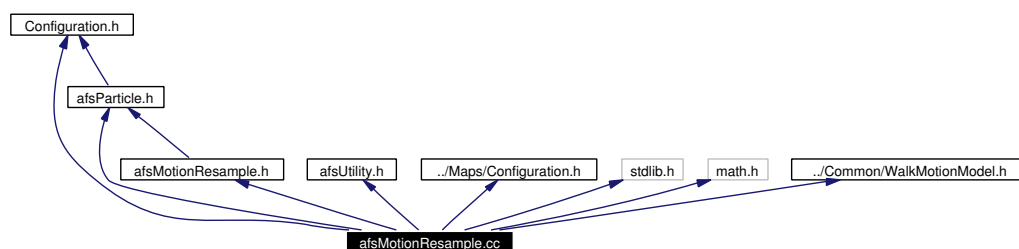
Definition at line 23 of file afsMeasurementUpdate.cc.

References [doKalmanUpdate\(\)](#), [doPriming\(\)](#), [_afsParticle::landmarks](#), [_afsParticle::pose](#), [_afsLandmarkLoc::state](#), and [_afsPose::theta](#).

8.7 afsMotionResample.cc File Reference

```
#include "afsMotionResample.h"
#include "afsParticle.h"
#include "afsUtility.h"
#include "Configuration.h"
#include "../Maps/Configuration.h"
#include <stdlib.h>
#include <math.h>
#include "../Common/WalkMotionModel.h"
```

Include dependency graph for afsMotionResample.cc:



Functions

- void [afsMotionResample](#) ([afsParticle](#) *p, double [dx](#), double [dy](#), double [da](#), unsigned int [time](#))

8.7.1 Function Documentation

8.7.1.1 void afsMotionResample ([afsParticle](#) *p, double [dx](#), double [dy](#), double [da](#), unsigned int [time](#))

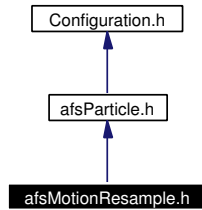
Definition at line 36 of file afsMotionResample.cc.

References [AFS_PERTURB_DA_MEAN](#), [AFS_PERTURB_DA_VARIANCE](#), [AFS_PERTURB_DX_MEAN](#), [AFS_PERTURB_DX_VARIANCE](#), [AFS_PERTURB_DY_MEAN](#), [AFS_PERTURB_DY_VARIANCE](#), [AIBO_TURN_PIVOT_DIST](#), [da](#), [dx](#), [normRand\(\)](#), [_afsParticle::pose](#), [_afsPose::theta](#), [time](#), [WalkMotionModel\(\)](#), [_afsPose::x](#), and [_afsPose::y](#).

8.8 afsMotionResample.h File Reference

```
#include "afsParticle.h"
```

Include dependency graph for afsMotionResample.h:



This graph shows which files directly or indirectly include this file:



Functions

- void **afsMotionResample** (**afsParticle** *p, double **dx**, double dy, double **da**, unsigned int **time**)

8.8.1 Function Documentation

8.8.1.1 void afsMotionResample (**afsParticle** *p, double **dx**, double **dy**, double **da**, unsigned int **time**)

Definition at line 36 of file afsMotionResample.cc.

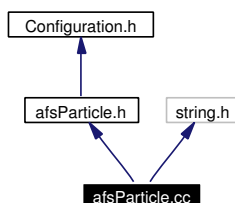
References AFS_PERTURB_DA_MEAN, AFS_PERTURB_DA_VARIANCE, AFS_PERTURB_DX_MEAN, AFS_PERTURB_DX_VARIANCE, AFS_PERTURB_DY_MEAN, AFS_PERTURB_DY_VARIANCE, AIBO_TURN_PIVOT_DIST, da, dx, normRand(), _afsParticle::pose, _afsPose::theta, time, WalkMotionModel(), _afsPose::x, and _afsPose::y.

8.9 afsParticle.cc File Reference

```
#include "afsParticle.h"
```

```
#include <string.h>
```

Include dependency graph for afsParticle.cc:



Functions

- void [afsParticleInit](#) ([afsParticle](#) *p)
- void [afsParticleCopy](#) ([afsParticle](#) *old, [afsParticle](#) *neu)

8.9.1 Function Documentation

8.9.1.1 void afsParticleCopy ([afsParticle](#) * old, [afsParticle](#) * neu)

Copy an afsParticle

Definition at line 30 of file afsParticle.cc.

8.9.1.2 void afsParticleInit ([afsParticle](#) * p)

Initialize a new afsParticle

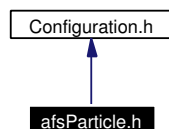
Definition at line 10 of file afsParticle.cc.

References `AFS_NUM_LANDMARKS`, `_afsLastObservation::empty`, `_afsParticle::gotweight`, `_afsParticle::landmarks`, `_afsLandmarkLoc::mean`, `_afsParticle::pose`, `_afsLandmarkLoc::priming`, `_afsLandmarkLoc::state`, `_afsPose::theta`, `_afsLandmarkLoc::variance`, `_afsParticle::weight`, `_afsPose::x`, and `_afsPose::y`.

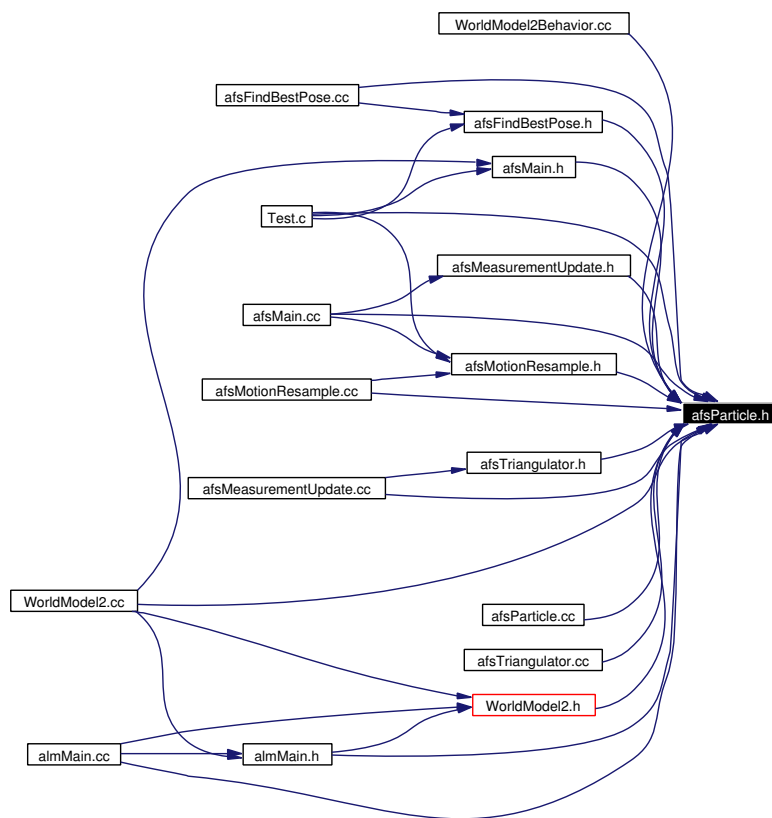
8.10 afsParticle.h File Reference

```
#include "Configuration.h"
```

Include dependency graph for afsParticle.h:



This graph shows which files directly or indirectly include this file:



Compounds

- struct [_afsLandmarkLoc](#)
- struct [_afsLastObservation](#)
- struct [_afsParticle](#)
- struct [_afsPose](#)

Typedefs

- typedef [_afsPose](#) [afsPose](#)
- typedef [_afsLastObservation](#) [afsLastObservation](#)
- typedef [_afsLandmarkLoc](#) [afsLandmarkLoc](#)
- typedef [_afsParticle](#) [afsParticle](#)

Functions

- void [afsParticleInit](#) ([afsParticle](#) *p)
- void [afsParticleCopy](#) ([afsParticle](#) *old, [afsParticle](#) *neu)

8.10.1 Typedef Documentation

8.10.1.1 typedef struct [_afsLandmarkLoc](#) [afsLandmarkLoc](#)

This structure is used within [afsParticle](#) structures to encode information about landmark location

8.10.1.2 typedef struct [_afsLastObservation](#) [afsLastObservation](#)

This structure is used within [afsLandmarkLoc](#) during the initialization phase, when it is necessary to triangulate the location of the landmark before using regular FastSLAM techniques.

8.10.1.3 typedef struct [_afsParticle](#) [afsParticle](#)

This structure contains data for each particle used by the particle filter. Since we have a fixed number of landmarks, we simply fix the number of [afsLandmarkLoc](#) structures inside.

8.10.1.4 typedef struct [_afsPose](#) [afsPose](#)

This structure contains a robot pose. Pretty simple.

8.10.2 Function Documentation

8.10.2.1 void afsParticleCopy (afsParticle * *old*, afsParticle * *neu*)

Copy an afsParticle

Definition at line 30 of file afsParticle.cc.

8.10.2.2 void afsParticleInit (afsParticle * *p*)

Initialize a new afsParticle

Definition at line 10 of file afsParticle.cc.

References AFS_NUM_LANDMARKS, _afsLastObservation::empty, _afsParticle::gotweight, _afsParticle::landmarks, _afsLandmarkLoc::mean, _afsParticle::pose, _afsLandmarkLoc::priming, _afsLandmarkLoc::state, _afsPose::theta, _afsLandmarkLoc::variance, _afsParticle::weight, _afsPose::x, and _afsPose::y.

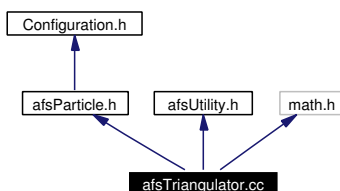
8.11 afsTriangulator.cc File Reference

```
#include "afsParticle.h"
```

```
#include "afsUtility.h"
```

```
#include <math.h>
```

Include dependency graph for afsTriangulator.cc:



Functions

- void [triangulate](#) (double *x1*, double *y1*, double *theta1*, double *x2*, double *y2*, double *theta2*, double **pos_x*, double **pos_y*)
- int [afsTriangulator](#) (double *x1*, double *y1*, double *theta1*, double *x2*, double *y2*, double *theta2*, [afsLandmarkLoc](#) **landmark*)

8.11.1 Function Documentation

8.11.1.1 int [afsTriangulator](#) (double *x1*, double *y1*, double *theta1*, double *x2*, double *y2*, double *theta2*, [afsLandmarkLoc](#) * *landmark*)

Definition at line 53 of file afsTriangulator.cc.

References [AFS_COVARIANCE_FUDGE](#), [AFS_MAX_TRIANG_ANGLE](#), [AFS_MEASURE_VARIANCE](#), [AFS_MIN_TRIANG_ANGLE](#), [find_dtheta\(\)](#), [_afsLandmarkLoc::mean](#), [triangulate\(\)](#), and [_afsLandmarkLoc::variance](#).

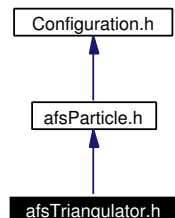
8.11.1.2 void [triangulate](#) (double *x1*, double *y1*, double *theta1*, double *x2*, double *y2*, double *theta2*, double **pos_x*, double **pos_y*)

Definition at line 26 of file afsTriangulator.cc.

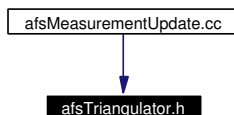
8.12 afsTriangulator.h File Reference

```
#include "afsParticle.h"
```

Include dependency graph for afsTriangulator.h:



This graph shows which files directly or indirectly include this file:



Functions

- int [afsTriangulator](#) (double *x1*, double *y1*, double *theta1*, double *x2*, double *y2*, double *theta2*, [afsLandmarkLoc](#) *landmark)

8.12.1 Function Documentation

8.12.1.1 int [afsTriangulator](#) (double *x1*, double *y1*, double *theta1*, double *x2*, double *y2*, double *theta2*, [afsLandmarkLoc](#) * *landmark*)

Definition at line 53 of file afsTriangulator.cc.

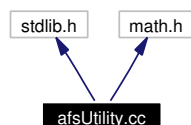
References [AFS_COVARIANCE_FUDGE](#), [AFS_MAX_TRIANG_ANGLE](#), [AFS_MEASURE_VARIANCE](#), [AFS_MIN_TRIANG_ANGLE](#), [find_dtheta\(\)](#), [_afsLandmarkLoc::mean](#), [triangulate\(\)](#), and [_afsLandmarkLoc::variance](#).

8.13 afsUtility.cc File Reference

```
#include <stdlib.h>
```

```
#include <math.h>
```

Include dependency graph for afsUtility.cc:



Functions

- double [find_dtheta](#) (double th1, double th2)
- double [normRand](#) ()

8.13.1 Function Documentation

8.13.1.1 double find_dtheta (double *th1*, double *th2*)

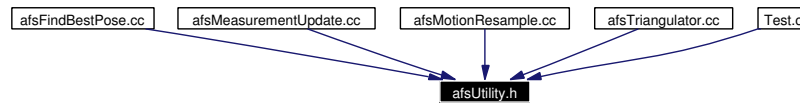
Definition at line 16 of file afsUtility.cc.

8.13.1.2 double normRand ()

Definition at line 27 of file afsUtility.cc.

8.14 afsUtility.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- double [find_dtheta](#) (double th1, double th2)
- double [normRand](#) ()

8.14.1 Function Documentation

8.14.1.1 double [find_dtheta](#) (double *th1*, double *th2*)

Definition at line 16 of file afsUtility.cc.

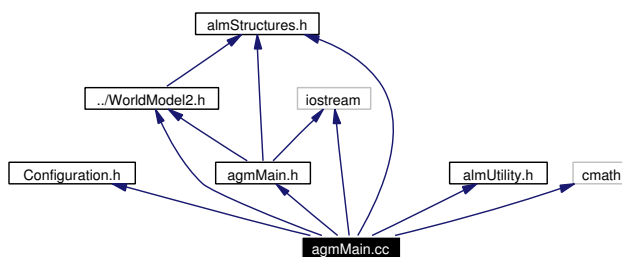
8.14.1.2 double [normRand](#) ()

Definition at line 27 of file afsUtility.cc.

8.15 agmMain.cc File Reference

```
#include "Configuration.h"
#include "agmMain.h"
#include "almStructures.h"
#include "almUtility.h"
#include "../WorldModel2.h"
#include <cmath>
#include <iostream>
```

Include dependency graph for agmMain.cc:



Variables

- `hm_cell GM` [GM.CELL_COUNT]

8.15.1 Variable Documentation

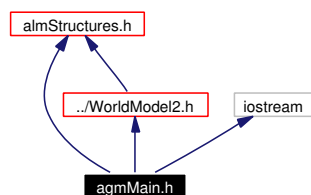
8.15.1.1 `hm_cell GM`[GM.CELL_COUNT]

Definition at line 29 of file `agmMain.cc`.

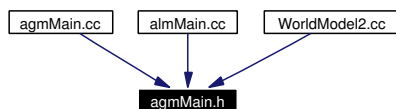
8.16 agmMain.h File Reference

```
#include "almStructures.h"  
#include "../WorldModel2.h"  
#include <iostream>
```

Include dependency graph for agmMain.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [AGM](#)

8.17 Aibo3DControllerBehavior.h File Reference

8.17.1 Detailed Description

Defines [Aibo3DControllerBehavior](#), which listens to commands from the Aibo3D gui and shows current state.

Author:

alokl (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.5

State

Rel

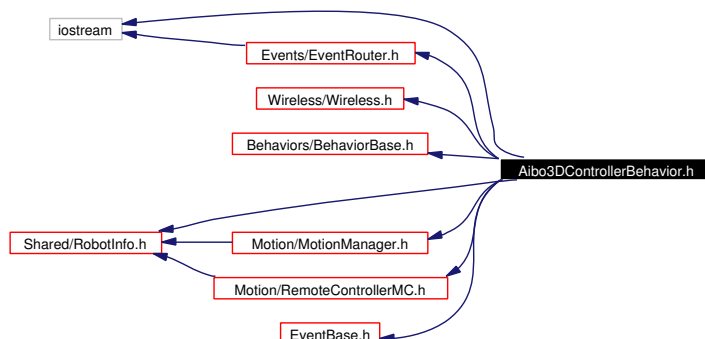
Date

2003/07/07 01:00:07

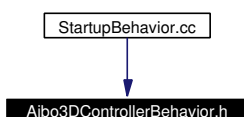
Definition in file [Aibo3DControllerBehavior.h](#).

```
#include <iostream>
#include "Wireless/Wireless.h"
#include "Behaviors/BehaviorBase.h"
#include "Motion/MotionManager.h"
#include "Motion/RemoteControllerMC.h"
#include "Events/EventRouter.h"
#include "Events/EventBase.h"
#include "Shared/RobotInfo.h"
```

Include dependency graph for Aibo3DControllerBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [Aibo3DControllerBehavior](#)
Listens to aibo3d control commands coming in from the command port.

Functions

- int [aibo3dcontrollercmd_callback](#) (char *buf, int bytes)
gets input from the GUI

Variables

- [Aibo3DControllerBehavior](#) * [aibo3dControllerBehavior](#) = NULL
so aibo3dcontrollercmd_callback knows where to send the input from the GUI

8.17.2 Function Documentation

8.17.2.1 `int aibo3dcontrollercmd_callback (char * buf, int bytes)`

gets input from the GUI

Definition at line 108 of file Aibo3DControllerBehavior.h.

References `aibo3dControllerBehavior`, and `Aibo3DControllerBehavior::registerData()`.

8.17.3 Variable Documentation

8.17.3.1 `Aibo3DControllerBehavior* aibo3dControllerBehavior = NULL`

so `aibo3dcontrollercmd_callback` knows where to send the input from the GUI

Definition at line 19 of file Aibo3DControllerBehavior.h.

8.18 Aibo3DMonitorBehavior.h File Reference

8.18.1 Detailed Description

Defines [Aibo3DMonitorBehavior](#), which sends current pose to Aibo3D GUI, ignores incoming commands.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

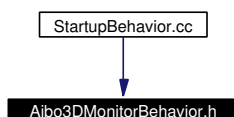
Rel

Date

2003/07/07 01:00:07

Definition in file [Aibo3DMonitorBehavior.h](#).

This graph shows which files directly or indirectly include this file:



Compounds

- class [Aibo3DMonitorBehavior](#)
Sends current pose to Aibo3D GUI, ignores incoming commands.

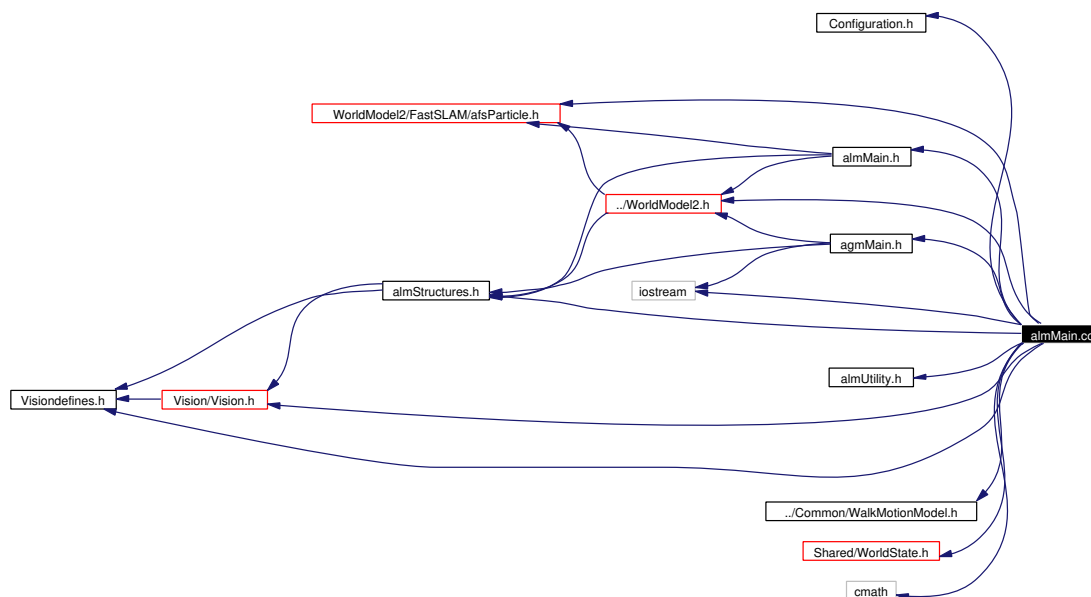
8.19 almMain.cc File Reference

```

#include "Configuration.h"
#include "almMain.h"
#include "almUtility.h"
#include "almStructures.h"
#include "agmMain.h"
#include "../Common/WalkMotionModel.h"
#include ../FastSLAM/afsParticle.h"
#include ../WorldModel2.h"
#include "Shared/WorldState.h"
#include "Vision/Vision.h"
#include "Vision/Visiondefines.h"
#include <cmath>
#include <iostream>

```

Include dependency graph for almMain.cc:



Defines

- `#define IROORDIST WorldState::IROORDist`
- `#define SQRT_2_PI 2.5066282746310002416`

Variables

- `hm_cell HMs [2][HM_CELL_COUNT]`
- `hm_cell * HM`
- `dm_cell DMs [2][DM_CELL_COUNT]`
- `dm_cell * DM`

8.19.1 Define Documentation

8.19.1.1 `#define IROORDIST WorldState::IROORDist`

Definition at line 35 of file almMain.cc.

8.19.1.2 `#define SQRT_2_PI 2.5066282746310002416`

Definition at line 42 of file almMain.cc.

8.19.2 Variable Documentation

8.19.2.1 `dm_cell* DM`

Definition at line 56 of file almMain.cc.

8.19.2.2 `dm_cell DMs[2][DM_CELL_COUNT]`

Definition at line 55 of file almMain.cc.

8.19.2.3 `hm_cell* HM`

Definition at line 51 of file almMain.cc.

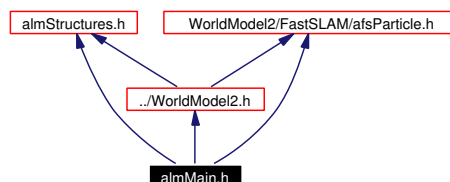
8.19.2.4 `hm_cell HMs[2][HM_CELL_COUNT]`

Definition at line 50 of file almMain.cc.

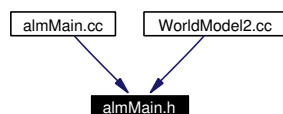
8.20 almMain.h File Reference

```
#include "almStructures.h"  
#include "../WorldModel2.h"  
#include "../FastSLAM/afsParticle.h"
```

Include dependency graph for almMain.h:



This graph shows which files directly or indirectly include this file:



Compounds

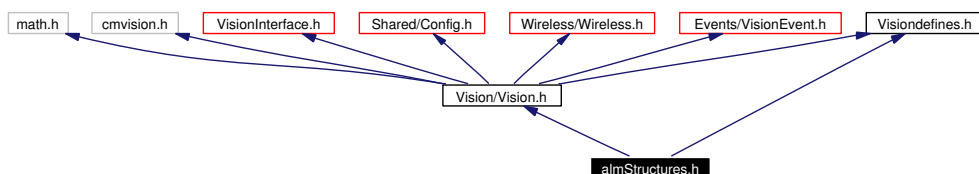
- class [ALM](#)

8.21 almStructures.h File Reference

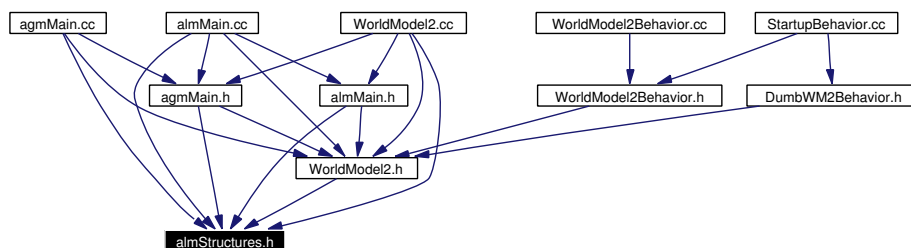
```
#include "Vision/Vision.h"
```

```
#include "Vision/Visiondefines.h"
```

Include dependency graph for almStructures.h:



This graph shows which files directly or indirectly include this file:



Compounds

- struct [_dm.cell](#)
A cell for the spherical depth map.
- struct [_hm.cell](#)
A cell for the horizontal height map.
- struct [dmPickCluster](#)
Picks cluster membership out of the SDM. For completeness—not use!
- struct [dmPickColor](#)
Picks color measurements out of the SDM.
- struct [dmPickConfidence](#)

Picks confidence values out of the SDM.

- struct [dmPickDepth](#)

Picks depth measurements out of the SDM.

- struct [dmPicker](#)

Abstract base class for depth map data pickers. Implementations available.

- struct [hmPickCluster](#)

Picks cluster membership out of the HHM or GHM. For completeness—not use!

- struct [hmPickColor](#)

Picks color measurements out of the HHM or GHM.

- struct [hmPickConfidence](#)

Picks confidence values out of the HHM or GHM.

- struct [hmPicker](#)

Abstract base class for height map data pickers. Implementations available.

- struct [hmPickHeight](#)

Picks height measurements out of the HHM or GHM.

- struct [hmPickTrav](#)

Picks traversability values out of the HHM or GHM.

Typedefs

- typedef int [colortype](#)
- typedef [_dm_cell](#) [dm_cell](#)

A cell for the spherical depth map.

- typedef [_hm_cell](#) [hm_cell](#)

A cell for the horizontal height map.

8.21.1 Typedef Documentation

8.21.1.1 typedef int [colortype](#)

Definition at line 19 of file almStructures.h.

8.21.1.2 typedef struct [_dm_cell](#) [dm_cell](#)

A cell for the spherical depth map.

8.21.1.3 typedef struct [_hm_cell](#) [hm_cell](#)

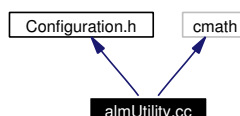
A cell for the horizontal height map.

8.22 almUtility.cc File Reference

```
#include "Configuration.h"
```

```
#include <cmath>
```

Include dependency graph for almUtility.cc:



Functions

- void [dm_index2angles](#) (int index, double &azimuth, double &altitude)
- bool [angles2dm_index](#) (double azimuth, double altitude, int &index)
- void [hm_index2xy](#) (int index, double &x, double &y)
- bool [xy2hm_index](#) (double x, double y, int &index)
- void [gm_index2xy](#) (int index, double &x, double &y)
- bool [xy2gm_index](#) (double x, double y, int &index)
- void [head_range2xyz](#) (double depth, double pan, double tilt, double &x, double &y, double &z)
- void [xyz2neck_range](#) (double x, double y, double z, double &depth, double &azimuth, double &altitude)
- void [neck_range2xyz](#) (double depth, double azimuth, double altitude, double &x, double &y, double &z)

8.22.1 Function Documentation

8.22.1.1 bool angles2dm_index (double azimuth, double altitude, int & index)

Definition at line 31 of file almUtility.cc.

References ALM_DM_BOTTOM, ALM_DM_H_SIZE, ALM_DM_LEFT, ALM_DM_RIGHT, ALM_DM_TOP, ALM_DM_V_SIZE, and dx.

8.22.1.2 void dm_index2angles (int index, double & azimuth, double & altitude)

Definition at line 18 of file almUtility.cc.

References ALM_DM_BOTTOM, ALM_DM_H_SIZE, ALM_DM_LEFT, ALM_DM_RIGHT, ALM_DM_TOP, and ALM_DM_V_SIZE.

8.22.1.3 void gm_index2xy (int *index*, double & *x*, double & *y*)

Definition at line 94 of file almUtility.cc.

References AGM_BOTTOM, AGM_H_SIZE, AGM_LEFT, AGM_RIGHT, AGM_TOP, and AGM_V_SIZE.

8.22.1.4 void head_range2xyz (double *depth*, double *pan*, double *tilt*, double & *x*, double & *y*, double & *z*)

Definition at line 134 of file almUtility.cc.

References AIBO_HEAD_LENGTH, AIBO_NECK_HEIGHT, and AIBO_TILT_PIVOT_HEIGHT.

8.22.1.5 void hm_index2xy (int *index*, double & *x*, double & *y*)

Definition at line 55 of file almUtility.cc.

References ALM_HM_RADIUS, and ALM_HM_SIZE.

8.22.1.6 void neck_range2xyz (double *depth*, double *azimuth*, double *altitude*, double & *x*, double & *y*, double & *z*)

Definition at line 181 of file almUtility.cc.

References AIBO_TILT_PIVOT_HEIGHT.

8.22.1.7 bool xy2gm_index (double *x*, double *y*, int & *index*)

Definition at line 108 of file almUtility.cc.

References AGM_BOTTOM, AGM_H_SIZE, AGM_LEFT, AGM_RIGHT, AGM_TOP, AGM_V_SIZE, and dx.

8.22.1.8 bool xy2hm_index (double *x*, double *y*, int & *index*)

Definition at line 71 of file almUtility.cc.

References ALM_HM_RADIUS, ALM_HM_SIZE, and dx.

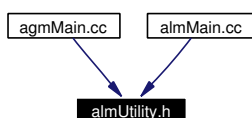
8.22.1.9 void xyz2neck_range (double *x*, double *y*, double *z*, double & *depth*, double & *azimuth*, double & *altitude*)

Definition at line 169 of file almUtility.cc.

References AIBO_TILT_PIVOT_HEIGHT.

8.23 almUtility.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void [dm_index2angles](#) (int index, double &azimuth, double &altitude)
- bool [angles2dm_index](#) (double azimuth, double altitude, int &index)
- void [hm_index2xy](#) (int index, double &x, double &y)
- bool [xy2hm_index](#) (double x, double y, int &index)
- void [gm_index2xy](#) (int index, double &x, double &y)
- bool [xy2gm_index](#) (double x, double y, int &index)
- void [head_range2xyz](#) (double depth, double pan, double tilt, double &x, double &y, double &z)
- void [xyz2neck_range](#) (double x, double y, double z, double &depth, double &azimuth, double &altitude)
- void [neck_range2xyz](#) (double depth, double azimuth, double altitude, double &x, double &y, double &z)

8.23.1 Function Documentation

8.23.1.1 bool angles2dm_index (double *azimuth*, double *altitude*, int & *index*)

Definition at line 31 of file almUtility.cc.

References ALM_DM_BOTTOM, ALM_DM_H_SIZE, ALM_DM_LEFT, ALM_DM_RIGHT, ALM_DM_TOP, ALM_DM_V_SIZE, and dx.

8.23.1.2 void dm_index2angles (int *index*, double & *azimuth*, double & *altitude*)

Definition at line 18 of file almUtility.cc.

References ALM_DM_BOTTOM, ALM_DM_H_SIZE, ALM_DM_LEFT, ALM_DM_RIGHT, ALM_DM_TOP, and ALM_DM_V_SIZE.

8.23.1.3 void gm_index2xy (int *index*, double & *x*, double & *y*)

Definition at line 94 of file almUtility.cc.

References AGM_BOTTOM, AGM_H_SIZE, AGM_LEFT, AGM_RIGHT, AGM_TOP, and AGM_V_SIZE.

8.23.1.4 void head_range2xyz (double *depth*, double *pan*, double *tilt*, double & *x*, double & *y*, double & *z*)

Definition at line 134 of file almUtility.cc.

References AIBO_HEAD_LENGTH, AIBO_NECK_HEIGHT, and AIBO_TILT_PIVOT_HEIGHT.

8.23.1.5 void hm_index2xy (int *index*, double & *x*, double & *y*)

Definition at line 55 of file almUtility.cc.

References ALM_HM_RADIUS, and ALM_HM_SIZE.

8.23.1.6 void neck_range2xyz (double *depth*, double *azimuth*, double *altitude*, double & *x*, double & *y*, double & *z*)

Definition at line 181 of file almUtility.cc.

References AIBO_TILT_PIVOT_HEIGHT.

8.23.1.7 bool xy2gm_index (double *x*, double *y*, int & *index*)

Definition at line 108 of file almUtility.cc.

References AGM_BOTTOM, AGM_H_SIZE, AGM_LEFT, AGM_RIGHT, AGM_TOP, AGM_V_SIZE, and dx.

8.23.1.8 bool xy2hm_index (double *x*, double *y*, int & *index*)

Definition at line 71 of file almUtility.cc.

References ALM_HM_RADIUS, ALM_HM_SIZE, and dx.

8.23.1.9 void xyz2neck_range (double *x*, double *y*, double *z*, double & *depth*, double & *azimuth*, double & *altitude*)

Definition at line 169 of file almUtility.cc.

References AIBO_TILT_PIVOT_HEIGHT.

8.24 AutoGetupBehavior.h File Reference

8.24.1 Detailed Description

Defines [AutoGetupBehavior](#), a little background behavior to keep the robot on its feet.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.8

State

Rel

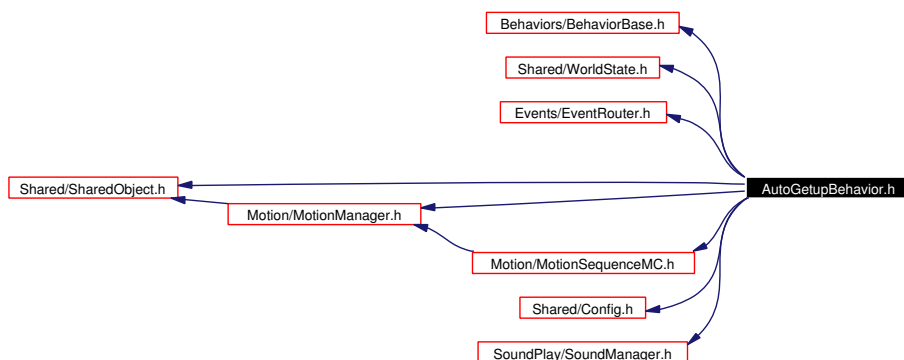
Date

2003/06/10 00:53:48

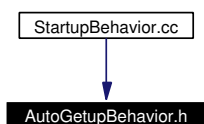
Definition in file [AutoGetupBehavior.h](#).

```
#include "Behaviors/BehaviorBase.h"
#include "Shared/WorldState.h"
#include "Events/EventRouter.h"
#include "Shared/SharedObject.h"
#include "Motion/MotionManager.h"
#include "Motion/MotionSequenceMC.h"
#include "Shared/Config.h"
#include "SoundPlay/SoundManager.h"
```

Include dependency graph for AutoGetupBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [AutoGetupBehavior](#)
a little background behavior to keep the robot on its feet

8.25 BanditMachine.h File Reference

8.25.1 Detailed Description

Defines [BanditMachine](#), A state machine for playing k-armed bandit.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.9

State

Rel

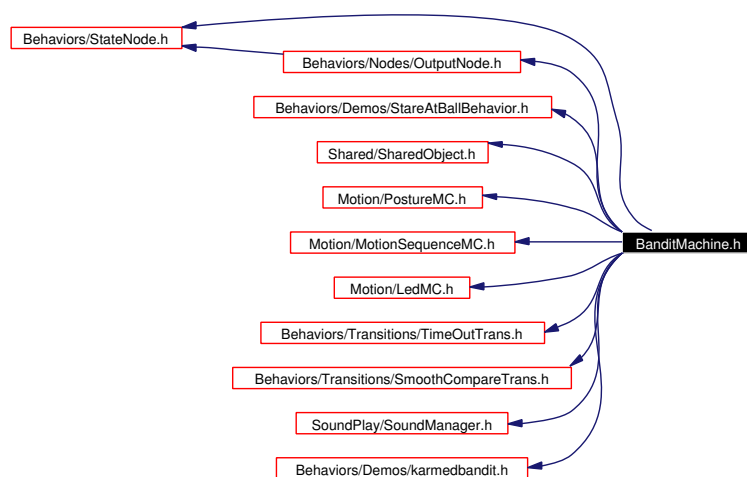
Date

2003/06/12 23:41:39

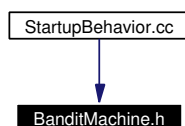
Definition in file [BanditMachine.h](#).

```
#include "Behaviors/StateNode.h"
#include "Behaviors/Demos/StareAtBallBehavior.h"
#include "Shared/SharedObject.h"
#include "Motion/PostureMC.h"
#include "Motion/MotionSequenceMC.h"
#include "Motion/LedMC.h"
#include "Behaviors/Transitions/TimeOutTrans.h"
#include "Behaviors/Transitions/SmoothCompareTrans.h"
#include "Behaviors/Nodes/OutputNode.h"
#include "SoundPlay/SoundManager.h"
#include "Behaviors/Demos/karmedbandit.h"
```

Include dependency graph for BanditMachine.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [BanditMachine](#)
Plays K -armed bandit.
- class [BanditMachine.DecideNode](#)
uses one of the algorithms in [karmedbandit.h](#) to decide which paw to press next
- class [BanditMachine.PressNode](#)
This node is used to move a paw down using a [MotionSequenceMC](#).
- class [BanditMachine.WaitNode](#)
Waits to see if a reward is received, lights up LEDs to let the user know.

8.26 BatteryCheckControl.h File Reference

8.26.1 Detailed Description

Defines [BatteryCheckControl](#), which will spew a power report to stdout upon activation.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.4

State

Rel

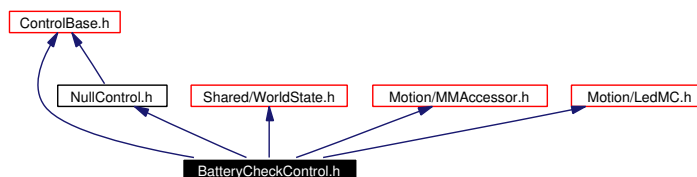
Date

2003/06/09 08:05:12

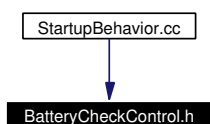
Definition in file [BatteryCheckControl.h](#).

```
#include "ControlBase.h"
#include "Shared/WorldState.h"
#include "Motion/MMAccessor.h"
#include "Motion/LedMC.h"
#include "NullControl.h"
```

Include dependency graph for BatteryCheckControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [BatteryCheckControl](#)
when activated, this will print a battery report to stdout and light up LEDs to specify power level

8.27 BatteryMonitorBehavior.h File Reference

8.27.1 Detailed Description

Defines [BatteryMonitorBehavior](#), a background behavior to trigger BatteryMonitorMC to warn when the power is low.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.10

State

Rel

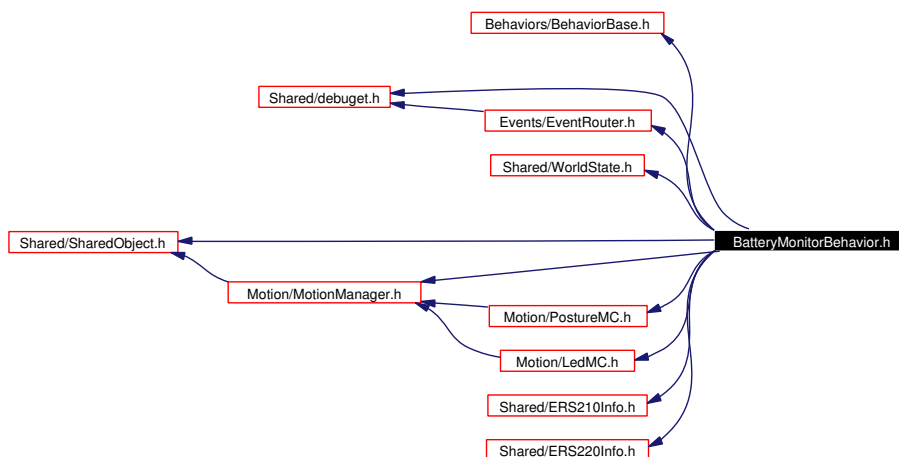
Date

2003/06/12 18:06:10

Definition in file [BatteryMonitorBehavior.h](#).

```
#include "Behaviors/BehaviorBase.h"
#include "Shared/debuget.h"
#include "Shared/WorldState.h"
#include "Events/EventRouter.h"
#include "Shared/SharedObject.h"
#include "Motion/MotionManager.h"
#include "Motion/PostureMC.h"
#include "Motion/LedMC.h"
#include "Shared/ERS210Info.h"
#include "Shared/ERS220Info.h"
```

Include dependency graph for BatteryMonitorBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [BatteryMonitorBehavior](#)

A background behavior which will monitor the power level and flip the ears when appropriate on a 210, or blink the headlight if a 220.

8.28 BehaviorActivatorControl.h File Reference

8.28.1 Detailed Description

Defines [BehaviorActivatorControl](#), which can either start, stop, or toggle a behavior when activated.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.7

State

Rel

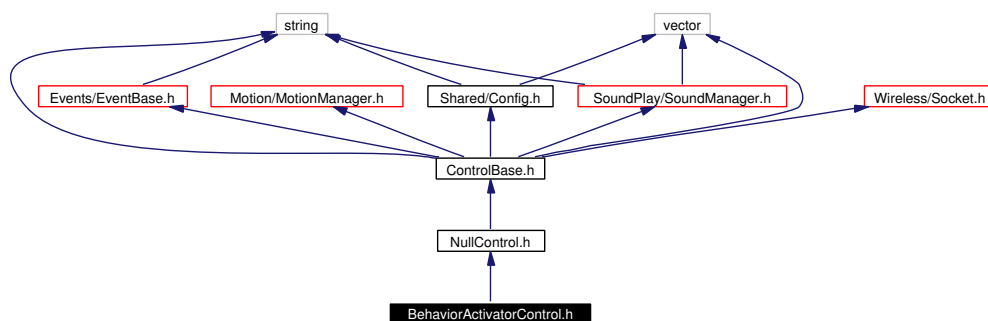
Date

2003/06/12 18:06:10

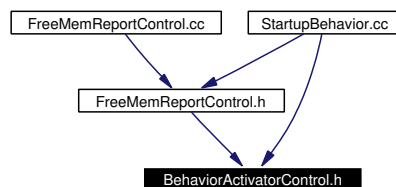
Definition in file [BehaviorActivatorControl.h](#).

```
#include "NullControl.h"
```

Include dependency graph for BehaviorActivatorControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [BehaviorActivatorControl](#)

Upon activation, will start, stop, or toggle a behavior.

8.29 BehaviorBase.h File Reference

8.29.1 Detailed Description

Defines [BehaviorBase](#) from which all Behaviors should inherit.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.5

State

Rel

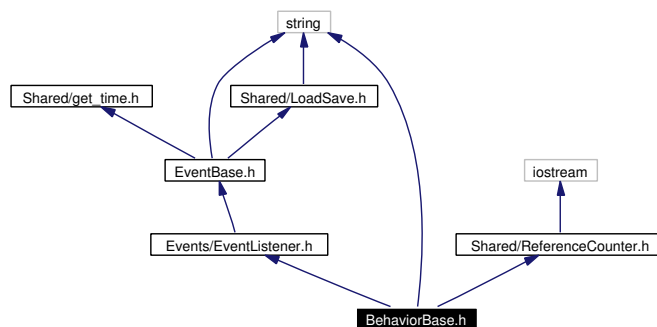
Date

2003/06/09 08:05:08

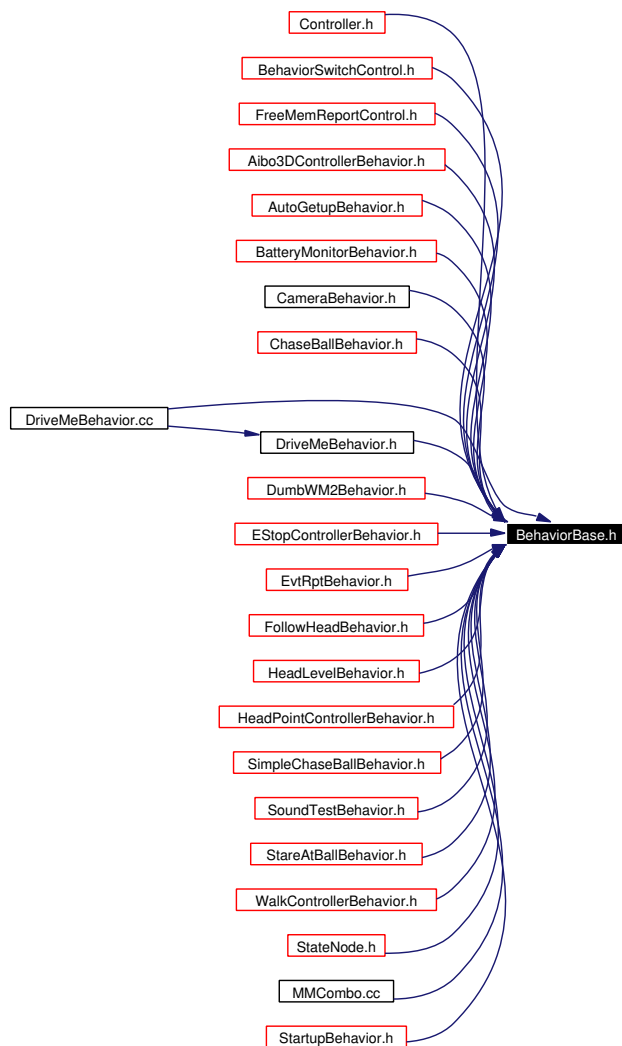
Definition in file [BehaviorBase.h](#).

```
#include "Events/EventListener.h"
#include "Shared/ReferenceCounter.h"
#include <string>
```

Include dependency graph for BehaviorBase.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [BehaviorBase](#)

The basis from which all other Behaviors should inherit.

8.30 BehaviorSwitchActivatorControl.h File Reference

8.30.1 Detailed Description

Defines [BehaviorSwitchActivatorControl](#), which will tell the specified [BehaviorSwitchControl](#) to start or stop the behavior.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

Date

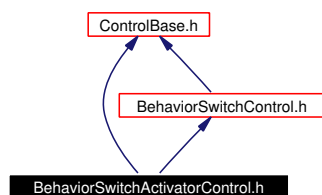
2003/06/12 18:06:10

Definition in file [BehaviorSwitchActivatorControl.h](#).

```
#include "ControlBase.h"
```

```
#include "BehaviorSwitchControl.h"
```

Include dependency graph for BehaviorSwitchActivatorControl.h:



Compounds

- class [BehaviorSwitchActivatorControl](#)

Upon activation, will tell the specified [BehaviorSwitchControl](#) to start or stop the behavior.

8.31 BehaviorSwitchControl.h File Reference

8.31.1 Detailed Description

Defines [BehaviorSwitchControl](#) and the BehaviorSwitch namespace - a control for turning behaviors on and off.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.7

State

Rel

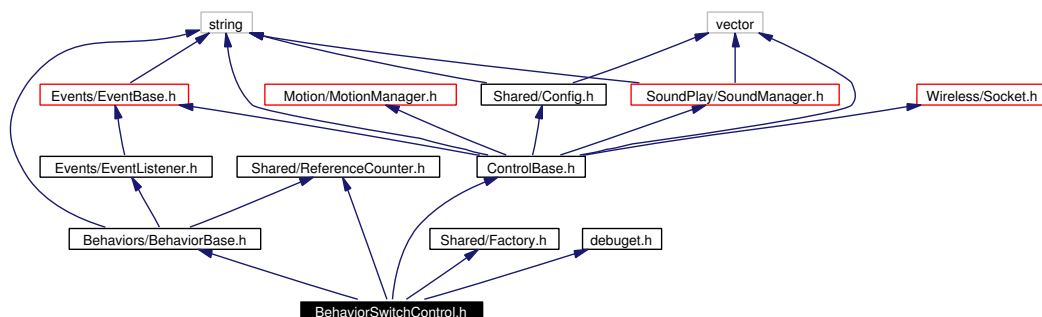
Date

2003/06/10 00:53:47

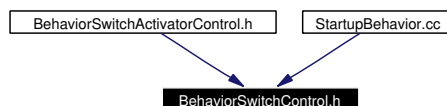
Definition in file [BehaviorSwitchControl.h](#).

```
#include "ControlBase.h"
#include "Behaviors/BehaviorBase.h"
#include "Shared/ReferenceCounter.h"
#include "Shared/Factory.h"
#include "Shared/debuget.h"
```

Include dependency graph for BehaviorSwitchControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [BehaviorSwitchControl](#)
Allows proper switching between major behaviors, calling DoStart and DoStop.
- class [BehaviorSwitchControlBase](#)
Holds some utility classes and functions for [BehaviorSwitchControl](#) which shouldn't be stored in a templated class.
- class [BehaviorSwitchControlBase.BehaviorGroup](#)
A simple utility class to allow the BehaviorSwitchControl's to be able to deactivate the current behavior when a new one becomes active.

8.32 CameraBehavior.h File Reference

8.32.1 Detailed Description

Defines [CameraBehavior](#), for taking pictures.

Author:

alokl (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.4

State

Rel

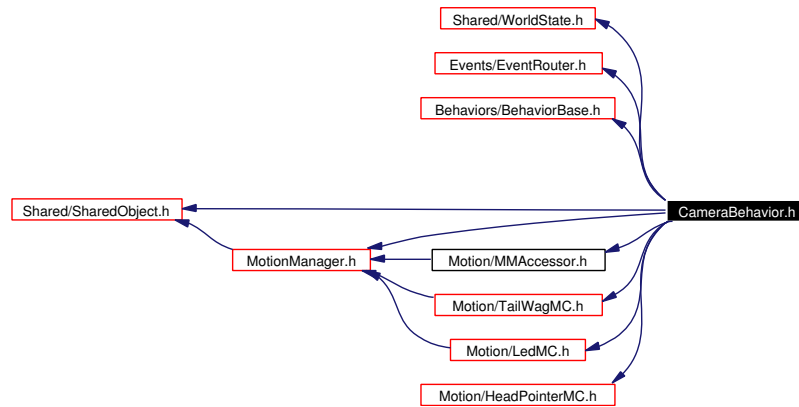
Date

2003/06/12 23:41:39

Definition in file [CameraBehavior.h](#).

```
#include "Shared/WorldState.h"
#include "Events/EventRouter.h"
#include "Behaviors/BehaviorBase.h"
#include "Motion/MMAccessor.h"
#include "Motion/MotionManager.h"
#include "Motion/HeadPointerMC.h"
#include "Motion/TailWagMC.h"
#include "Motion/LedMC.h"
#include "Shared/SharedObject.h"
```

Include dependency graph for CameraBehavior.h:



Compounds

- class [CameraBehavior](#)

Will take images and write to log file.

8.33 ChaseBallBehavior.cc File Reference

8.33.1 Detailed Description

Implements [ChaseBallBehavior](#), which runs around after whatever the dog sees.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.2

State

Rel

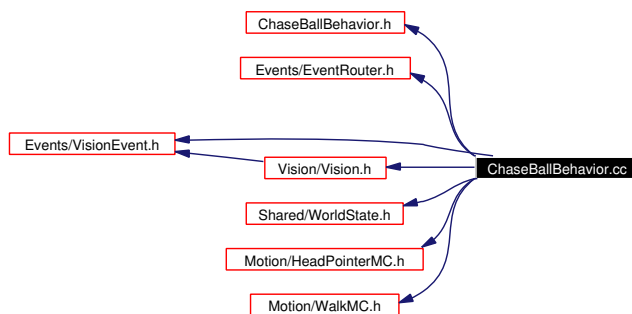
Date

2003/03/09 02:45:21

Definition in file [ChaseBallBehavior.cc](#).

```
#include "ChaseBallBehavior.h"
#include "Events/EventRouter.h"
#include "Events/VisionEvent.h"
#include "Shared/WorldState.h"
#include "Motion/HeadPointerMC.h"
#include "Motion/WalkMC.h"
#include "Vision/Vision.h"
```

Include dependency graph for ChaseBallBehavior.cc:



Functions

- double **DtoR** (double deg)
Converts degrees to radians.

8.33.2 Function Documentation

8.33.2.1 double **DtoR** (double *deg*) [`inline`]

Converts degrees to radians.

Definition at line 10 of file ChaseBallBehavior.cc.

8.34 ChaseBallBehavior.h File Reference

8.34.1 Detailed Description

Describes [ChaseBallBehavior](#), which runs around after whatever the dog sees.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

Date

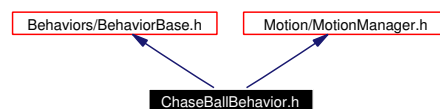
2003/06/05 17:03:15

Definition in file [ChaseBallBehavior.h](#).

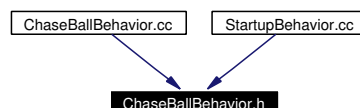
```
#include "Behaviors/BehaviorBase.h"
```

```
#include "Motion/MotionManager.h"
```

Include dependency graph for ChaseBallBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [ChaseBallBehavior](#)

A simple behavior to chase after any objects seen by the vision system.

8.35 CompareTrans.h File Reference

8.35.1 Detailed Description

Defines [CompareTrans](#), which causes a transition if a value (through a pointer) goes above a given value.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

Date

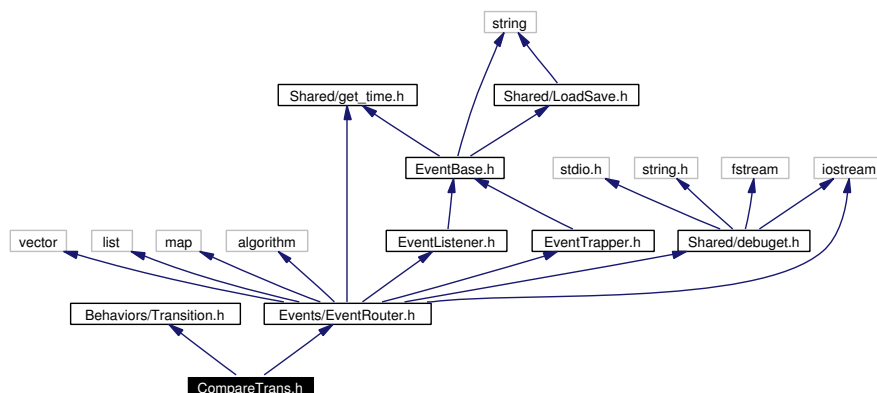
2003/03/09 02:45:22

Definition in file [CompareTrans.h](#).

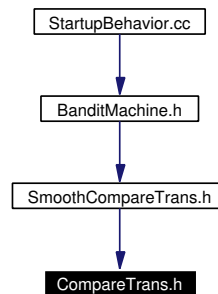
```
#include "Behaviors/Transition.h"
```

```
#include "Events/EventRouter.h"
```

Include dependency graph for CompareTrans.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [CompareTrans](#)
causes a transition if a value (through a pointer) goes above a given value

8.36 Config.cc File Reference

8.36.1 Detailed Description

Implements [Config](#), which provides global access to system configuration information.

Author:

alokl (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.14

State

Rel

Date

2003/07/07 04:32:47

Definition in file [Config.cc](#).

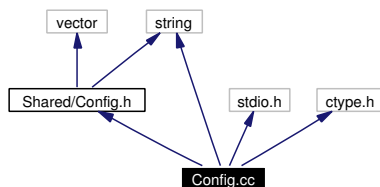
```
#include <Shared/Config.h>
```

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

```
#include <string>
```

```
#include <ctype.h>
```

Include dependency graph for Config.cc:



Variables

- [Config](#) * [config](#) = NULL
allows global access to current settings

8.36.2 Variable Documentation

8.36.2.1 `Config*` `config` = NULL

allows global access to current settings

Definition at line 6 of file Config.cc.

8.37 Config.h File Reference

8.37.1 Detailed Description

Describes [Config](#), which provides global access to system configuration information.

Author:

alokl (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.16

State

Rel

Date

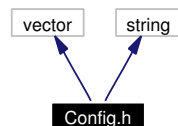
2003/07/07 04:32:47

Definition in file [Config.h](#).

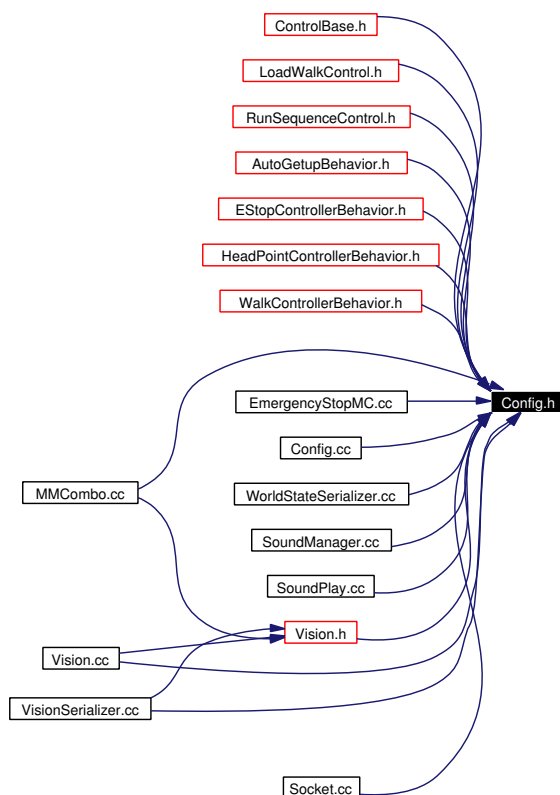
```
#include <vector>
```

```
#include <string>
```

Include dependency graph for Config.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [Config](#)
provides global access to system configuration information
- struct [Config.behaviors_config](#)
placeholder
- struct [Config.controller_config](#)
controller information
- struct [Config.main_config](#)
core functionality information
- struct [Config.motion_config](#)
motion information

- struct [Config.sound_config](#)
sound information
- struct [Config.vision_config](#)
vision information
- struct [Config.wireless_config](#)
wirless information
- struct [Config.worldmodel2_config](#)
world model information

Variables

- [Config * config](#)
allows global access to current settings

8.37.2 Variable Documentation

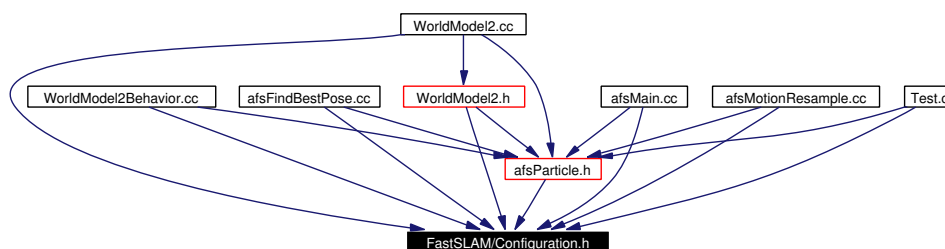
8.37.2.1 [Config* config](#)

allows global access to current settings

Definition at line 150 of file Config.h.

8.38 Configuration.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define [AFS_NUM_PARTICLES](#) 400
- #define [AFS_NUM_LANDMARKS](#) 12
- #define [AFS_PERTURB_DX_MEAN](#) 0
- #define [AFS_PERTURB_DX_VARIANCE](#) 0.1
- #define [AFS_PERTURB_DY_MEAN](#) 0
- #define [AFS_PERTURB_DY_VARIANCE](#) 0.1
- #define [AFS_PERTURB_DA_MEAN](#) 0
- #define [AFS_PERTURB_DA_VARIANCE](#) 0.1
- #define [AFS_MIN_TRIANG_ANGLE](#) 5*M_PI/180
- #define [AFS_MAX_TRIANG_ANGLE](#) 175*M_PI/180
- #define [AFS_MEASURE_VARIANCE](#) 4*M_PI/180
- #define [AFS_VARIANCE_MULTIPLIER](#) AFS_MEASURE_VARIANCE/4000
- #define [AFS_COVARIANCE_FUDGE](#) 15

8.38.1 Define Documentation

8.38.1.1 #define AFS_COVARIANCE_FUDGE 15

Definition at line 50 of file FastSLAM/Configuration.h.

8.38.1.2 #define AFS_MAX_TRIANG_ANGLE 175*M_PI/180

Definition at line 30 of file FastSLAM/Configuration.h.

8.38.1.3 #define AFS_MEASURE_VARIANCE 4*M_PI/180

Definition at line 46 of file FastSLAM/Configuration.h.

8.38.1.4 #define AFS_MIN_TRIANG_ANGLE 5*M_PI/180

Definition at line 29 of file FastSLAM/Configuration.h.

8.38.1.5 #define AFS_NUM_LANDMARKS 12

Definition at line 16 of file FastSLAM/Configuration.h.

8.38.1.6 #define AFS_NUM_PARTICLES 400

Definition at line 13 of file FastSLAM/Configuration.h.

8.38.1.7 #define AFS_PERTURB_DA_MEAN 0

Definition at line 23 of file FastSLAM/Configuration.h.

8.38.1.8 #define AFS_PERTURB_DA_VARIANCE 0.1

Definition at line 24 of file FastSLAM/Configuration.h.

8.38.1.9 #define AFS_PERTURB_DX_MEAN 0

Definition at line 19 of file FastSLAM/Configuration.h.

8.38.1.10 #define AFS_PERTURB_DX_VARIANCE 0.1

Definition at line 20 of file FastSLAM/Configuration.h.

8.38.1.11 #define AFS_PERTURB_DY_MEAN 0

Definition at line 21 of file FastSLAM/Configuration.h.

8.38.1.12 #define AFS_PERTURB_DY_VARIANCE 0.1

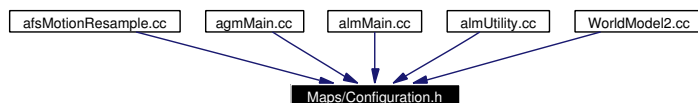
Definition at line 22 of file FastSLAM/Configuration.h.

8.38.1.13 `#define AFS_VARIANCE_MULTIPLIER`
`AFS_MEASURE_VARIANCE/4000`

Definition at line 47 of file FastSLAM/Configuration.h.

8.39 Configuration.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define [R\(item\)](#) (((double)(item))*M_PI/180)
- #define [ALM_DM_TAX](#) 0.8
- #define [ALM_HM_TAX](#) 0.9
- #define [ALM_GM_TAX](#) 0.9
- #define [ALM_DM_TOP](#) R(25.0)
- #define [ALM_DM_BOTTOM](#) R(-75.0)
- #define [ALM_DM_LEFT](#) R(85.0)
- #define [ALM_DM_RIGHT](#) R(-85.0)
- #define [ALM_DM_V_SIZE](#) 51
- #define [ALM_DM_H_SIZE](#) 89
- #define [ALM_HM_RADIUS](#) 600
- #define [ALM_HM_SIZE](#) 40
- #define [AGM_TOP](#) 2000
- #define [AGM_BOTTOM](#) -2000
- #define [AGM_LEFT](#) -2000
- #define [AGM_RIGHT](#) 2000
- #define [AGM_V_SIZE](#) 100
- #define [AGM_H_SIZE](#) 100
- #define [HM_CELL_COUNT](#) 4*ALM_HM_SIZE*ALM_HM_SIZE
- #define [DM_CELL_COUNT](#) ALM_DM_V_SIZE*ALM_DM_H_SIZE
- #define [GM_CELL_COUNT](#) AGM_V_SIZE*AGM_H_SIZE
- #define [ALM_DM_NUMCLUSTERS](#) 4
- #define [ALM_HM_NUMCLUSTERS](#) 4
- #define [AGM_NUMCLUSTERS](#) 4
- #define [AM_KMEANS_ITERATIONS](#) 5
- #define [ALM_IR_SPLAT_STDDEV](#) 3/600
- #define [ALM_GPA_CONFIDENCE](#) 0.7
- #define [AIBO_NECK_HEIGHT](#) 48.0
- #define [AIBO_HEAD_LENGTH](#) 66.6
- #define [AIBO_TILT_PIVOT_HEIGHT](#) 185.0

- #define [AIBO_CAM_H_FOV](#) R(57.6)
- #define [AIBO_CAM_V_FOV](#) R(47.8)
- #define [AIBO_TURN_PIVOT_DIST](#) 90.0
- #define [AIBO_MIN_CLEARANCE](#) 300
- #define [AIBO_MAX_BUMP](#) 10
- #define [AIBO_IR_CAL_MULTIPLIER](#) 0.78807
- #define [AIBO_IR_CAL_OFFSET](#) -15.19878

8.39.1 Define Documentation

8.39.1.1 #define AGM_BOTTOM -2000

Definition at line 96 of file Maps/Configuration.h.

8.39.1.2 #define AGM_H_SIZE 100

Definition at line 105 of file Maps/Configuration.h.

8.39.1.3 #define AGM_LEFT -2000

Definition at line 98 of file Maps/Configuration.h.

8.39.1.4 #define AGM_NUMCLUSTERS 4

Definition at line 122 of file Maps/Configuration.h.

8.39.1.5 #define AGM_RIGHT 2000

Definition at line 100 of file Maps/Configuration.h.

8.39.1.6 #define AGM_TOP 2000

Definition at line 94 of file Maps/Configuration.h.

8.39.1.7 #define AGM_V_SIZE 100

Definition at line 103 of file Maps/Configuration.h.

8.39.1.8 #define AIBO_CAM_H_FOV R(57.6)

Definition at line 147 of file Maps/Configuration.h.

8.39.1.9 #define AIBO_CAM_V_FOV R(47.8)

Definition at line 149 of file Maps/Configuration.h.

8.39.1.10 #define AIBO_HEAD_LENGTH 66.6

Definition at line 142 of file Maps/Configuration.h.

8.39.1.11 #define AIBO_IR_CAL_MULTIPLIER 0.78807

Definition at line 170 of file Maps/Configuration.h.

8.39.1.12 #define AIBO_IR_CAL_OFFSET -15.19878

Definition at line 171 of file Maps/Configuration.h.

8.39.1.13 #define AIBO_MAX_BUMP 10

Definition at line 158 of file Maps/Configuration.h.

8.39.1.14 #define AIBO_MIN_CLEARANCE 300

Definition at line 155 of file Maps/Configuration.h.

8.39.1.15 #define AIBO_NECK_HEIGHT 48.0

Definition at line 140 of file Maps/Configuration.h.

8.39.1.16 #define AIBO_TILT_PIVOT_HEIGHT 185.0

Definition at line 145 of file Maps/Configuration.h.

8.39.1.17 #define AIBO_TURN_PIVOT_DIST 90.0

Definition at line 152 of file Maps/Configuration.h.

8.39.1.18 #define ALM_DM_BOTTOM R(-75.0)

Definition at line 44 of file Maps/Configuration.h.

8.39.1.19 #define ALM_DM_H_SIZE 89

Definition at line 55 of file Maps/Configuration.h.

8.39.1.20 #define ALM_DM_LEFT R(85.0)

Definition at line 46 of file Maps/Configuration.h.

8.39.1.21 #define ALM_DM_NUMCLUSTERS 4

Definition at line 118 of file Maps/Configuration.h.

8.39.1.22 #define ALM_DM_RIGHT R(-85.0)

Definition at line 48 of file Maps/Configuration.h.

8.39.1.23 #define ALM_DM_TAX 0.8

Definition at line 19 of file Maps/Configuration.h.

8.39.1.24 #define ALM_DM_TOP R(25.0)

Definition at line 42 of file Maps/Configuration.h.

8.39.1.25 #define ALM_DM_V_SIZE 51

Definition at line 52 of file Maps/Configuration.h.

8.39.1.26 #define ALM_GM_TAX 0.9

Definition at line 23 of file Maps/Configuration.h.

8.39.1.27 #define ALM_GPA_CONFIDENCE 0.7

Definition at line 134 of file Maps/Configuration.h.

8.39.1.28 #define ALM_HM_NUMCLUSTERS 4

Definition at line 120 of file Maps/Configuration.h.

8.39.1.29 #define ALM_HM_RADIUS 600

Definition at line 68 of file Maps/Configuration.h.

8.39.1.30 #define ALM_HM_SIZE 40

Definition at line 72 of file Maps/Configuration.h.

8.39.1.31 #define ALM_HM_TAX 0.9

Definition at line 21 of file Maps/Configuration.h.

8.39.1.32 #define ALM_IR_SPLAT_STDDEV 3/600

Definition at line 129 of file Maps/Configuration.h.

8.39.1.33 #define AM_KMEANS_ITERATIONS 5

Definition at line 124 of file Maps/Configuration.h.

8.39.1.34 #define DM_CELL_COUNT ALM_DM_V_SIZE*ALM_DM_H_SIZE

Definition at line 112 of file Maps/Configuration.h.

8.39.1.35 #define GM_CELL_COUNT AGM_V_SIZE*AGM_H_SIZE

Definition at line 113 of file Maps/Configuration.h.

8.39.1.36 #define HM_CELL_COUNT 4*ALM_HM_SIZE*ALM_HM_SIZE

Definition at line 111 of file Maps/Configuration.h.

8.39.1.37 #define R(item) (((double)(item))*M.PI/180)

Definition at line 12 of file Maps/Configuration.h.

8.40 ControlBase.cc File Reference

8.40.1 Detailed Description

Implements [ControlBase](#) from which all items in the control system should inherit.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.6

State

Rel

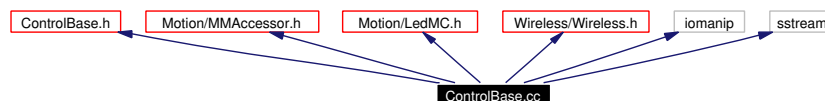
Date

2003/06/12 18:06:10

Definition in file [ControlBase.cc](#).

```
#include "ControlBase.h"
#include "Motion/MMAccessor.h"
#include "Motion/LedMC.h"
#include "Wireless/Wireless.h"
#include <iomanip>
#include <sstream>
```

Include dependency graph for ControlBase.cc:



8.41 ControlBase.h File Reference

8.41.1 Detailed Description

Defines [ControlBase](#) from which all items in the control system should inherit.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.15

State

Rel

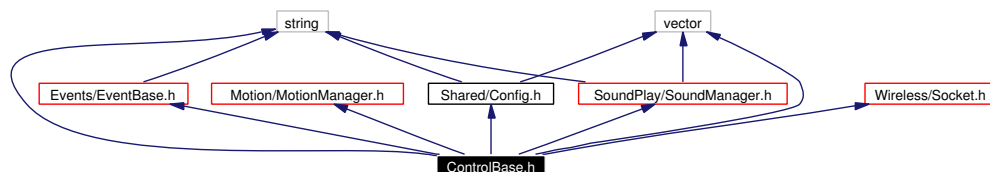
Date

2003/06/12 23:41:35

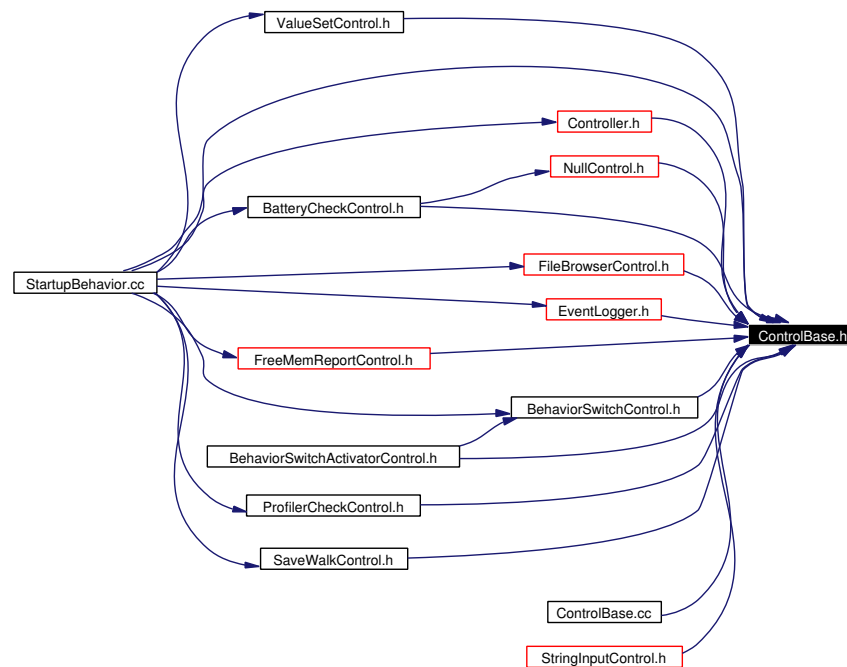
Definition in file [ControlBase.h](#).

```
#include "Events/EventBase.h"
#include "Motion/MotionManager.h"
#include "SoundPlay/SoundManager.h"
#include "Shared/Config.h"
#include "Wireless/Socket.h"
#include <string>
#include <vector>
```

Include dependency graph for ControlBase.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [ControlBase](#)
Base class for all items in the [Controller](#) hierarchy.

8.42 Controller.cc File Reference

8.42.1 Detailed Description

Implements [Controller](#) class, a behavior that should be started whenever the emergency stop goes on to provide menus for robot control.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.20

State

Rel

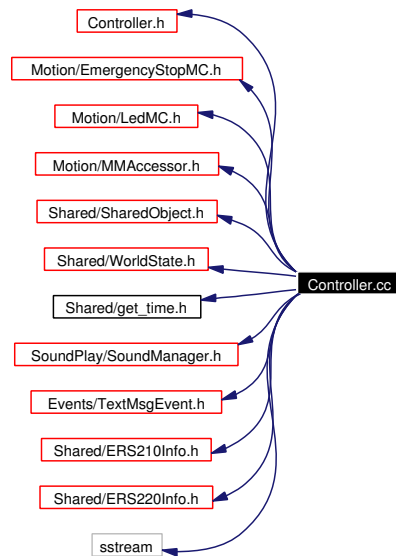
Date

2003/07/07 01:00:07

Definition in file [Controller.cc](#).

```
#include "Controller.h"
#include "Motion/EmergencyStopMC.h"
#include "Motion/LedMC.h"
#include "Motion/MMAccessor.h"
#include "Shared/SharedObject.h"
#include "Shared/WorldState.h"
#include "Shared/get_time.h"
#include "SoundPlay/SoundManager.h"
#include "Events/TextMsgEvent.h"
#include "Shared/ERS210Info.h"
#include "Shared/ERS220Info.h"
#include <sstream>
```

Include dependency graph for Controller.cc:



8.43 Controller.h File Reference

8.43.1 Detailed Description

Describes [Controller](#) class, a behavior that should be started whenever the emergency stop goes on to provide menus for robot control.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.16

State

Rel

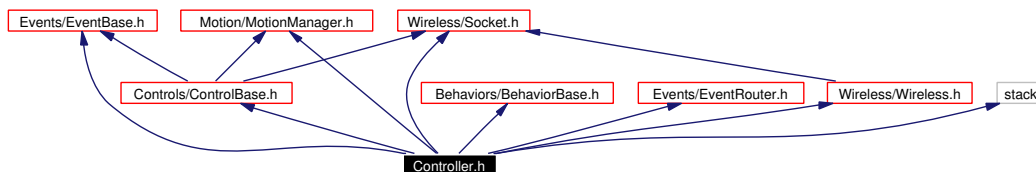
Date

2003/07/08 18:01:21

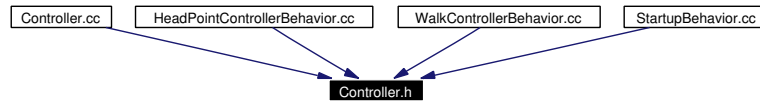
Definition in file [Controller.h](#).

```
#include "Controls/ControlBase.h"
#include "Behaviors/BehaviorBase.h"
#include "Events/EventBase.h"
#include "Events/EventRouter.h"
#include "Motion/MotionManager.h"
#include "Wireless/Wireless.h"
#include "Wireless/Socket.h"
#include <stack>
```

Include dependency graph for Controller.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [Controller](#)

Handles the menu/command system... when it detects the [EmergencyStopMC](#) is activated, it'll kick into high priority.

8.44 debuget.h File Reference

8.44.1 Detailed Description

Defines several debugging functions and macros, including [ASSERT](#) (and variations).

Author:

ejt (Creator)

Author

alokl

Name

tekkotsu-1.4.1

Revision

1.1.1.1

State

Rel

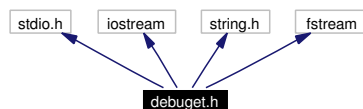
Date

2002/09/30 18:19:48

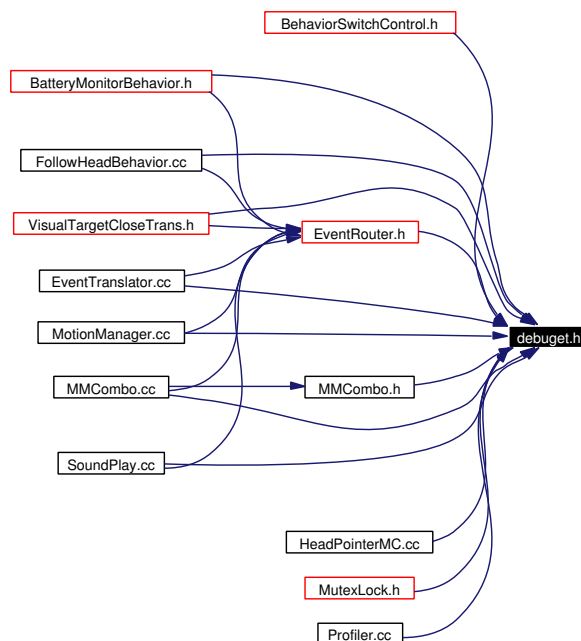
Definition in file [debuget.h](#).

```
#include <stdio.h>
#include <iostream>
#include <string.h>
#include <fstream>
```

Include dependency graph for debuget.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define **ASSERT**(b, str) {if(!(b)) std::cout << "ASSERT:"<<_extract-
Filename(__FILE__)<<'.'<<__LINE__<<'.'<< str << std::endl;}
if the bool b is false, std::cout the string
 - #define **ASSERTRET**(b, str) {if(!(b)) { std::cout << "ASSERT:"<<_extract-
Filename(__FILE__)<<'.'<<__LINE__<<'.'<< str << std::endl; return; }}
 - #define **ASSERTRETVAL**(b, str, v) {if(!(b)) { std::cout << "ASSERT:"<<_extract-
Filename(__FILE__)<<'.'<<__LINE__<<'.'<< str << std::endl; return
v; }}
 - #define **ASSERTFATAL**(b, str, x) {if(!(b)) { std::cout << "ASSERT:"<<_extract-
Filename(__FILE__)<<'.'<<__LINE__<<'.'<< str << std::endl;
exit(x); }}
- if the bool b is false, std::cout the string and exit(x)*

Functions

- const char * [_extractFilename](#) (const char *path)
for historical reasons - the previous compiler give the entire path for `__FILE__`, for display, just use the filename
- char [hexdigit](#) (int c)
returns the hex char that corresponds to c, which should be 0-16 (returns '.' otherwise)
- void [charhexout](#) (char c)
printf's the two hex digits corresponding to a byte
- void [hexout](#) (const void *p, size_t n)
charhexout's n bytes starting at p

8.44.2 Define Documentation

8.44.2.1 `#define ASSERT(b, str) {if(!(b)) std::cout << "ASSERT:"<<_
_extractFilename(__FILE__)<<'.'<<__LINE__<<'.'<< str <<
std::endl;}`

if the bool b is false, std::cout the string

Definition at line 19 of file debug.h.

8.44.2.2 `#define ASSERTFATAL(b, str, x) {if(!(b)) { std::cout
<< "ASSERT:"<<_extractFilename(__FILE__-
)<<'.'<<__LINE__<<'.'<< str << std::endl; exit(x);
}}`

if the bool b is false, std::cout the string and exit(x)

Definition at line 25 of file debug.h.

8.44.2.3 `#define ASSERTRET(b, str) {if(!(b)) { std::cout << "ASSERT:"<<_
_extractFilename(__FILE__)<<'.'<<__LINE__<<'.'<< str <<
std::endl; return; }}`

if the bool b is false, std::cout the string and return

Definition at line 21 of file debug.h.

```

8.44.2.4 #define ASSERTRETURN(b, str, v) {if(!(b)) { std::cout
    << "ASSERT:"<<_extractFilename(__FILE__-
    )<<'.'<<__LINE__<<':'<< str << std::endl; return v;
    }}

```

if the bool `b` is false, `std::cout` the string and return the value

Definition at line 23 of file `debuget.h`.

8.44.3 Function Documentation

8.44.3.1 `const char* _extractFilename (const char * path)` [inline]

for historical reasons - the previous compiler give the entire path for `__FILE__`, for display, just use the filename

Definition at line 11 of file `debuget.h`.

8.44.3.2 `void charhexout (char c)` [inline]

`printf`'s the two hex digits corresponding to a byte

Definition at line 50 of file `debuget.h`.

References `hexdigit()`.

8.44.3.3 `char hexdigit (int c)` [inline]

returns the hex char that corresponds to `c`, which should be 0-16 (returns '.' otherwise)

Definition at line 39 of file `debuget.h`.

8.44.3.4 `void hexout (const void * p, size_t n)` [inline]

`charhexout`'s `n` bytes starting at `p`

Definition at line 55 of file `debuget.h`.

References `charhexout()`.

8.45 DriveMeBehavior.cc File Reference

8.45.1 Detailed Description

Implements [DriveMeBehavior](#), a very simple behavior that asks the user for [WalkMC](#) walking parameters and a walk duration.

Author:

tss (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

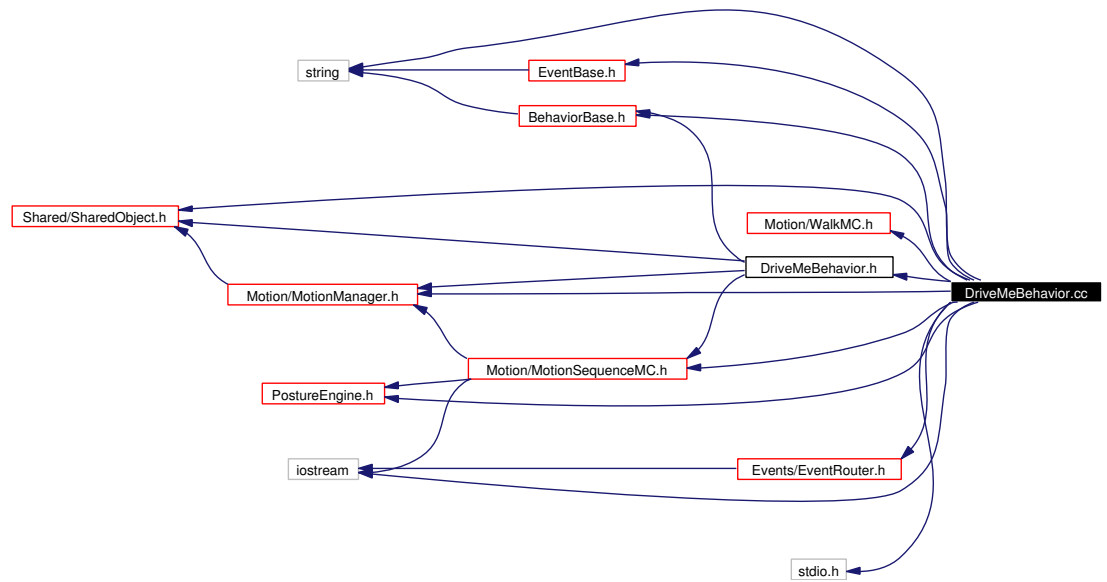
Date

2003/06/12 23:41:39

Definition in file [DriveMeBehavior.cc](#).

```
#include "BehaviorBase.h"
#include "Motion/MotionManager.h"
#include "Motion/WalkMC.h"
#include "Motion/MotionSequenceMC.h"
#include "Motion/PostureEngine.h"
#include "Shared/SharedObject.h"
#include "Events/EventRouter.h"
#include "Events/EventBase.h"
#include "DriveMeBehavior.h"
#include <iostream>
#include <string>
#include <stdio.h>
```

Include dependency graph for DriveMeBehavior.cc:



8.46 DriveMeBehavior.h File Reference

8.46.1 Detailed Description

Describes [DriveMeBehavior](#), a very simple behavior that asks the user for [WalkMC](#) walking parameters and a walk duration.

Author:

tss (Creator)

Author

ejt

Name

tekkotsu-1.4_1

Revision

1.4

State

Rel

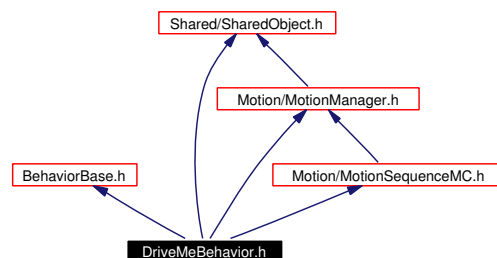
Date

2003/06/12 23:41:39

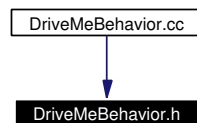
Definition in file [DriveMeBehavior.h](#).

```
#include "BehaviorBase.h"
#include "Motion/MotionManager.h"
#include "Motion/MotionSequenceMC.h"
#include "Shared/SharedObject.h"
```

Include dependency graph for DriveMeBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [DriveMeBehavior](#)

A very simple behavior that asks the user for [WalkMC](#) walking parameters and a walk duration.

8.47 DumbWM2Behavior.h File Reference

8.47.1 Detailed Description

Describes [DumbWM2Behavior](#) - Simply turns on a WM2 object. Useful for running concurrently with other behaviors and seeing what shows up in the world model.

Author:

tss (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.7

State

Rel

Date

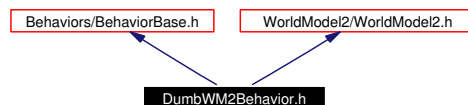
2003/06/12 23:41:39

Definition in file [DumbWM2Behavior.h](#).

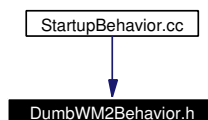
```
#include "Behaviors/BehaviorBase.h"
```

```
#include "WorldModel2/WorldModel2.h"
```

Include dependency graph for DumbWM2Behavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [DumbWM2Behavior](#)

Simply turns on a WM2 object. Useful for running concurrently with other behaviors and seeing what shows up in the world model.

8.48 DumpFileControl.h File Reference

8.48.1 Detailed Description

Defines [DumpFileControl](#), which when activated, plays a sound selected from the memory stick.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.1

State

Rel

Date

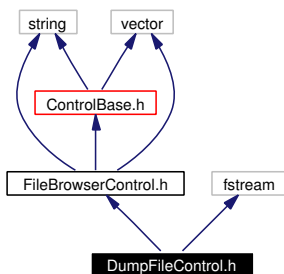
2003/06/11 01:22:47

Definition in file [DumpFileControl.h](#).

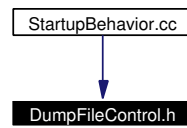
```
#include "FileBrowserControl.h"
```

```
#include <fstream>
```

Include dependency graph for DumpFileControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [DumpFileControl](#)

Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...).

8.49 DynamicMotionSequence.h File Reference

8.49.1 Detailed Description

Uses STL's vector for dynamic memory allocation - don't use this as a motion command, pointers in shared memory regions can be invalid in other processes.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.3

State

Rel

Date

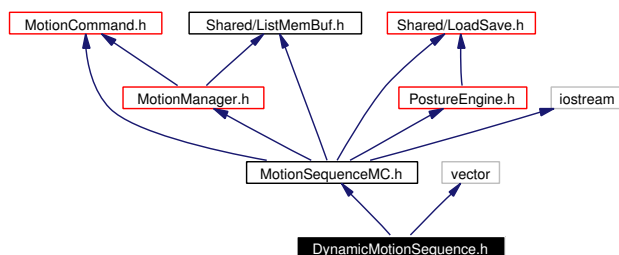
2003/04/25 18:22:39

Definition in file [DynamicMotionSequence.h](#).

```
#include "MotionSequenceMC.h"
```

```
#include <vector>
```

Include dependency graph for DynamicMotionSequence.h:



Compounds

- class [DynamicMotionSequence](#)

Uses STL's vector for dynamic memory allocation - don't use this as a motion command, pointers in shared memory regions can be invalid in other processes.

8.50 EmergencyStopMC.cc File Reference

8.50.1 Detailed Description

Implements [EmergencyStopMC](#), overrides all joints, allows modelling, blinks tail.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.13

State

Rel

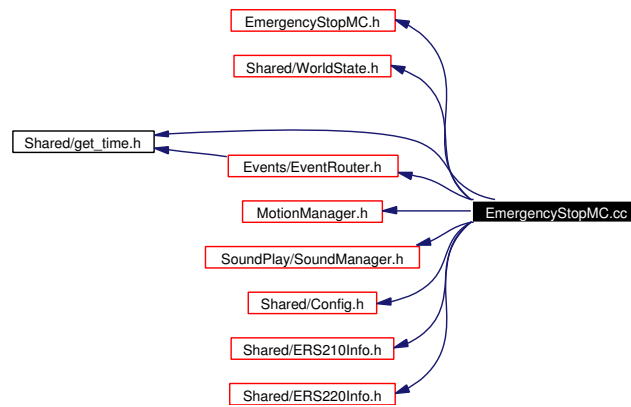
Date

2003/06/10 18:30:00

Definition in file [EmergencyStopMC.cc](#).

```
#include "EmergencyStopMC.h"
#include "Shared/WorldState.h"
#include "Shared/get_time.h"
#include "Motion/MotionManager.h"
#include "SoundPlay/SoundManager.h"
#include "Shared/Config.h"
#include "Events/EventRouter.h"
#include "Shared/ERS210Info.h"
#include "Shared/ERS220Info.h"
```

Include dependency graph for EmergencyStopMC.cc:



8.51 EmergencyStopMC.h File Reference

8.51.1 Detailed Description

Describes [EmergencyStopMC](#), overrides all joints, allows modelling, blinks tail.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.7

State

Rel

Date

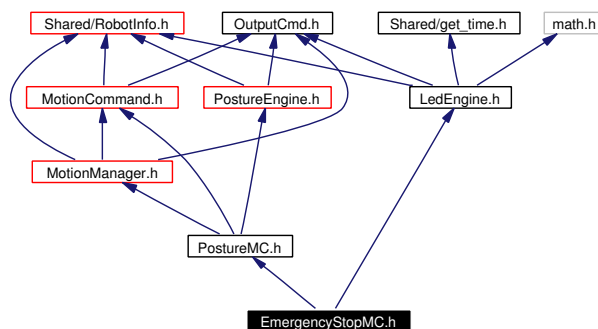
2003/05/13 21:40:10

Definition in file [EmergencyStopMC.h](#).

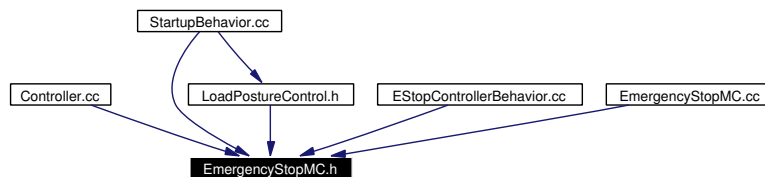
```
#include "PostureMC.h"
```

```
#include "LedEngine.h"
```

Include dependency graph for EmergencyStopMC.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [EmergencyStopMC](#)
overrides all joints with high priority freeze, blinks tail pink/red/blue cycle

8.52 entry.h File Reference

8.53 ERS210Info.h File Reference

8.53.1 Detailed Description

Defines [RobotInfo](#) namespace for ERS-210 models, gives some information about the robot's capabilities, such as joint counts, offsets, names and PID values.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.6

State

Rel

Date

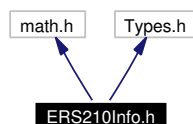
2003/06/12 23:41:40

Definition in file [ERS210Info.h](#).

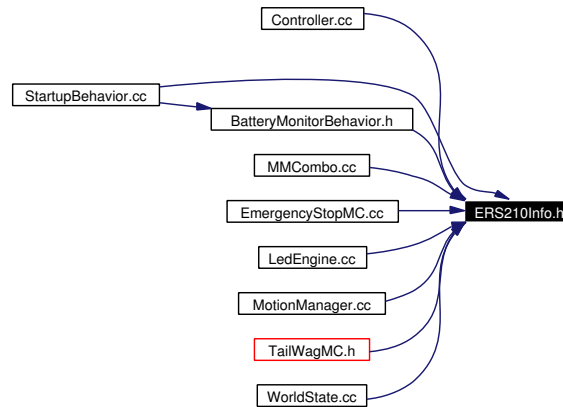
```
#include <math.h>
```

```
#include <Types.h>
```

Include dependency graph for ERS210Info.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [ERS210Info](#)

Defines

- `#define RAD(deg) (((deg) * M_PI) / 180.0)`
Just a little macro for converting degrees to radians.
- `#define __RI_RAD_FLAG`
a flag so we undef these after we're done - do you have a cleaner solution?

8.53.2 Define Documentation

8.53.2.1 `#define __RI_RAD_FLAG`

a flag so we undef these after we're done - do you have a cleaner solution?

Definition at line 374 of file ERS210Info.h.

8.53.2.2 `#define RAD(deg) (((deg) * M_PI) / 180.0)`

Just a little macro for converting degrees to radians.

Definition at line 372 of file ERS210Info.h.

8.54 ERS220Info.h File Reference

8.54.1 Detailed Description

Defines [RobotInfo](#) namespace for ERS-220 models, gives some information about the robot's capabilities, such as joint counts, offsets, names and PID values.

Author:

Daishi MORI (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.6

State

Rel

Date

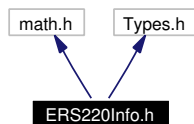
2003/06/12 04:16:44

Definition in file [ERS220Info.h](#).

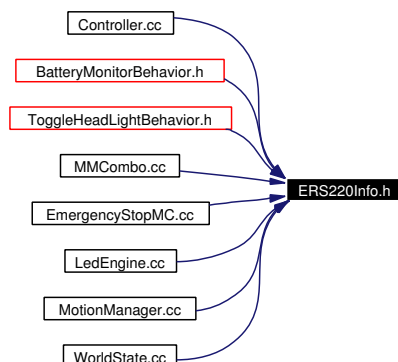
```
#include <math.h>
```

```
#include <Types.h>
```

Include dependency graph for ERS220Info.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [ERS220Info](#)

Defines

- #define [RAD\(deg\)](#) (((deg) * M_PI) / 180.0)
Just a little macro for converting degrees to radians.
- #define [_RI_RAD_FLAG](#)
a flag so we undef these after we're done - do you have a cleaner solution?

8.54.2 Define Documentation

8.54.2.1 #define _RI_RAD_FLAG

a flag so we undef these after we're done - do you have a cleaner solution?

Definition at line 476 of file ERS220Info.h.

8.54.2.2 #define RAD(deg) (((deg) * M_PI) / 180.0)

Just a little macro for converting degrees to radians.

Definition at line 474 of file ERS220Info.h.

8.55 ERS2xxInfo.h File Reference

8.55.1 Detailed Description

Defines [RobotInfo](#) namespace for ERS-2xx series of robots, such as joint counts, offsets, names and PID values.

Author:

Daishi MORI (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

Date

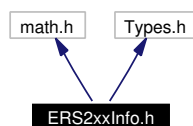
2003/06/12 23:41:40

Definition in file [ERS2xxInfo.h](#).

```
#include <math.h>
```

```
#include <Types.h>
```

Include dependency graph for ERS2xxInfo.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [ERS2xxInfo](#)

Defines

- `#define RAD(deg) (((deg) * M_PI) / 180.0)`
Just a little macro for converting degrees to radians.
- `#define _RLRAD_FLAG`
a flag so we undef these after we're done - do you have a cleaner solution?

8.55.2 Define Documentation

8.55.2.1 `#define _RLRAD_FLAG`

a flag so we undef these after we're done - do you have a cleaner solution?

Definition at line 541 of file ERS2xxInfo.h.

8.55.2.2 `#define RAD(deg) (((deg) * M_PI) / 180.0)`

Just a little macro for converting degrees to radians.

Definition at line 539 of file ERS2xxInfo.h.

8.56 EStopControllerBehavior.cc File Reference

8.56.1 Detailed Description

Implements [EStopControllerBehavior](#), listens to commands coming in from the command port for remotely controlling toggling the estop.

Author:

tss (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

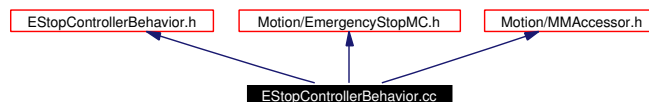
Date

2003/07/09 00:10:57

Definition in file [EStopControllerBehavior.cc](#).

```
#include "EStopControllerBehavior.h"  
#include "Motion/EmergencyStopMC.h"  
#include "Motion/MMAccessor.h"
```

Include dependency graph for EStopControllerBehavior.cc:



8.57 EStopControllerBehavior.h File Reference

8.57.1 Detailed Description

Describes [EStopControllerBehavior](#), listens to control commands coming in from the command port for remotely toggling the estop.

Author:

tss (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.1

State

Rel

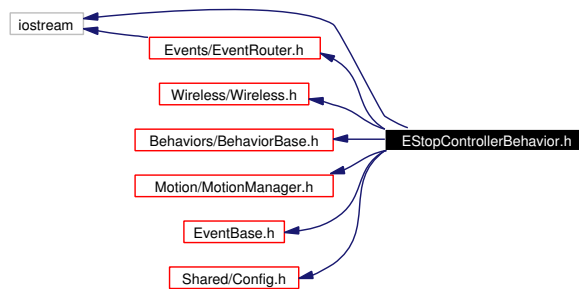
Date

2003/07/07 07:08:27

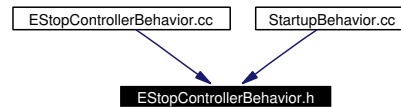
Definition in file [EStopControllerBehavior.h](#).

```
#include <iostream>
#include "Wireless/Wireless.h"
#include "Behaviors/BehaviorBase.h"
#include "Motion/MotionManager.h"
#include "Events/EventRouter.h"
#include "Events/EventBase.h"
#include "Shared/Config.h"
```

Include dependency graph for EStopControllerBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [EStopControllerBehavior](#)

Listens to control commands coming in from the command port for remotely controlling the head.

8.58 EventBase.cc File Reference

8.58.1 Detailed Description

Implements [EventBase](#), the basic class for sending events around the system.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.10

State

Rel

Date

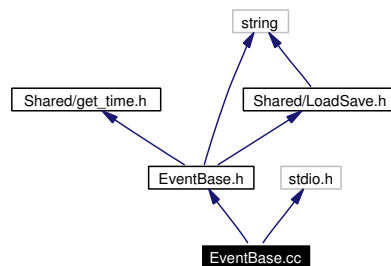
2003/04/25 18:22:39

Definition in file [EventBase.cc](#).

```
#include "EventBase.h"
```

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

Include dependency graph for EventBase.cc:



8.59 EventBase.h File Reference

8.59.1 Detailed Description

Describes [EventBase](#), the basic class for sending events around the system.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.12

State

Rel

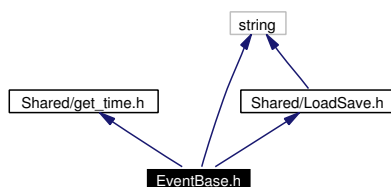
Date

2003/04/09 04:48:15

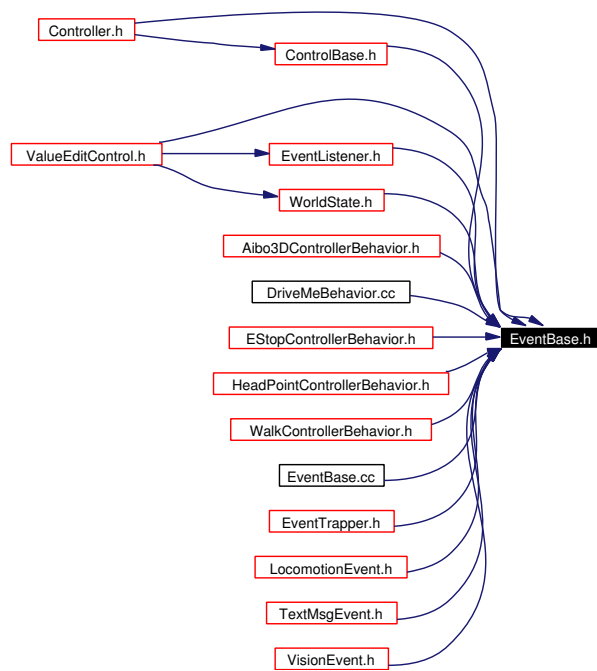
Definition in file [EventBase.h](#).

```
#include "Shared/get_time.h"
#include "Shared/LoadSave.h"
#include <string>
```

Include dependency graph for EventBase.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [EventBase](#)

The basis of events passed around the high level system.

8.60 EventListener.h File Reference

8.60.1 Detailed Description

Defines [EventListener](#) class, an interface for anything that wants to receive events.

Author:

ejt (Creator)

Author

alokl

Name

tekkotsu-1.4.1

Revision

1.1.1.1

State

Rel

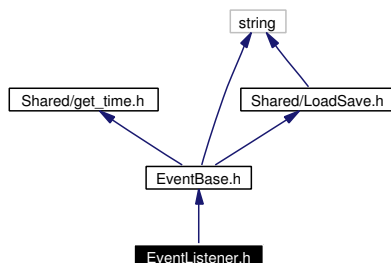
Date

2002/09/30 18:19:47

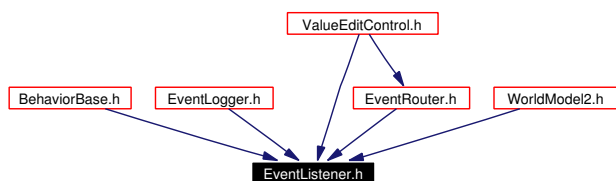
Definition in file [EventListener.h](#).

```
#include "EventBase.h"
```

Include dependency graph for EventListener.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [EventListener](#)

An interface to allow a standard method of passing events.

8.61 EventLogger.cc File Reference

8.61.1 Detailed Description

Describes [EventLogger](#), which allows logging of events to the console or a file.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.7

State

Rel

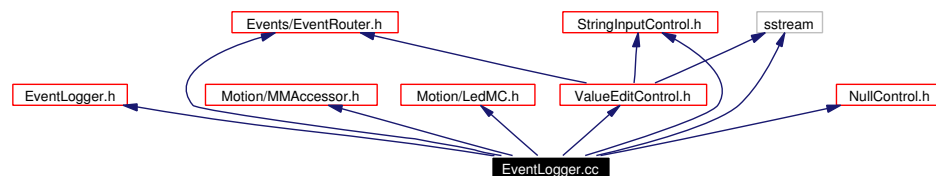
Date

2003/06/10 00:53:48

Definition in file [EventLogger.cc](#).

```
#include "EventLogger.h"
#include "Events/EventRouter.h"
#include "Motion/MMAccessor.h"
#include "Motion/LedMC.h"
#include "ValueEditControl.h"
#include "StringInputControl.h"
#include "NullControl.h"
#include <sstream>
```

Include dependency graph for EventLogger.cc:



8.62 EventLogger.h File Reference

8.62.1 Detailed Description

Describes [EventLogger](#), which allows logging of events to the console or a file.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4_1

Revision

1.4

State

Rel

Date

2003/06/09 08:05:12

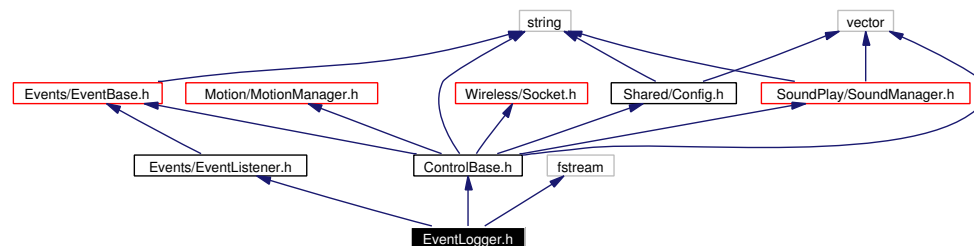
Definition in file [EventLogger.h](#).

```
#include "ControlBase.h"
```

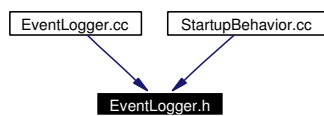
```
#include "Events/EventListener.h"
```

```
#include <fstream>
```

Include dependency graph for EventLogger.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class `EventLogger`
allows logging of events to the console or a file

8.63 EventRouter.cc File Reference

8.63.1 Detailed Description

Implements [EventRouter](#) class, for distribution and trapping of events to listeners.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.8

State

Rel

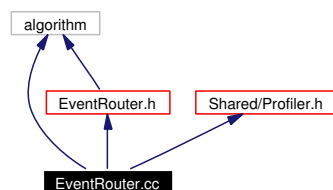
Date

2003/06/09 08:05:13

Definition in file [EventRouter.cc](#).

```
#include "EventRouter.h"  
#include "Shared/Profiler.h"  
#include <algorithm>
```

Include dependency graph for EventRouter.cc:



Variables

- [EventRouter](#) * [erouter](#) = NULL

a global router for cross communication, probably the most common usage, although perhaps there may be times you'd rather have multiple event routers for smaller sections

8.63.2 Variable Documentation

8.63.2.1 `EventRouter* erouter = NULL`

a global router for cross communication, probably the most common usage, although perhaps there may be times you'd rather have multiple event routers for smaller sections

Definition at line 5 of file EventRouter.cc.

8.64 EventRouter.h File Reference

8.64.1 Detailed Description

Describes [EventRouter](#) class, for distribution and trapping of events to listeners.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.10

State

Rel

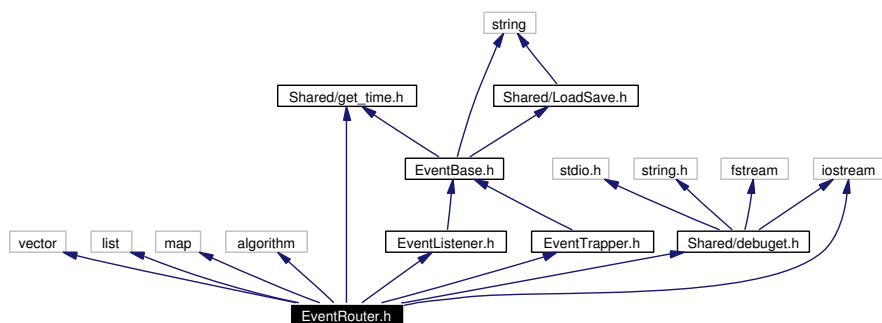
Date

2003/04/25 18:22:39

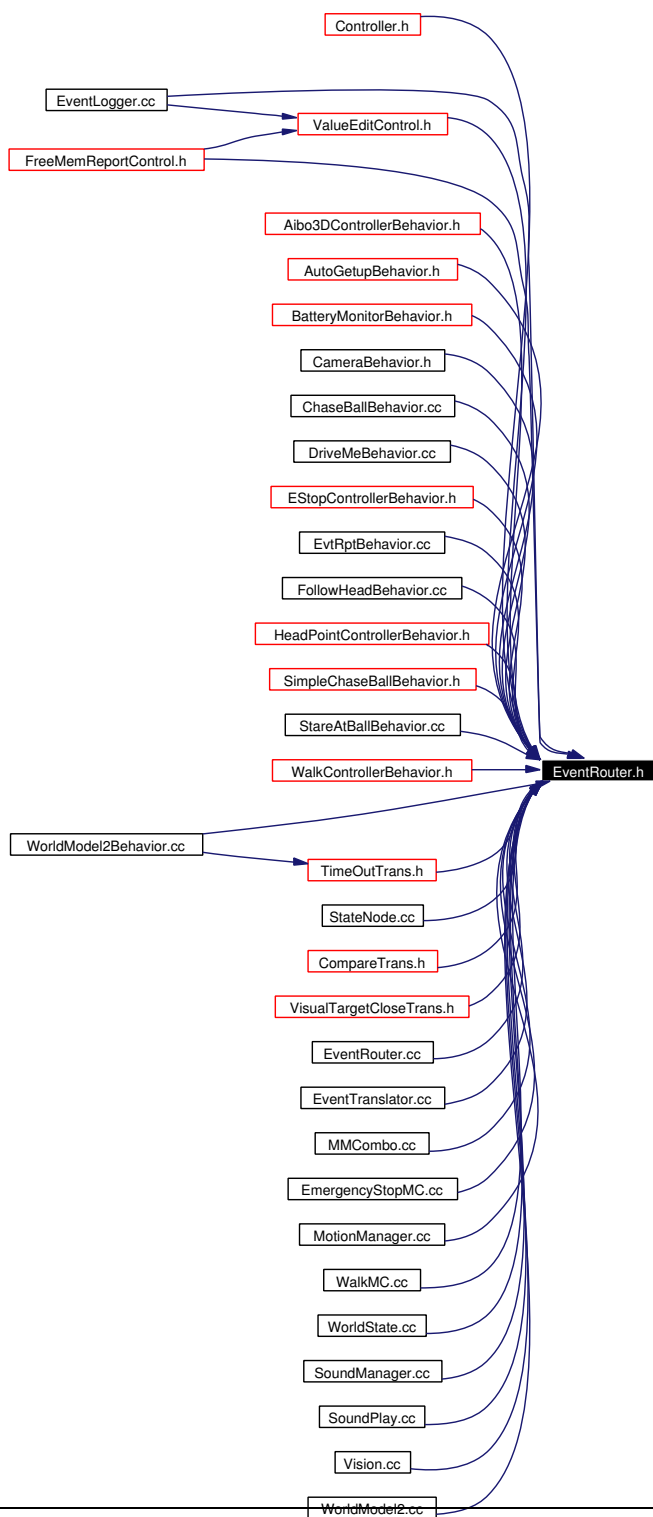
Definition in file [EventRouter.h](#).

```
#include <vector>
#include <list>
#include <map>
#include <algorithm>
#include "EventListener.h"
#include "EventTrapper.h"
#include "Shared/get_time.h"
#include "Shared/debuget.h"
#include <iostream>
```

Include dependency graph for EventRouter.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [EventRouter](#)
This class will handle distribution of events as well as management of timers.
- class [EventRouter.EventMapper](#)
Does the actual storage of the mapping between EventBase's and the Event-Listeners/EventTrappers who should receive them.
- struct [EventRouter.TimerEntry](#)
Contains all the information needed to maintain a timer by the [EventRouter](#).
- class [EventRouter.TimerEntryPtrCmp](#)
Used by STL to sort the timer list in order of activation time.

Variables

- [EventRouter](#) * [erouter](#)
a global router for cross communication, probably the most common usage, although perhaps there may be times you'd rather have multiple event routers for smaller sections

8.64.2 Variable Documentation

8.64.2.1 [EventRouter](#)* [erouter](#)

a global router for cross communication, probably the most common usage, although perhaps there may be times you'd rather have multiple event routers for smaller sections

Definition at line 240 of file EventRouter.h.

8.65 EventTranslator.cc File Reference

8.65.1 Detailed Description

Implements [EventTranslator](#), which receives events from EventRouters in non-Main processes and adds them into a [SharedQueue](#) for Main to pick up.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.4

State

Rel

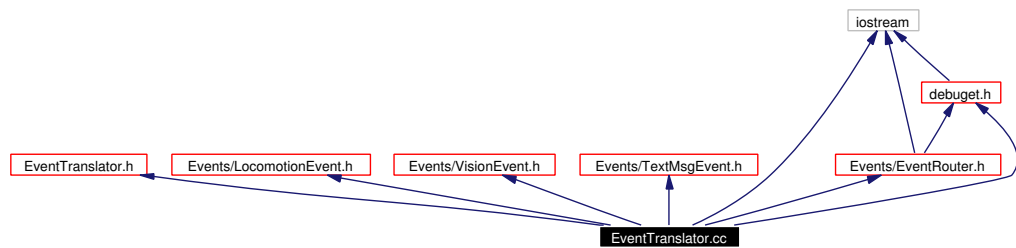
Date

2003/06/12 23:41:40

Definition in file [EventTranslator.cc](#).

```
#include "EventTranslator.h"
#include "Events/LocomotionEvent.h"
#include "Events/VisionEvent.h"
#include "Events/TextMsgEvent.h"
#include "Events/EventRouter.h"
#include "Shared/debuget.h"
#include <iostream>
```

Include dependency graph for EventTranslator.cc:



8.66 EventTranslator.h File Reference

8.66.1 Detailed Description

Describes [EventTranslator](#), which receives events from EventRouters in non-Main processes and adds them into a [SharedQueue](#) for Main to pick up.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.2

State

Rel

Date

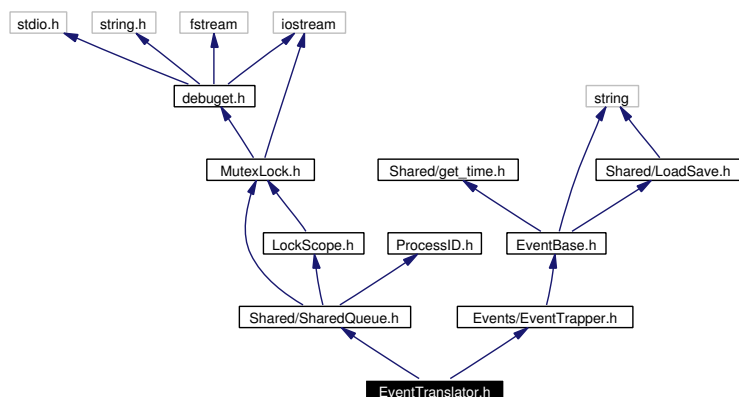
2003/06/12 23:41:40

Definition in file [EventTranslator.h](#).

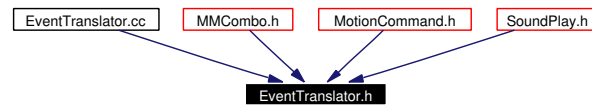
```
#include "Shared/SharedQueue.h"
```

```
#include "Events/EventTrapper.h"
```

Include dependency graph for EventTranslator.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [EventTranslator](#)

EventTranslator receives events from *EventRouters* in non-Main processes and adds them into a [SharedQueue](#) for Main to pick up.

8.67 EventTrapper.h File Reference

8.67.1 Detailed Description

Defines [EventTrapper](#) class, an interface for anything that wants to trap events.

Author:

ejt (Creator)

Author

alokl

Name

tekkotsu-1.4.1

Revision

1.1.1.1

State

Rel

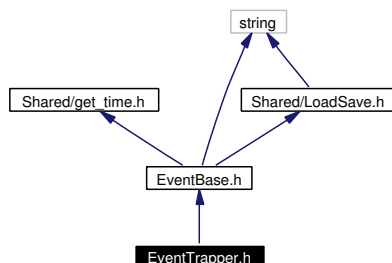
Date

2002/09/30 18:19:47

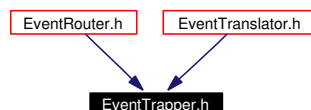
Definition in file [EventTrapper.h](#).

```
#include "EventBase.h"
```

Include dependency graph for EventTrapper.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [EventTrapper](#)

An interface to allow a standard method of trapping events.

8.68 EvtRptBehavior.cc File Reference

8.68.1 Detailed Description

Implements [EvtRptBehavior](#), which counts information about events it sees.

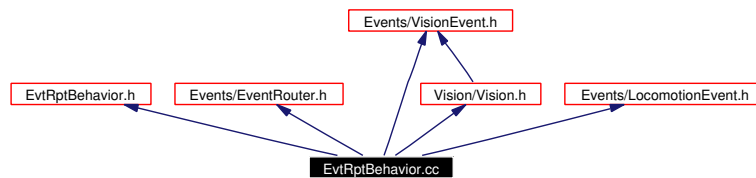
Author:

tss (Creator)

Definition in file [EvtRptBehavior.cc](#).

```
#include "EvtRptBehavior.h"
#include "Events/EventRouter.h"
#include "Events/VisionEvent.h"
#include "Events/LocomotionEvent.h"
#include "Vision/Vision.h"
```

Include dependency graph for EvtRptBehavior.cc:



8.69 EvtRptBehavior.h File Reference

8.69.1 Detailed Description

Describes [EvtRptBehavior](#), which counts information about events it sees.

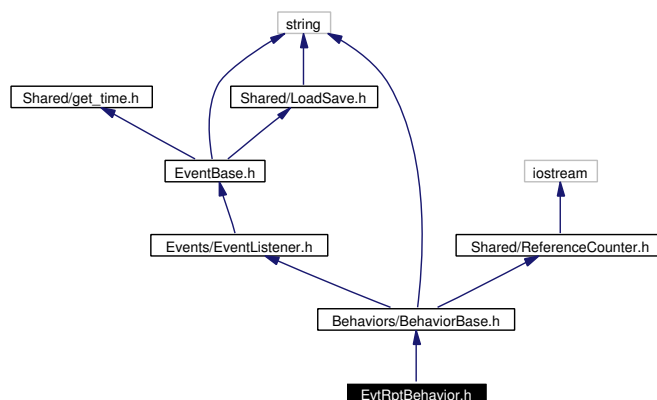
Author:

tss (Creator)

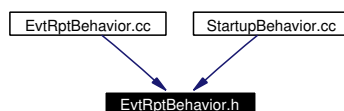
Definition in file [EvtRptBehavior.h](#).

```
#include "Behaviors/BehaviorBase.h"
```

Include dependency graph for EvtRptBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [EvtRptBehavior](#)

A simple behavior to test event reports.

8.70 Factory.h File Reference

8.70.1 Detailed Description

Defines [Factory](#), a lightweight class to override for constructing new objects.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

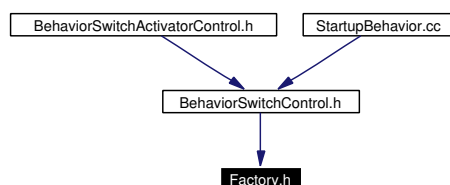
Rel

Date

2003/01/09 02:02:59

Definition in file [Factory.h](#).

This graph shows which files directly or indirectly include this file:



Compounds

- class [Factory](#)
A lightweight class to override for constructing new objects (if you need to pass constructors parameters, etc.).
- class [Factory1Arg](#)
Uses template to specify a constant parameter to the constructor.

8.71 FileBrowserControl.cc File Reference

8.71.1 Detailed Description

Implements [FileBrowserControl](#), which displays the contents of a directory.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.7

State

Rel

Date

2003/06/12 23:41:36

Definition in file [FileBrowserControl.cc](#).

```
#include "FileBrowserControl.h"
```

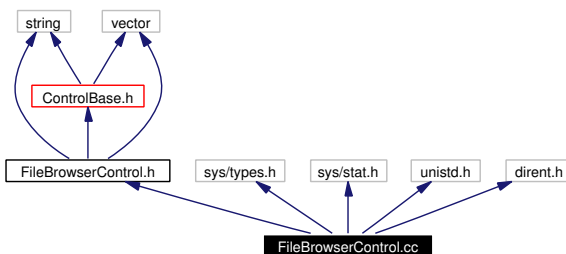
```
#include <sys/types.h>
```

```
#include <sys/stat.h>
```

```
#include <unistd.h>
```

```
#include <dirent.h>
```

Include dependency graph for FileBrowserControl.cc:



8.72 FileBrowserControl.h File Reference

8.72.1 Detailed Description

Describes [FileBrowserControl](#), which displays the contents of a directory.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.7

State

Rel

Date

2003/06/10 00:53:48

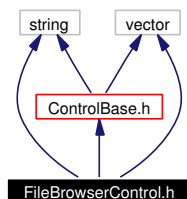
Definition in file [FileBrowserControl.h](#).

```
#include "ControlBase.h"
```

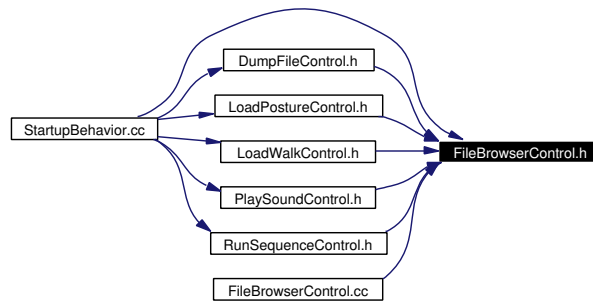
```
#include <string>
```

```
#include <vector>
```

Include dependency graph for FileBrowserControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [FileBrowserControl](#)

Displays the contents of a directory in a control menu, probably useful as a baseclass for other controls.

8.73 FollowHeadBehavior.cc File Reference

8.73.1 Detailed Description

Implements [FollowHeadBehavior](#), walks where the head is pointing.

Author:

ejt (Creator)

Author

ejt

\$Name \$

Revision

1.3

State

Rel

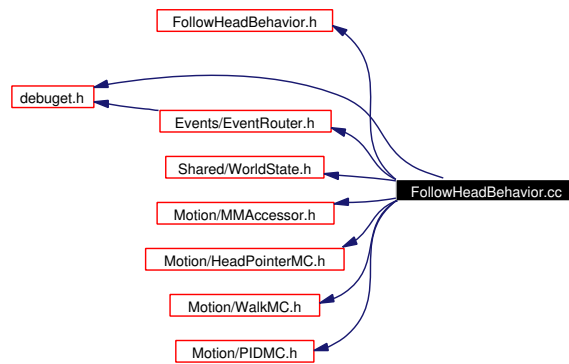
Date

2003/03/09 02:45:21

Definition in file [FollowHeadBehavior.cc](#).

```
#include "FollowHeadBehavior.h"
#include "Events/EventRouter.h"
#include "Shared/debuget.h"
#include "Shared/WorldState.h"
#include "Motion/MMAccessor.h"
#include "Motion/HeadPointerMC.h"
#include "Motion/WalkMC.h"
#include "Motion/PIDMC.h"
```

Include dependency graph for FollowHeadBehavior.cc:



8.74 FollowHeadBehavior.h File Reference

8.74.1 Detailed Description

Describes [FollowHeadBehavior](#), walks where the head is pointing.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

Date

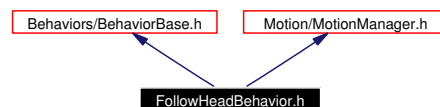
2003/06/05 17:03:15

Definition in file [FollowHeadBehavior.h](#).

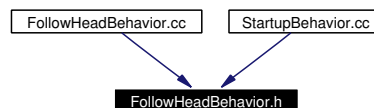
```
#include "Behaviors/BehaviorBase.h"
```

```
#include "Motion/MotionManager.h"
```

Include dependency graph for FollowHeadBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [FollowHeadBehavior](#)

Will walk where the head is pointing.

8.75 FreeMemReportControl.cc File Reference

8.75.1 Detailed Description

Implements [FreeMemReportControl](#), which gives reports on free memory size at various (configurable) rates.

Author:

ejt (object), alokl (core function)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

Date

2003/06/12 23:41:36

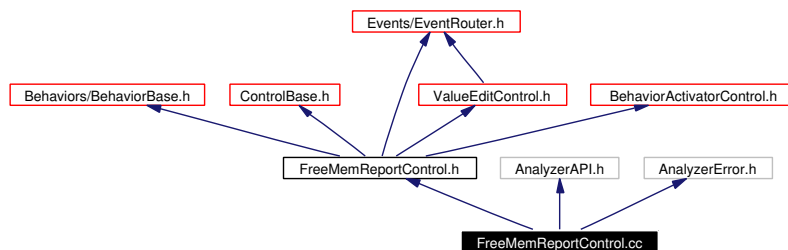
Definition in file [FreeMemReportControl.cc](#).

```
#include "FreeMemReportControl.h"
```

```
#include <AnalyzerAPI.h>
```

```
#include <AnalyzerError.h>
```

Include dependency graph for FreeMemReportControl.cc:



8.76 FreeMemReportControl.h File Reference

8.76.1 Detailed Description

Describes [FreeMemReportControl](#), which gives reports on free memory size at various (configurable) rates.

Author:

ejt (object), alokl (core function)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

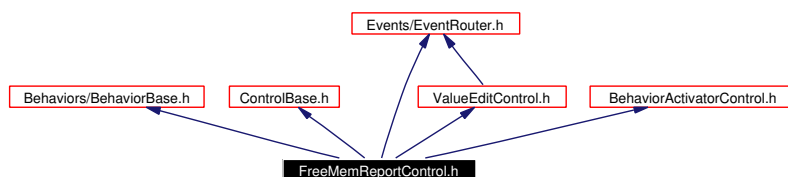
Date

2003/06/12 23:41:36

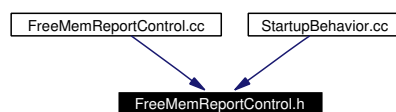
Definition in file [FreeMemReportControl.h](#).

```
#include "Behaviors/BehaviorBase.h"
#include "ControlBase.h"
#include "Events/EventRouter.h"
#include "BehaviorActivatorControl.h"
#include "ValueEditControl.h"
```

Include dependency graph for FreeMemReportControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [FreeMemReportControl](#)
gives reports on free memory size at variable rates, can also warn if free memory drops too low.

8.77 Geometry.h File Reference

8.77.1 Detailed Description

typedefs commonly used GVector's to vector3d, vector2d, vector3f, and vector2f

Author:

CMU RoboSoccer 2001-2002 (Creator)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
-----

This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
-----

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed. In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

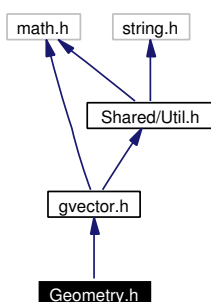
Date

2003/01/17 23:15:51

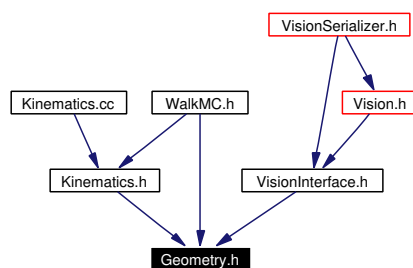
Definition in file [Geometry.h](#).

```
#include "gvector.h"
```

Include dependency graph for Geometry.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef [GVector::vector3d](#)< double > [vector3d](#)
- typedef [vector3d](#) [point3d](#)
- typedef [GVector::vector2d](#)< double > [vector2d](#)
- typedef [vector2d](#) [point2d](#)
- typedef [GVector::vector3d](#)< float > [vector3f](#)
- typedef [vector3f](#) [point3f](#)
- typedef [GVector::vector2d](#)< float > [vector2f](#)
- typedef [vector2f](#) [point2f](#)

8.77.2 Typedef Documentation

8.77.2.1 typedef [vector2d](#) [point2d](#)

Definition at line 39 of file Geometry.h.

8.77.2.2 typedef [vector2f](#) [point2f](#)

Definition at line 45 of file Geometry.h.

8.77.2.3 typedef [vector3d](#) [point3d](#)

Definition at line 36 of file Geometry.h.

8.77.2.4 typedef [vector3f](#) [point3f](#)

Definition at line 42 of file Geometry.h.

8.77.2.5 typedef [GVector::vector2d](#)<double> [vector2d](#)

Definition at line 38 of file Geometry.h.

8.77.2.6 typedef [GVector::vector2d](#)<float> [vector2f](#)

Definition at line 44 of file Geometry.h.

8.77.2.7 typedef [GVector::vector3d](#)<double> [vector3d](#)

Definition at line 35 of file Geometry.h.

8.77.2.8 typedef [GVector::vector3d](#)<float> [vector3f](#)

Definition at line 41 of file Geometry.h.

8.78 get_time.cc File Reference

8.78.1 Detailed Description

Implementation of [get_time\(\)](#), a simple way to get the current time since boot in milliseconds.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

Date

2003/01/09 02:03:00

Definition in file [get_time.cc](#).

```
#include "get_time.h"
```

```
#include <MCOOP.h>
```

Include dependency graph for get_time.cc:



Functions

- unsigned int [get_time](#) ()
a simple way to get the current time since boot in milliseconds (see [TimeET](#) for better accuracy)

8.78.2 Function Documentation

8.78.2.1 unsigned int `get_time` ()

a simple way to get the current time since boot in milliseconds (see [TimeET](#) for better accuracy)

Definition at line 10 of file `get_time.cc`.

References `time`.

8.79 `get_time.h` File Reference

8.79.1 Detailed Description

prototype for [get_time\(\)](#), a simple way to get the current time since boot in milliseconds

Author:

ejt (Creator)

Author

alokl

Name

tekkotsu-1_4_1

Revision

1.1.1.1

State

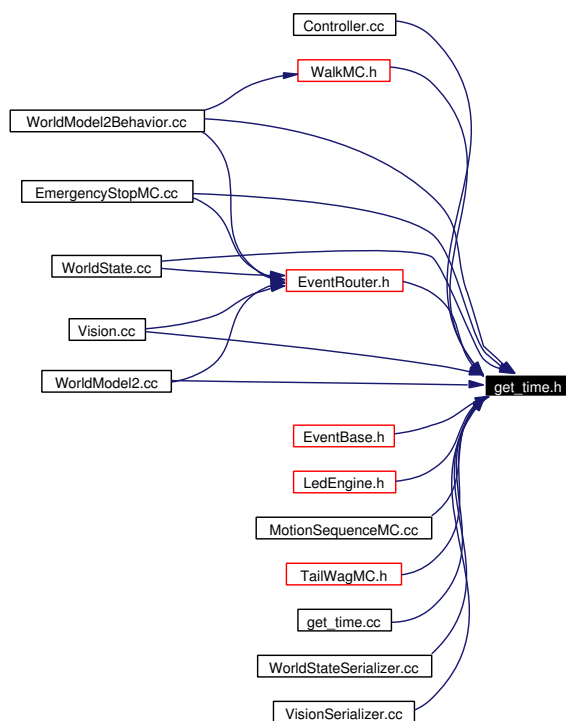
Rel

Date

2002/09/30 18:19:48

Definition in file [get_time.h](#).

This graph shows which files directly or indirectly include this file:



Functions

- unsigned int [get_time](#) ()
a simple way to get the current time since boot in milliseconds (see [TimeET](#) for better accuracy)

8.79.2 Function Documentation

8.79.2.1 unsigned int [get_time](#) ()

a simple way to get the current time since boot in milliseconds (see [TimeET](#) for better accuracy)

Definition at line 10 of file [get_time.cc](#).

References [time](#).

8.80 gvector.h File Reference

8.80.1 Detailed Description

vector class to aid in mathematical vector calculations

Author:

James R. Bruce (Creator)

```
=====
Vector.h : Simple vector class for 2D and 3D vectors
-----
Copyright (C) 1999-2002 James R. Bruce
School of Computer Science, Carnegie Mellon University
-----
This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
=====
*
```

Author

ejt

Name

tekkotsu-1.4_1

Revision

1.4

State

Rel

Date

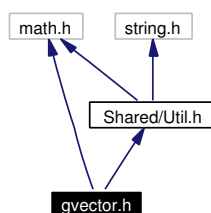
2003/01/23 18:14:04

Definition in file [gvector.h](#).

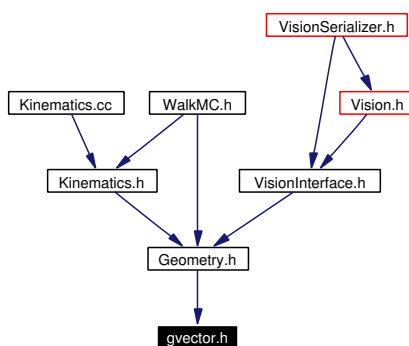
```
#include <math.h>
```

```
#include "Shared/Util.h"
```

Include dependency graph for gvector.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [GVector](#)

Defines

- #define [V3COMP\(p\)](#) p.x,p.y,p.z
- #define [V2COMP\(p\)](#) p.x,p.y
- #define [VECTOR3D_EQUAL_BINARY_OPERATOR](#)(opr)
- #define [VECTOR3D_BINARY_OPERATOR](#)(opr)
- #define [VECTOR3D_SCALAR_OPERATOR](#)(opr)
- #define [VECTOR3D_EQUAL_SCALAR_OPERATOR](#)(opr)
- #define [VECTOR3D_LOGIC_OPERATOR](#)(opr, combine)
- #define [VECTOR2D_EQUAL_BINARY_OPERATOR](#)(opr)
- #define [VECTOR2D_BINARY_OPERATOR](#)(opr)
- #define [VECTOR2D_SCALAR_OPERATOR](#)(opr)
- #define [VECTOR2D_EQUAL_SCALAR_OPERATOR](#)(opr)
- #define [VECTOR2D_LOGIC_OPERATOR](#)(opr, combine)

8.80.2 Define Documentation

8.80.2.1 #define V2COMP(p) p.x,p.y

Definition at line 23 of file gvector.h.

8.80.2.2 #define V3COMP(p) p.x,p.y,p.z

Definition at line 22 of file gvector.h.

8.80.2.3 #define VECTOR2D_BINARY_OPERATOR(opr)

Value:

```
template <class num> \
    vector2d<num> vector2d<num>::operator opr (const vector2d<num> p) const \
    { \
        vector2d<num> r; \
        r.x = x opr p.x; \
        r.y = y opr p.y; \
        return(r); \
    }
```

8.80.2.4 #define VECTOR2D_EQUAL_BINARY_OPERATOR(opr)

Value:

```
template <class num> \
    vector2d<num> vector2d<num>::operator opr (const vector2d<num> p) \
    { \
        x = x opr p.x; \
        y = y opr p.y; \
        return(*this); \
    }
```

Definition at line 466 of file gvector.h.

8.80.2.5 #define VECTOR2D_EQUAL_SCALAR_OPERATOR(opr)

Value:

```
template <class num> \
    vector2d<num> vector2d<num>::operator opr (num f) \
    { \
        x = x opr f; \
    }
```

```

        y = y opr f; \
        return(*this); \
    }

```

8.80.2.6 #define VECTOR2D_LOGIC_OPERATOR(opr, combine)

Value:

```

template <class num> \
    bool vector2d<num>::operator opr (const vector2d<num> p) const \
    { \
        return((x opr p.x) combine \
                (y opr p.y)); \
    }

```

8.80.2.7 #define VECTOR2D_SCALAR_OPERATOR(opr)

Value:

```

template <class num> \
    vector2d<num> vector2d<num>::operator opr (const num f) const \
    { \
        vector2d<num> r; \
        r.x = x opr f; \
        r.y = y opr f; \
        return(r); \
    }

```

8.80.2.8 #define VECTOR3D_BINARY_OPERATOR(opr)

Value:

```

template <class num> \
    vector3d<num> vector3d<num>::operator opr (const vector3d<num> p) const \
    { \
        vector3d<num> r; \
        r.x = x opr p.x; \
        r.y = y opr p.y; \
        r.z = z opr p.z; \
        return(r); \
    }

```

8.80.2.9 #define VECTOR3D_EQUAL_BINARY_OPERATOR(opr)**Value:**

```
template <class num> \
    vector3d<num> vector3d<num>::operator opr (const vector3d<num> p) \
    { \
        x = x opr p.x; \
        y = y opr p.y; \
        z = z opr p.z; \
        return(*this); \
    }
```

Definition at line 170 of file gvector.h.

8.80.2.10 #define VECTOR3D_EQUAL_SCALAR_OPERATOR(opr)**Value:**

```
template <class num> \
    vector3d<num> vector3d<num>::operator opr (num f) \
    { \
        x = x opr f; \
        y = y opr f; \
        z = z opr f; \
        return(*this); \
    }
```

8.80.2.11 #define VECTOR3D_LOGIC_OPERATOR(opr, combine)**Value:**

```
template <class num> \
    bool vector3d<num>::operator opr (const vector3d<num> p) const \
    { \
        return((x opr p.x) combine \
                (y opr p.y) combine \
                (z opr p.z)); \
    }
```

8.80.2.12 #define VECTOR3D_SCALAR_OPERATOR(opr)**Value:**

```
template <class num> \
    vector3d<num> vector3d<num>::operator opr (const num f) const \
```

```
{  
    vector3d<num> r; \  
    r.x = x opr f;  \  
    r.y = y opr f;  \  
    r.z = z opr f;  \  
    return(r);      \  
}
```

8.81 HeadLevelBehavior.h File Reference

8.81.1 Detailed Description

Defines [HeadLevelBehavior](#), which a prototypes head leveler.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.3

State

Rel

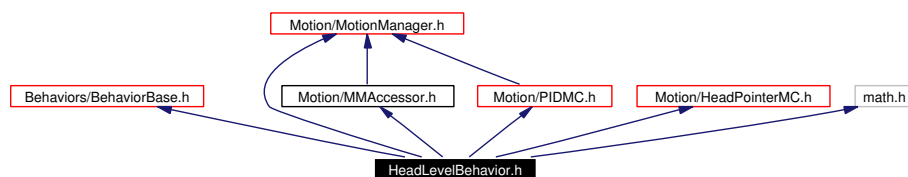
Date

2003/06/05 17:03:15

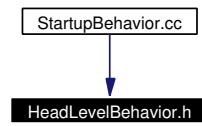
Definition in file [HeadLevelBehavior.h](#).

```
#include "Behaviors/BehaviorBase.h"
#include "Motion/MotionManager.h"
#include "Motion/MMAccessor.h"
#include "Motion/HeadPointerMC.h"
#include "Motion/PIDMC.h"
#include <math.h>
```

Include dependency graph for HeadLevelBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [HeadLevelBehavior](#)

Tests the head leveling code of [HeadPointerMC](#).

8.82 HeadPointControllerBehavior.cc File Reference

8.82.1 Detailed Description

Implements [HeadPointControllerBehavior](#), listens to control commands coming in from the command port for remotely controlling the head.

Author:

tss (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.1

State

Rel

Date

2003/07/07 01:00:08

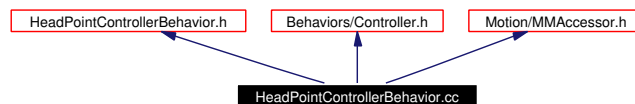
Definition in file [HeadPointControllerBehavior.cc](#).

```
#include "HeadPointControllerBehavior.h"
```

```
#include "Behaviors/Controller.h"
```

```
#include "Motion/MMAccessor.h"
```

Include dependency graph for HeadPointControllerBehavior.cc:



8.83 HeadPointControllerBehavior.h File Reference

8.83.1 Detailed Description

Describes [HeadPointControllerBehavior](#), listens to control commands coming in from the command port for remotely controlling the head.

Author:

tss (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.1

State

Rel

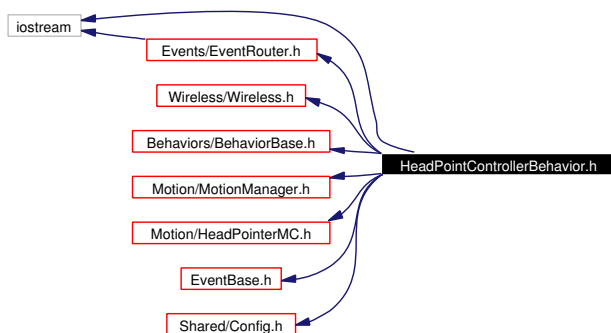
Date

2003/07/07 01:00:08

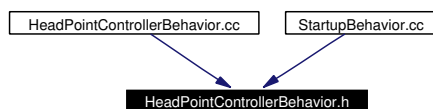
Definition in file [HeadPointControllerBehavior.h](#).

```
#include <iostream>
#include "Wireless/Wireless.h"
#include "Behaviors/BehaviorBase.h"
#include "Motion/MotionManager.h"
#include "Motion/HeadPointerMC.h"
#include "Events/EventRouter.h"
#include "Events/EventBase.h"
#include "Shared/Config.h"
```

Include dependency graph for HeadPointControllerBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [HeadPointControllerBehavior](#)

Listens to control commands coming in from the command port for remotely controlling the head.

8.84 HeadPointerMC.cc File Reference

8.84.1 Detailed Description

Implements [HeadPointerMC](#), a class for various ways to control where the head is looking.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.6

State

Rel

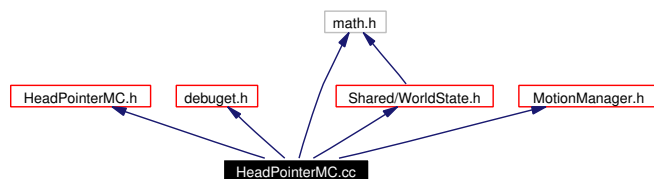
Date

2003/06/12 23:41:40

Definition in file [HeadPointerMC.cc](#).

```
#include "HeadPointerMC.h"
#include "Shared/debuget.h"
#include "Shared/WorldState.h"
#include "MotionManager.h"
#include <math.h>
```

Include dependency graph for HeadPointerMC.cc:



8.85 HeadPointerMC.h File Reference

8.85.1 Detailed Description

Describes [HeadPointerMC](#), a class for various ways to control where the head is looking.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.5

State

Rel

Date

2003/07/07 01:00:08

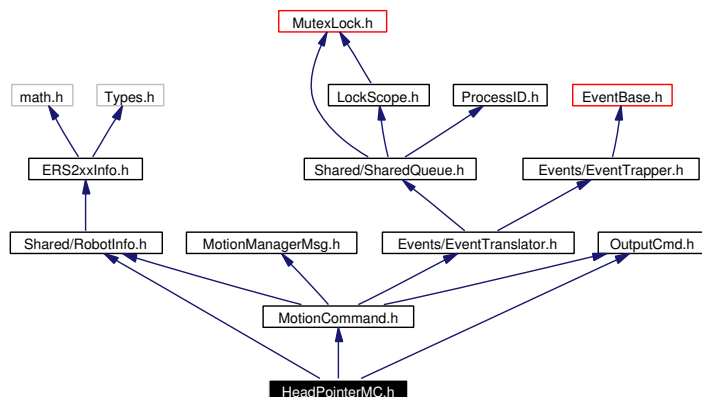
Definition in file [HeadPointerMC.h](#).

```
#include "MotionCommand.h"
```

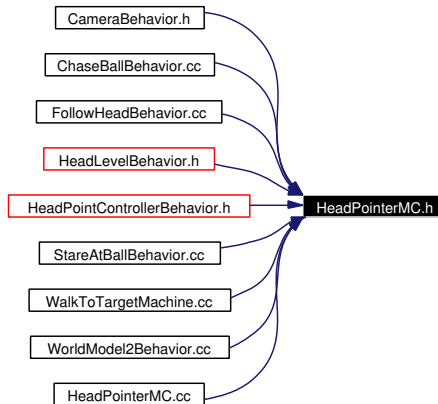
```
#include "OutputCmd.h"
```

```
#include "Shared/RobotInfo.h"
```

Include dependency graph for HeadPointerMC.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [HeadPointerMC](#)

This class gives some quick and easy functions to point the head at things.

8.86 HelpControl.cc File Reference

8.86.1 Detailed Description

Implements [HelpControl](#), which recurses through the menu system and outputs the name and description of each item.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

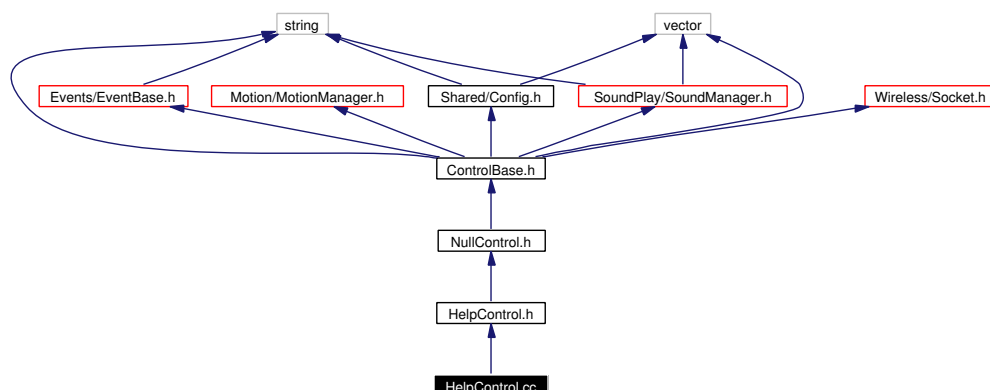
Date

2003/07/09 00:10:57

Definition in file [HelpControl.cc](#).

```
#include "HelpControl.h"
```

Include dependency graph for HelpControl.cc:



8.87 HelpControl.h File Reference

8.87.1 Detailed Description

Describes [HelpControl](#), which recurses through the menu system and outputs the name and description of each item.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

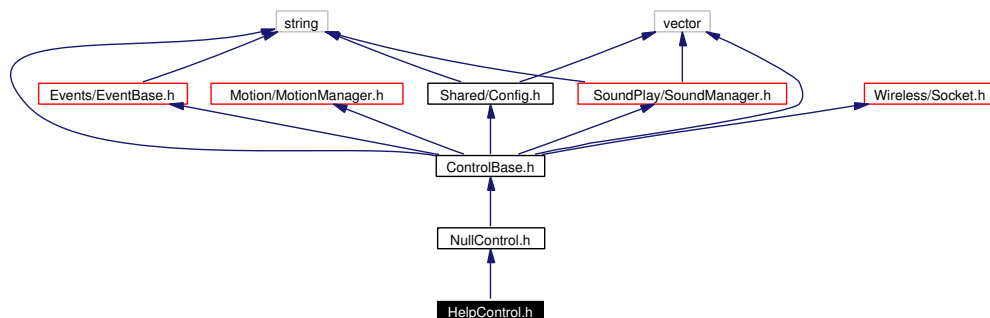
Date

2003/06/12 23:41:36

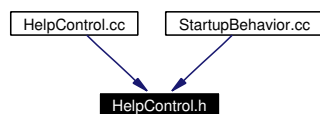
Definition in file [HelpControl.h](#).

```
#include "NullControl.h"
```

Include dependency graph for HelpControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

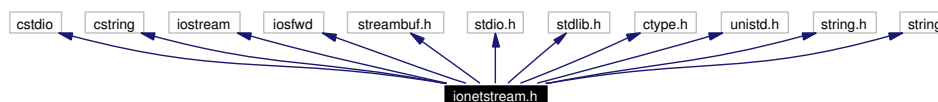
- class [HelpControl](#)

Recurses through the menu system and outputs the name and description of each item.

8.88 ionetstream.h File Reference

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <iosfwd>
#include <streambuf.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <unistd.h>
#include <string.h>
#include <string>
```

Include dependency graph for ionetstream.h:



Compounds

- class [basic_iNetStream](#)
- class [basic_ioNetStream](#)
- class [basic_netbuf](#)
- class [basic_oNetStream](#)
- class [char_traits](#)

Defines

- #define [INVALID_SOCKET](#) (~0)

Typedefs

- typedef [basic_netbuf](#)< char, [char_traits](#)< char > > [netbuf](#)
- typedef [basic_iNetStream](#)< char, [char_traits](#)< char > > [iNetStream](#)
- typedef [basic_oNetStream](#)< char, [char_traits](#)< char > > [oNetStream](#)

- typedef [basic_ioNetStream](#)< char, [char_traits](#)< char > > [ioNetStream](#)

8.88.1 Define Documentation

8.88.1.1 #define INVALID_SOCKET (~0)

Definition at line 18 of file ionetstream.h.

8.88.2 Typedef Documentation

8.88.2.1 typedef [basic_iNetStream](#)<char, [char_traits](#)<char> > [iNetStream](#)

Definition at line 226 of file ionetstream.h.

8.88.2.2 typedef [basic_ioNetStream](#)<char, [char_traits](#)<char> > [ioNetStream](#)

Definition at line 228 of file ionetstream.h.

8.88.2.3 typedef [basic_netbuf](#)<char, [char_traits](#)<char> > [netbuf](#)

Definition at line 225 of file ionetstream.h.

8.88.2.4 typedef [basic_oNetStream](#)<char, [char_traits](#)<char> > [oNetStream](#)

Definition at line 227 of file ionetstream.h.

8.89 karmedbandit.h File Reference

8.89.1 Detailed Description

Defines karmedbandit - implements an algorithm which makes decisions regarding an adversarial k-armed bandit.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

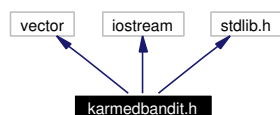
Date

2003/03/03 01:18:12

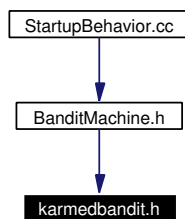
Definition in file [karmedbandit.h](#).

```
#include <vector>
#include <iostream>
#include <stdlib.h>
```

Include dependency graph for karmedbandit.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [karmedbanditExp3](#)
Makes decisions regarding an adversarial k -armed bandit.
- class [karmedbanditExp3_1](#)
Makes decisions regarding an adversarial k -armed bandit.

8.90 Kinematics.cc File Reference

8.90.1 Detailed Description

Functions to provide kinematics calculations.

Author:

CMU RoboSoccer 2001-2002 (Creator)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
-----

This software is distributed under the GNU General Public License,
version 2.  If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA.  This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
-----

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed.  In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

Date

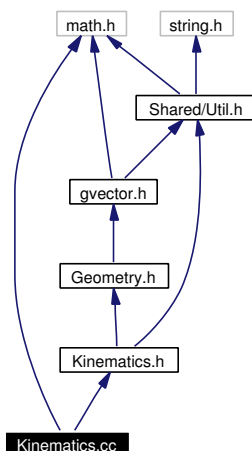
2003/01/17 23:15:51

Definition in file [Kinematics.cc](#).

```
#include <math.h>
```

```
#include "Kinematics.h"
```

Include dependency graph for Kinematics.cc:



Functions

- const [vector3d](#) [f_body_to_shoulder](#) (59.50, 59.20, 0.00)
- const [vector3d](#) [f_leg_shoulder_to_knee](#) (12.80, 0.50,-64.00)
- const [vector3d](#) [f_leg_knee_to_ball](#) (-18.00, 0.00,-67.23)
- const [vector3d](#) [h_body_to_shoulder](#) (-59.50, 59.20, 0.00)
- const [vector3d](#) [h_leg_shoulder_to_knee](#) (-12.80, 0.50,-64.00)
- const [vector3d](#) [h_leg_knee_to_ball](#) (18.00, 0.00,-74.87)
- void [KinClearErrors](#) ()
- int [KinGetErrors](#) ()
- double [GetTrigAngle](#) (double a, double b, double d, double mn, double mx, bool add)
- void [GetLegAngles](#) (double *angles, [vector3d](#) target, int leg)
- void [GetLegAngles](#) (double *angles, [vector3d](#) target[4], double body_angle, double body_height)
- void [GetLegAngles](#) (double *angles, [vector3d](#) target[4], [BodyPosition](#) &bp)
- void [GetLegAngles](#) (double *angles, [vector3d](#) target, [BodyPosition](#) &bp, int leg)
- void [GetLegPosition](#) ([vector3d](#) &p, const double *ang, int leg)
- void [GetBodyLocation](#) ([vector3d](#) &ball, [vector3d](#) &toe, const double *ang, int leg)

- void [GetHeadAngles](#) (double *angles, [vector3d](#) target, double body_angle, double body_height)
- [vector3d RunForwardModel](#) (double *angles, double body_angle, double body_height, [vector3d](#) point)
- void [GetHeadPosition](#) ([vector3d](#) &location, [vector3d](#) &direction, [vector3d](#) &up, [vector3d](#) &right, double *angles, double body_angle, double body_height)

Variables

- const [vector3d f_upper](#) = f_leg_shoulder_to_knee
- const [vector3d f_lower](#) = f_leg_knee_to_ball
- const [vector3d h_upper](#) = h_leg_shoulder_to_knee
- const [vector3d h_lower](#) = h_leg_knee_to_ball
- int [errors](#)
- const double [rotator_min](#) = RAD(-117.0)
- const double [rotator_max](#) = RAD(117.0)
- const double [shoulder_min](#) = RAD(-11.0)
- const double [shoulder_max](#) = RAD(97.0)
- const double [knee_max](#) = RAD(147.0)
- const double [knee_min](#) = RAD(-27.0)
- const double [rotator_kmin](#) = RAD(-90.0)
- const double [rotator_kmax](#) = RAD(90.0)
- const double [shoulder_kmin](#) = [shoulder_min](#)
- const double [shoulder_kmax](#) = RAD(90.0)
- const double [knee_kmax](#) = [knee_max](#)
- const double [f_knee_kmin](#) = 0.2616
- const double [h_knee_kmin](#) = 0.2316
- const double [tail_min](#) = RAD(-22)
- const double [tail_max](#) = RAD(22)
- const double [head_tilt_min](#) = RAD(-88.5)
- const double [head_tilt_max](#) = RAD(43.0)
- const double [head_pan_min](#) = RAD(-89.6)
- const double [head_pan_max](#) = RAD(89.6)
- const double [head_roll_min](#) = RAD(-29.0)
- const double [head_roll_max](#) = RAD(29.0)

8.90.2 Function Documentation

8.90.2.1 `const vector3d f_body_to_shoulder (59. 50, 59. 20, 0. 00)`

8.90.2.2 `const vector3d f_leg_knee_to_ball (-18. 00, 0. 00, -67. 23)`

8.90.2.3 `const vector3d f_leg_shoulder_to_knee (12. 80, 0. 50, -64. 00)`

8.90.2.4 `void GetBodyLocation (vector3d & ball, vector3d & toe, const double * ang, int leg)`

Definition at line 315 of file Kinematics.cc.

References `f_body_to_shoulder()`, `f_leg_knee_to_ball()`, `f_leg_shoulder_to_knee()`, `h_body_to_shoulder()`, `h_leg_knee_to_ball()`, `h_leg_shoulder_to_knee()`, `GVector::vector3d< double >::rotate_x()`, `GVector::vector3d< double >::rotate_y()`, and `GVector::vector3d< double >::y`.

8.90.2.5 `void GetHeadAngles (double * angles, vector3d target, double body_angle, double body_height)`

Definition at line 337 of file Kinematics.cc.

References `body_to_neck()`, `bound()`, `head_pan_max`, `head_pan_min`, `head_tilt_max`, `head_tilt_min`, `GVector::vector3d< double >::length()`, `neck_to_camera()`, `GVector::vector3d< double >::rotate_y()`, `GVector::vector3d< double >::x`, `GVector::vector3d< double >::y`, and `GVector::vector3d< double >::z`.

8.90.2.6 `void GetHeadPosition (vector3d & location, vector3d & direction, vector3d & up, vector3d & right, double * angles, double body_angle, double body_height)`

Definition at line 423 of file Kinematics.cc.

References `GVector::vector3d< double >::norm()`, `RunForwardModel()`, and `vector3d`.

8.90.2.7 `void GetLegAngles (double * angles, vector3d target, BodyPosition & bp, int leg)`

Definition at line 289 of file Kinematics.cc.

References `BodyPosition::angle`, `BodyPosition::loc`, `GVector::vector3d< double >::rotate_z()`, `GVector::vector3d< double >::y`, and `GVector::vector3d< double >::z`.

8.90.2.8 void GetLegAngles (double * *angles*, [vector3d](#) *target*[4], [BodyPosition](#) & *bp*)

Definition at line 278 of file Kinematics.cc.

8.90.2.9 void GetLegAngles (double * *angles*, [vector3d](#) *target*[4], double *body_angle*, double *body_height*)

Definition at line 264 of file Kinematics.cc.

8.90.2.10 void GetLegAngles (double * *angles*, [vector3d](#) *target*, int *leg*)

Definition at line 185 of file Kinematics.cc.

8.90.2.11 void GetLegPosition ([vector3d](#) & *p*, const double * *ang*, int *leg*)

Definition at line 298 of file Kinematics.cc.

References `f_body_to_shoulder()`, `f_leg_knee_to_ball()`, `f_leg_shoulder_to_knee()`, `h_body_to_shoulder()`, `h_leg_knee_to_ball()`, `h_leg_shoulder_to_knee()`, `GVector::vector3d< double >::rotate_x()`, `GVector::vector3d< double >::rotate_y()`, and `GVector::vector3d< double >::y`.

8.90.2.12 double GetTrigAngle (double *a*, double *b*, double *d*, double *mn*, double *mx*, bool *add*)

Definition at line 101 of file Kinematics.cc.

References errors.

8.90.2.13 const [vector3d](#) h_body_to_shoulder (-59. 50, 59. 20, 0. 00)

8.90.2.14 const [vector3d](#) h_leg_knee_to_ball (18. 00, 0. 00, -74. 87)

8.90.2.15 const [vector3d](#) h_leg_shoulder_to_knee (-12. 80, 0. 50, -64. 00)

8.90.2.16 void KinClearErrors ()

Definition at line 91 of file Kinematics.cc.

References errors.

8.90.2.17 int KinGetErrors ()

Definition at line 96 of file Kinematics.cc.

References errors.

8.90.2.18 **vector3d** RunForwardModel (double * *angles*, double *body_angle*, double *body_height*, **vector3d** *point*)

Definition at line 395 of file Kinematics.cc.

References body_to_neck(), neck_to_camera(), GVector::vector3d< double >::rotate_x(), GVector::vector3d< double >::rotate_y(), GVector::vector3d< double >::rotate_z(), and GVector::vector3d< double >::z.

8.90.3 Variable Documentation

8.90.3.1 int **errors** [static]

Definition at line 89 of file Kinematics.cc.

8.90.3.2 const double **f.knee.kmin** = 0.2616

Definition at line 171 of file Kinematics.cc.

8.90.3.3 const **vector3d** **f.lower** = f_leg_knee_to_ball

Definition at line 79 of file Kinematics.cc.

8.90.3.4 const **vector3d** **f.upper** = f_leg_shoulders_to_knee

Definition at line 78 of file Kinematics.cc.

8.90.3.5 const double **h.knee.kmin** = 0.2316

Definition at line 172 of file Kinematics.cc.

8.90.3.6 const **vector3d** **h.lower** = h_leg_knee_to_ball

Definition at line 81 of file Kinematics.cc.

8.90.3.7 `const vector3d h_upper = h_leg_shoulder_to_knee`

Definition at line 80 of file Kinematics.cc.

8.90.3.8 `const double head_pan_max = RAD(89.6)`

Definition at line 180 of file Kinematics.cc.

8.90.3.9 `const double head_pan_min = RAD(-89.6)`

Definition at line 179 of file Kinematics.cc.

8.90.3.10 `const double head_roll_max = RAD(29.0)`

Definition at line 182 of file Kinematics.cc.

8.90.3.11 `const double head_roll_min = RAD(-29.0)`

Definition at line 181 of file Kinematics.cc.

8.90.3.12 `const double head_tilt_max = RAD(43.0)`

Definition at line 178 of file Kinematics.cc.

8.90.3.13 `const double head_tilt_min = RAD(-88.5)`

Definition at line 177 of file Kinematics.cc.

8.90.3.14 `const double knee_kmax = knee_max`

Definition at line 170 of file Kinematics.cc.

8.90.3.15 `const double knee_max = RAD(147.0)`

Definition at line 163 of file Kinematics.cc.

8.90.3.16 `const double knee_min = RAD(-27.0)`

Definition at line 164 of file Kinematics.cc.

8.90.3.17 `const double rotator_kmax = RAD(90.0)`

Definition at line 167 of file Kinematics.cc.

8.90.3.18 `const double rotator_kmin = RAD(-90.0)`

Definition at line 166 of file Kinematics.cc.

8.90.3.19 `const double rotator_max = RAD(117.0)`

Definition at line 160 of file Kinematics.cc.

8.90.3.20 `const double rotator_min = RAD(-117.0)`

Definition at line 159 of file Kinematics.cc.

8.90.3.21 `const double shoulder_kmax = RAD(90.0)`

Definition at line 169 of file Kinematics.cc.

8.90.3.22 `const double shoulder_kmin = shoulder_min`

Definition at line 168 of file Kinematics.cc.

8.90.3.23 `const double shoulder_max = RAD(97.0)`

Definition at line 162 of file Kinematics.cc.

8.90.3.24 `const double shoulder_min = RAD(-11.0)`

Definition at line 161 of file Kinematics.cc.

8.90.3.25 `const double tail_max = RAD(22)`

Definition at line 175 of file Kinematics.cc.

8.90.3.26 `const double tail_min = RAD(-22)`

Definition at line 174 of file Kinematics.cc.

8.91 Kinematics.h File Reference

8.91.1 Detailed Description

Functions to provide kinematics calculations.

Author:

CMU RoboSoccer 2001-2002 (Creator)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
-----

This software is distributed under the GNU General Public License,
version 2.  If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA.  This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
-----

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed.  In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.6

State

Rel

Date

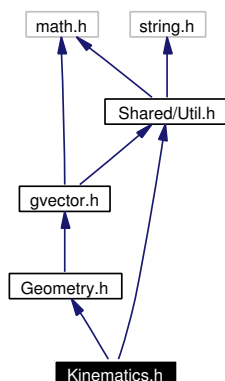
2003/03/03 01:18:13

Definition in file [Kinematics.h](#).

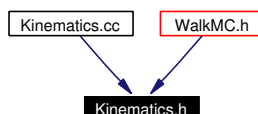
```
#include "Geometry.h"
```

```
#include "Shared/Util.h"
```

Include dependency graph for Kinematics.h:



This graph shows which files directly or indirectly include this file:



Compounds

- struct [BodyPosition](#)
holds the current location of the body, as a delta from when walking started

Defines

- #define [RAD](#)(deg) (((deg) * M_PI) / 180.0)
- #define [DEG](#)(rad) (((rad) * 180.0) / M_PI)

Functions

- const [vector3d](#) [body_to_neck](#) (75.00, 0.00, 50.00)

- const [vector3d](#) [neck_to_camera](#) (65.00, 0.00, 48.00)
- void [KinClearErrors](#) ()
- int [KinGetErrors](#) ()
- void [GetLegAngles](#) (double *angles, [vector3d](#) target, int leg)
- void [GetLegAngles](#) (double *angles, [vector3d](#) target[4], double body_angle, double body_height)
- void [GetLegAngles](#) (double *angles, [vector3d](#) target[4], [BodyPosition](#) &bp)
- void [GetLegAngles](#) (double *angles, [vector3d](#) target, [BodyPosition](#) &bp, int leg)
- void [GetLegPosition](#) ([vector3d](#) &p, const double *ang, int leg)
- void [GetBodyLocation](#) ([vector3d](#) &ball, [vector3d](#) &toe, const double *ang, int leg)
- void [GetHeadAngles](#) (double *angles, [vector3d](#) target, double body_angle, double body_height)
- [vector3d](#) [RunForwardModel](#) (double *angles, double body_angle, double body_height, [vector3d](#) point)
- void [GetHeadPosition](#) ([vector3d](#) &location, [vector3d](#) &direction, [vector3d](#) &up, [vector3d](#) &right, double *angles, double body_angle, double body_height)

Variables

- const double [rotator_min](#)
- const double [rotator_max](#)
- const double [shoulder_min](#)
- const double [shoulder_max](#)
- const double [knee_max](#)
- const double [knee_min](#)
- const double [rotator_kmin](#)
- const double [rotator_kmax](#)
- const double [shoulder_kmin](#)
- const double [shoulder_kmax](#)
- const double [knee_kmax](#)
- const double [f_knee_kmin](#)
- const double [h_knee_kmin](#)
- const double [tail_min](#)
- const double [tail_max](#)
- const double [head_tilt_min](#)
- const double [head_tilt_max](#)
- const double [head_pan_min](#)
- const double [head_pan_max](#)
- const double [head_roll_min](#)
- const double [head_roll_max](#)

8.91.2 Define Documentation

8.91.2.1 `#define DEG(rad) (((rad) * 180.0) / M_PI)`

Definition at line 38 of file Kinematics.h.

8.91.2.2 `#define RAD(deg) (((deg) * M_PI) / 180.0)`

Definition at line 37 of file Kinematics.h.

8.91.3 Function Documentation

8.91.3.1 `const vector3d body_to_neck (75. 00, 0. 00, 50. 00)`

8.91.3.2 `void GetBodyLocation (vector3d & ball, vector3d & toe, const double * ang, int leg)`

Definition at line 315 of file Kinematics.cc.

References `f_body_to_shoulder()`, `f_leg_knee_to_ball()`, `f_leg_shoulder_to_knee()`, `h_body_to_shoulder()`, `h_leg_knee_to_ball()`, `h_leg_shoulder_to_knee()`, `GVector::vector3d< double >::rotate_x()`, `GVector::vector3d< double >::rotate_y()`, and `GVector::vector3d< double >::y`.

8.91.3.3 `void GetHeadAngles (double * angles, vector3d target, double body_angle, double body_height)`

Definition at line 337 of file Kinematics.cc.

References `body_to_neck()`, `bound()`, `head_pan_max`, `head_pan_min`, `head_tilt_max`, `head_tilt_min`, `GVector::vector3d< double >::length()`, `neck_to_camera()`, `GVector::vector3d< double >::rotate_y()`, `GVector::vector3d< double >::x`, `GVector::vector3d< double >::y`, and `GVector::vector3d< double >::z`.

8.91.3.4 `void GetHeadPosition (vector3d & location, vector3d & direction, vector3d & up, vector3d & right, double * angles, double body_angle, double body_height)`

Definition at line 423 of file Kinematics.cc.

References `GVector::vector3d< double >::norm()`, `RunForwardModel()`, and `vector3d`.

8.91.3.5 void GetLegAngles (double * *angles*, [vector3d](#) *target*, [BodyPosition](#) & *bp*, int *leg*)

Definition at line 289 of file Kinematics.cc.

References [BodyPosition::angle](#), [BodyPosition::loc](#), [GVector::vector3d< double >::rotate_z\(\)](#), [GVector::vector3d< double >::y](#), and [GVector::vector3d< double >::z](#).

8.91.3.6 void GetLegAngles (double * *angles*, [vector3d](#) *target*[4], [BodyPosition](#) & *bp*)

Definition at line 278 of file Kinematics.cc.

References [BodyPosition::angle](#), [GetLegAngles\(\)](#), [BodyPosition::loc](#), [GVector::vector3d< double >::rotate_z\(\)](#), [GVector::vector3d< double >::y](#), and [GVector::vector3d< double >::z](#).

8.91.3.7 void GetLegAngles (double * *angles*, [vector3d](#) *target*[4], double *body_angle*, double *body_height*)

Definition at line 264 of file Kinematics.cc.

References [GetLegAngles\(\)](#), [GVector::vector3d< double >::rotate_y\(\)](#), and [GVector::vector3d< double >::z](#).

8.91.3.8 void GetLegAngles (double * *angles*, [vector3d](#) *target*, int *leg*)

Definition at line 185 of file Kinematics.cc.

References [f_body_to_shoulder\(\)](#), [f_knee_kmin](#), [f_leg_knee_to_ball\(\)](#), [f_leg_shoulder_to_knee\(\)](#), [f_lower](#), [f_upper](#), [GetTrigAngle\(\)](#), [h_body_to_shoulder\(\)](#), [h_knee_kmin](#), [h_leg_knee_to_ball\(\)](#), [h_leg_shoulder_to_knee\(\)](#), [h_lower](#), [h_upper](#), [knee_kmax](#), [GVector::vector3d< double >::rotate_y\(\)](#), [rotator_max](#), [rotator_min](#), [shoulder_kmax](#), [shoulder_kmin](#), [GVector::vector3d< double >::sqlength\(\)](#), [GVector::vector3d< double >::x](#), [GVector::vector3d< double >::y](#), and [GVector::vector3d< double >::z](#).

8.91.3.9 void GetLegPosition ([vector3d](#) & *p*, const double * *ang*, int *leg*)

Definition at line 298 of file Kinematics.cc.

References [f_body_to_shoulder\(\)](#), [f_leg_knee_to_ball\(\)](#), [f_leg_shoulder_to_knee\(\)](#), [h_body_to_shoulder\(\)](#), [h_leg_knee_to_ball\(\)](#), [h_leg_shoulder_to_knee\(\)](#), [GVector::vector3d< double >::rotate_x\(\)](#), [GVector::vector3d< double >::rotate_y\(\)](#), and [GVector::vector3d< double >::y](#).

8.91.3.10 void KinClearErrors ()

Definition at line 91 of file Kinematics.cc.

References errors.

8.91.3.11 int KinGetErrors ()

Definition at line 96 of file Kinematics.cc.

References errors.

8.91.3.12 const [vector3d](#) neck_to_camera (65. 00, 0. 00, 48. 00)**8.91.3.13 [vector3d](#) RunForwardModel (double * angles, double body_angle, double body_height, [vector3d](#) point)**

Definition at line 395 of file Kinematics.cc.

References [body_to_neck\(\)](#), [neck_to_camera\(\)](#), [GVector::vector3d< double >::rotate_x\(\)](#), [GVector::vector3d< double >::rotate_y\(\)](#), [GVector::vector3d< double >::rotate_z\(\)](#), and [GVector::vector3d< double >::z](#).

8.91.4 Variable Documentation**8.91.4.1 const double [f.knee.kmin](#)**

Definition at line 52 of file Kinematics.h.

8.91.4.2 const double [h.knee.kmin](#)

Definition at line 53 of file Kinematics.h.

8.91.4.3 const double [head.pan_max](#)

Definition at line 61 of file Kinematics.h.

8.91.4.4 const double [head.pan_min](#)

Definition at line 60 of file Kinematics.h.

8.91.4.5 const double [head_roll_max](#)

Definition at line 63 of file Kinematics.h.

8.91.4.6 const double [head_roll_min](#)

Definition at line 62 of file Kinematics.h.

8.91.4.7 const double [head_tilt_max](#)

Definition at line 59 of file Kinematics.h.

8.91.4.8 const double [head_tilt_min](#)

Definition at line 58 of file Kinematics.h.

8.91.4.9 const double [knee_kmax](#)

Definition at line 51 of file Kinematics.h.

8.91.4.10 const double [knee_max](#)

Definition at line 44 of file Kinematics.h.

8.91.4.11 const double [knee_min](#)

Definition at line 45 of file Kinematics.h.

8.91.4.12 const double [rotator_kmax](#)

Definition at line 48 of file Kinematics.h.

8.91.4.13 const double [rotator_kmin](#)

Definition at line 47 of file Kinematics.h.

8.91.4.14 const double [rotator_max](#)

Definition at line 41 of file Kinematics.h.

8.91.4.15 **const double** [rotator_min](#)

Definition at line 40 of file Kinematics.h.

8.91.4.16 **const double** [shoulder_kmax](#)

Definition at line 50 of file Kinematics.h.

8.91.4.17 **const double** [shoulder_kmin](#)

Definition at line 49 of file Kinematics.h.

8.91.4.18 **const double** [shoulder_max](#)

Definition at line 43 of file Kinematics.h.

8.91.4.19 **const double** [shoulder_min](#)

Definition at line 42 of file Kinematics.h.

8.91.4.20 **const double** [tail_max](#)

Definition at line 56 of file Kinematics.h.

8.91.4.21 **const double** [tail_min](#)

Definition at line 55 of file Kinematics.h.

8.92 LedEngine.cc File Reference

8.92.1 Detailed Description

Implements [LedEngine](#), which provides basic LED effects to anything that inherits or instantiates it.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.8

State

Rel

Date

2003/06/10 18:30:00

Definition in file [LedEngine.cc](#).

```
#include "LedEngine.h"  
#include "MotionManager.h"  
#include "Shared/WorldState.h"  
#include "Shared/ERS210Info.h"  
#include "Shared/ERS220Info.h"
```

Include dependency graph for LedEngine.cc:



8.93 LedEngine.h File Reference

8.93.1 Detailed Description

Describes [LedEngine](#), which provides basic LED effects to anything that inherits or instantiates it.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.9

State

Rel

Date

2003/06/12 18:06:11

Definition in file [LedEngine.h](#).

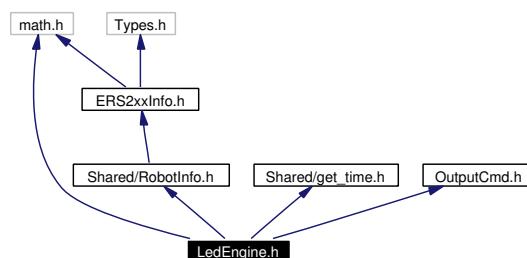
```
#include "Shared/RobotInfo.h"
```

```
#include "Shared/get_time.h"
```

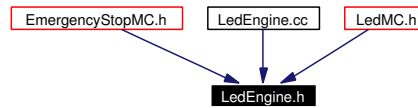
```
#include "OutputCmd.h"
```

```
#include <math.h>
```

Include dependency graph for LedEngine.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [LedEngine](#)
Provides basic LED effects to anything that inherits from (recommended method for MotionCommands) or instantiates it (just in case you have reason to).
- struct [LedEngine.LEDInfo](#)
Holds all the information needed by each of the LEDs.

8.94 LedMC.h File Reference

8.94.1 Detailed Description

Defines [LedMC](#), which provides a basic [MotionCommand](#) wrapper to [LedEngine](#).

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.5

State

Rel

Date

2003/03/04 07:00:15

Definition in file [LedMC.h](#).

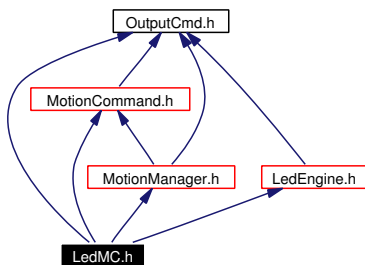
```
#include "MotionCommand.h"
```

```
#include "LedEngine.h"
```

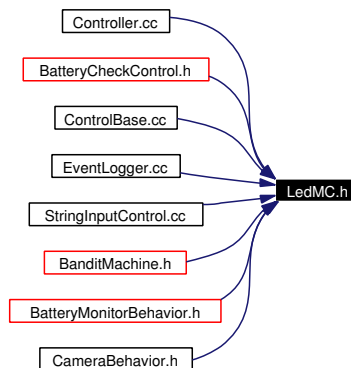
```
#include "OutputCmd.h"
```

```
#include "MotionManager.h"
```

Include dependency graph for LedMC.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [LedMC](#)

This is just a simple wrapper - you probably want to be looking at [LedEngine.h](#).

8.95 ListMemBuf.h File Reference

8.95.1 Detailed Description

Defines [ListMemBuf](#), which provides some degree of dynamic allocation of a templated type from a buffer of set size.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.7

State

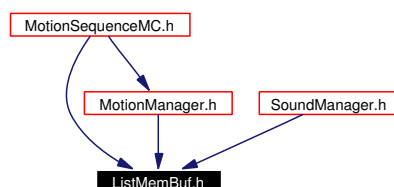
Rel

Date

2003/03/04 05:46:07

Definition in file [ListMemBuf.h](#).

This graph shows which files directly or indirectly include this file:



Compounds

- class [ListMemBuf](#)
Provides some degree of dynamic allocation of a templated type from a buffer of set size.
- struct [ListMemBuf.entry_t](#)
holds data about an entry in the free/used lists

8.96 LoadPostureControl.h File Reference

8.96.1 Detailed Description

Defines [LoadPostureControl](#), which when activated, loads a position from a file name read from cin (stored in ms/data/motion...).

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.9

State

Rel

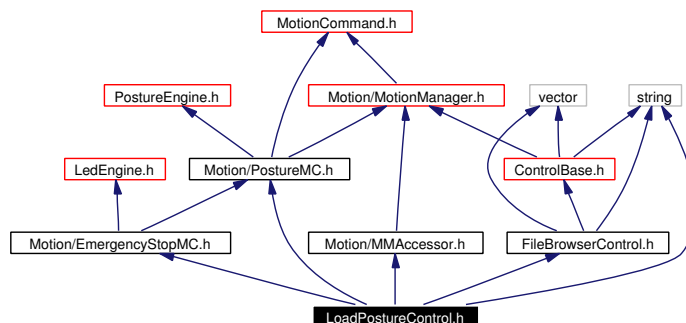
Date

2003/06/10 00:53:48

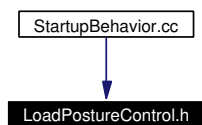
Definition in file [LoadPostureControl.h](#).

```
#include "FileBrowserControl.h"
#include "Motion/PostureMC.h"
#include "Motion/MMAccessor.h"
#include "Motion/EmergencyStopMC.h"
#include <string>
```

Include dependency graph for LoadPostureControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [LoadPostureControl](#)
Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...).

8.97 LoadSave.cc File Reference

8.97.1 Detailed Description

Implements [LoadSave](#), inherit from this to use a standard interface for loading and saving.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

Date

2003/01/09 02:02:59

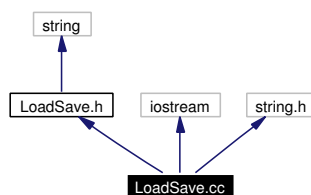
Definition in file [LoadSave.cc](#).

```
#include "LoadSave.h"
```

```
#include <iostream>
```

```
#include <string.h>
```

Include dependency graph for LoadSave.cc:



8.98 LoadSave.h File Reference

8.98.1 Detailed Description

Describes [LoadSave](#), inherit from this to use a standard interface for loading and saving.

Todo

detect appropriate byte ordering for other platforms

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

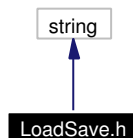
Date

2003/01/10 07:45:21

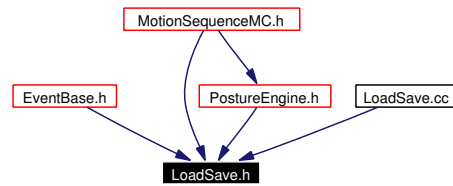
Definition in file [LoadSave.h](#).

```
#include <string>
```

Include dependency graph for LoadSave.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [LoadSave](#)

Intended as an interface to allow easy and uniform file operations.

8.99 LoadWalkControl.h File Reference

8.99.1 Detailed Description

Defines [LoadWalkControl](#), which when activated, loads a set of walk parameters from a file read from cin.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4_1

Revision

1.4

State

Rel

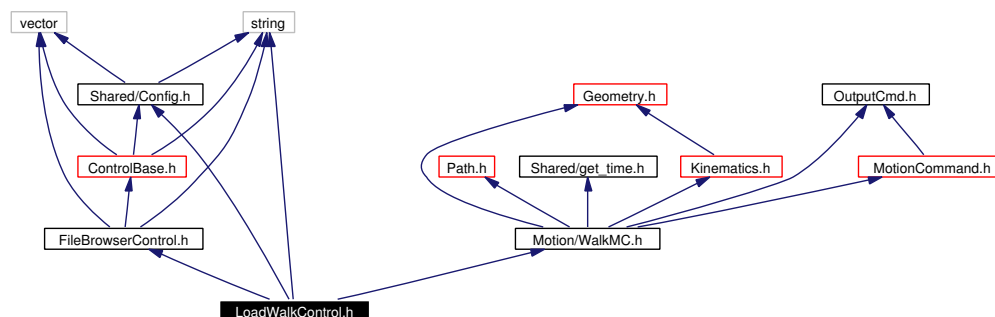
Date

2003/06/10 00:53:48

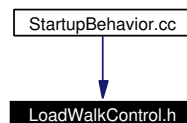
Definition in file [LoadWalkControl.h](#).

```
#include "FileBrowserControl.h"
#include "Motion/WalkMC.h"
#include "Shared/Config.h"
#include <string>
```

Include dependency graph for LoadWalkControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [LoadWalkControl](#)

When activated, loads a set of walk parameters from a file specified by user.

8.100 LockScope.h File Reference

8.100.1 Detailed Description

Defines [LockScope](#), which locks a [MutexLock](#) until the [LockScope](#) goes out of scope.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

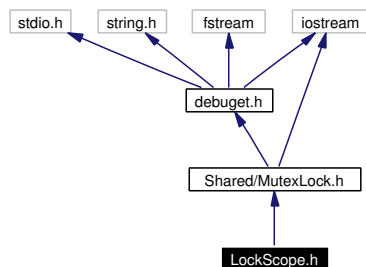
Date

2003/04/06 20:57:45

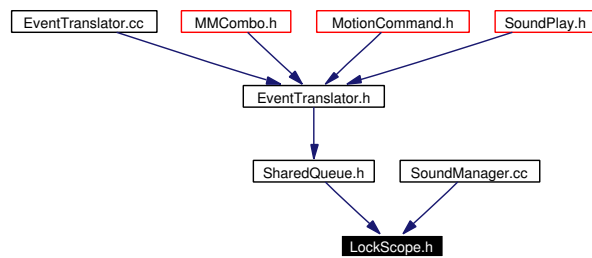
Definition in file [LockScope.h](#).

```
#include "Shared/MutexLock.h"
```

Include dependency graph for LockScope.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [LockScope](#)

Locks a [MutexLock](#) until the [LockScope](#) goes out of scope.

8.101 LocomotionEvent.h File Reference

8.101.1 Detailed Description

Defines [LocomotionEvent](#), which gives updates regarding the current movement of the robot through the world.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4_1

Revision

1.4

State

Rel

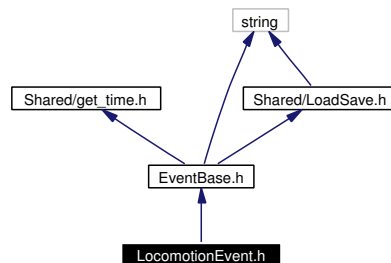
Date

2003/04/06 20:57:44

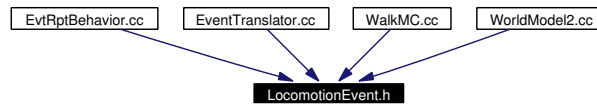
Definition in file [LocomotionEvent.h](#).

```
#include "EventBase.h"
```

Include dependency graph for LocomotionEvent.h:



This graph shows which files directly or indirectly include this file:



Compounds

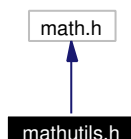
- class [LocomotionEvent](#)

Gives updates regarding the current movement of the robot through the world.

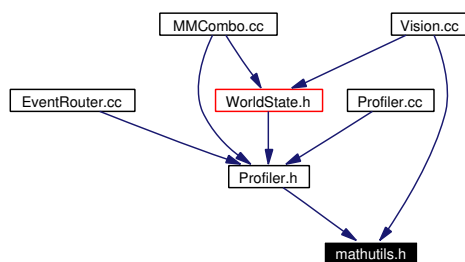
8.102 mathutils.h File Reference

```
#include <math.h>
```

Include dependency graph for mathutils.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [mathutils](#)

8.103 MCValueEditControl.h File Reference

8.103.1 Detailed Description

Defines [MCValueEditControl](#), which allows you to modify a value in memory, much like [ValueEditControl](#), but will check out a [MotionCommand](#) first to maintain proper mutual exclusion.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

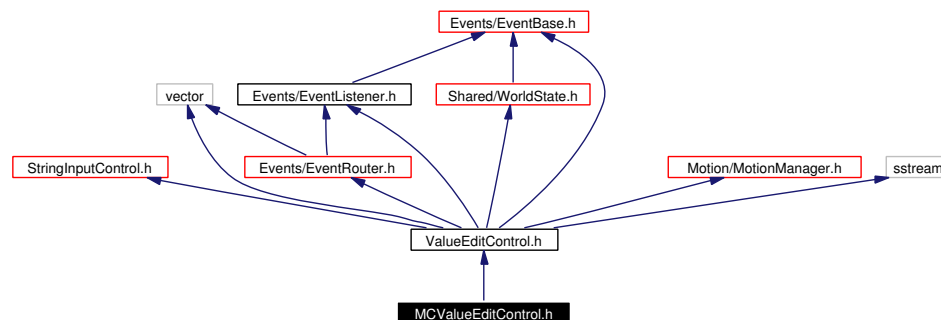
Date

2003/03/03 01:18:11

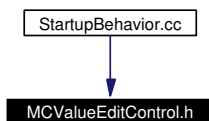
Definition in file [MCValueEditControl.h](#).

```
#include "ValueEditControl.h"
```

Include dependency graph for MCValueEditControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [MCValueEditControl](#)
allows you to modify a value in memory, much like [ValueEditControl](#), but will check out a [MotionCommand](#) first to maintain proper mutual exclusion.

8.104 MMAccessor.h File Reference

8.104.1 Detailed Description

Defines [MMAccessor](#), allows convenient ways to check MotionCommands in and out of the [MotionManager](#).

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.7

State

Rel

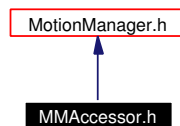
Date

2003/03/09 02:45:23

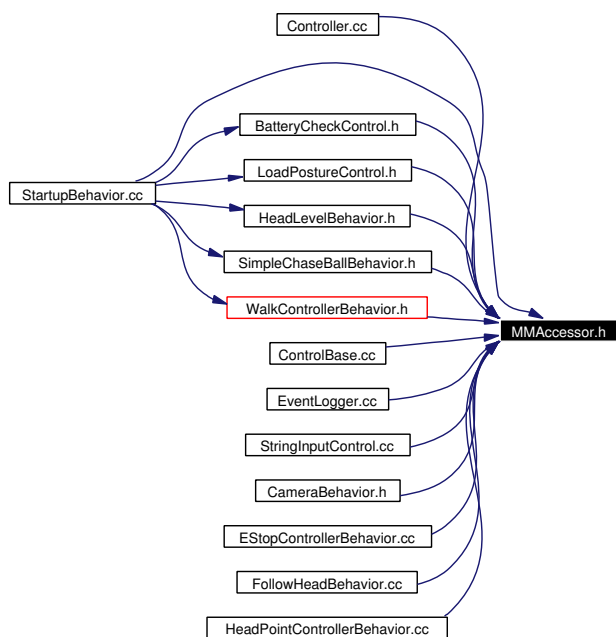
Definition in file [MMAccessor.h](#).

```
#include "MotionManager.h"
```

Include dependency graph for MMAccessor.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [MMAccessor](#)

This class allows convenient ways of accessing a motion command.

8.105 MMCombo.cc File Reference

8.105.1 Detailed Description

Implements [MMCombo](#), the [OObject](#) which "forks" (sort of) into Main and Motion processes.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.36

State

Rel

Date

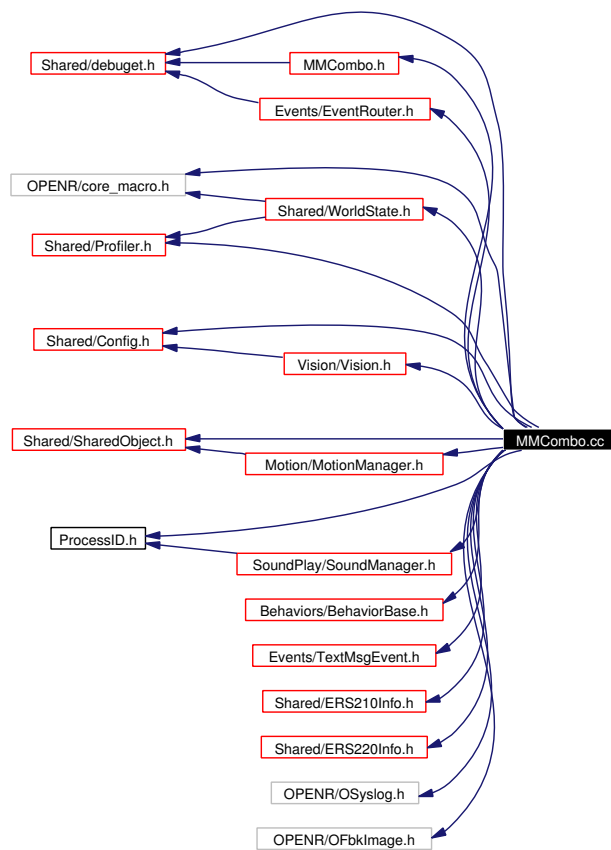
2003/06/12 18:06:11

Definition in file [MMCombo.cc](#).

```
#include "MMCombo.h"
#include "Shared/WorldState.h"
#include "Shared/Profiler.h"
#include "Shared/debuget.h"
#include "Shared/Config.h"
#include "Shared/SharedObject.h"
#include "Shared/ProcessID.h"
#include "Events/EventRouter.h"
#include "Behaviors/BehaviorBase.h"
#include "Motion/MotionManager.h"
#include "SoundPlay/SoundManager.h"
#include "Vision/Vision.h"
#include "Events/TextMsgEvent.h"
```

```
#include "Shared/ERS210Info.h"
#include "Shared/ERS220Info.h"
#include <OPENR/OSyslog.h>
#include <OPENR/core_macro.h>
#include <OPENR/OFbkImage.h>
```

Include dependency graph for MMCombo.cc:



Namespaces

- namespace [std](#)

Variables

- [BehaviorBase](#) & [startupBehavior](#)
used by [MMCombo](#) as the init behavior

8.105.2 Variable Documentation

8.105.2.1 [BehaviorBase](#) & [startupBehavior](#)

used by [MMCombo](#) as the init behavior

Definition at line 24 of file MMCombo.cc.

8.106 MMCombo.h File Reference

8.106.1 Detailed Description

Describes [MMCombo](#), the [OObject](#) which "forks" (sort of) into Main and Motion processes.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.20

State

Rel

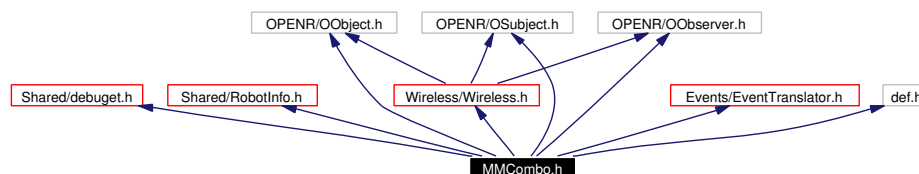
Date

2003/06/11 01:18:55

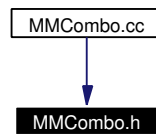
Definition in file [MMCombo.h](#).

```
#include "Shared/debuget.h"
#include "Shared/RobotInfo.h"
#include "Wireless/Wireless.h"
#include "Events/EventTranslator.h"
#include <OPENR/OObject.h>
#include <OPENR/OSubject.h>
#include <OPENR/OObserver.h>
#include "def.h"
```

Include dependency graph for MMCombo.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [MMCombo](#)
Contains code for both MainObj and MotoObj processes.

8.107 MotionCommand.cc File Reference

8.107.1 Detailed Description

Declares the static [MotionCommand::queue](#) variable, that's all.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

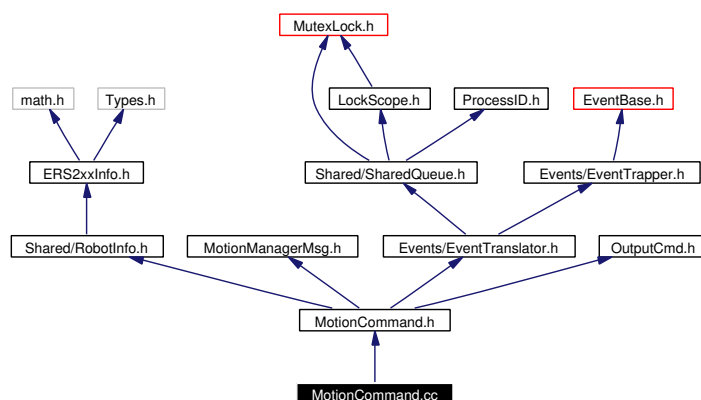
Date

2003/06/12 23:41:40

Definition in file [MotionCommand.cc](#).

```
#include "MotionCommand.h"
```

Include dependency graph for MotionCommand.cc:



8.108 MotionCommand.h File Reference

8.108.1 Detailed Description

Defines the [MotionCommand](#) class, used for creating motions of arbitrary complexity.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.15

State

Rel

Date

2003/04/09 03:33:56

Definition in file [MotionCommand.h](#).

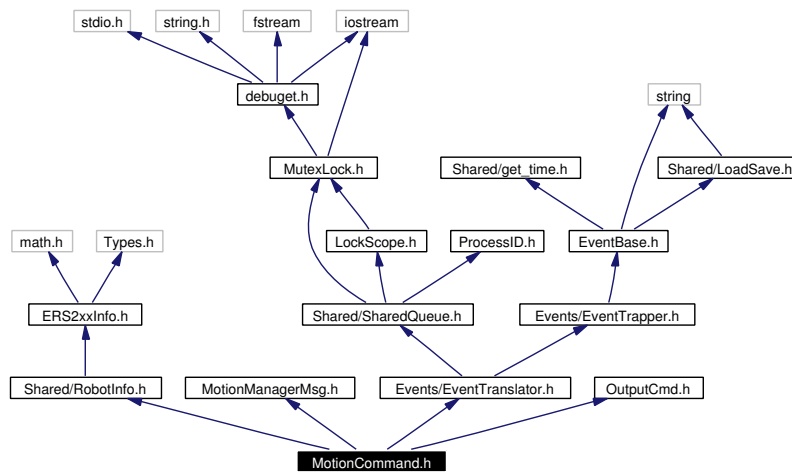
```
#include "Shared/RobotInfo.h"
```

```
#include "MotionManagerMsg.h"
```

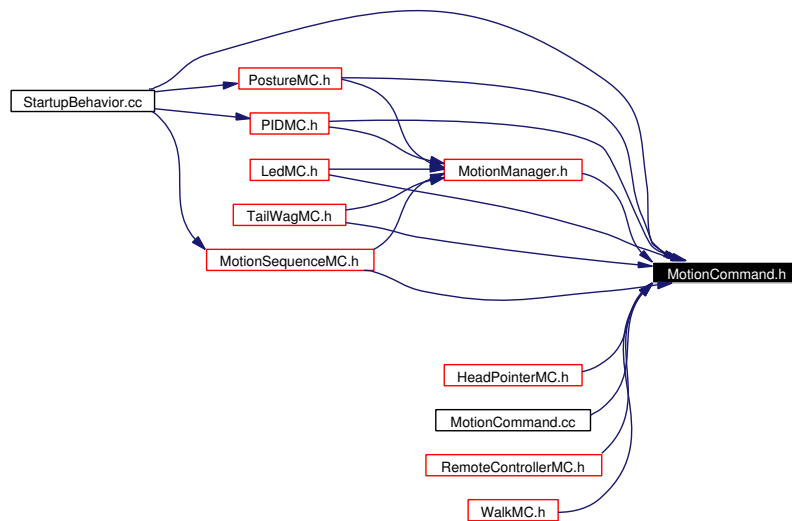
```
#include "Events/EventTranslator.h"
```

```
#include "OutputCmd.h"
```

Include dependency graph for MotionCommand.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [MotionCommand](#)

The abstract base class for motions, provides common interface. All motions should inherit from this.

8.109 MotionManager.cc File Reference

8.109.1 Detailed Description

Implements [MotionManager](#), simplifies sharing of MotionCommand's and provides mutual exclusion to their access.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.22

State

Rel

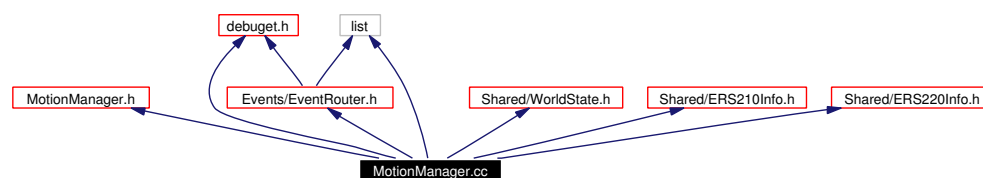
Date

2003/06/12 04:16:43

Definition in file [MotionManager.cc](#).

```
#include "MotionManager.h"
#include "Shared/debuget.h"
#include "Shared/WorldState.h"
#include "Events/EventRouter.h"
#include "Shared/ERS210Info.h"
#include "Shared/ERS220Info.h"
#include <list>
```

Include dependency graph for MotionManager.cc:



Typedefs

- typedef unsigned int [uint](#)
just for convenience

Variables

- [MotionManager](#) * [motman](#) = NULL
*anyone who #includes [MotionManager.h](#) will be wanting to use the global motman...
don't want multiple of these! created by MotoObj*

8.109.2 Typedef Documentation

8.109.2.1 typedef unsigned int [uint](#)

just for convenience

Definition at line 28 of file MotionManager.cc.

8.109.3 Variable Documentation

8.109.3.1 [MotionManager](#)* [motman](#) = NULL

anyone who #includes [MotionManager.h](#) will be wanting to use the global motman...
don't want multiple of these! created by MotoObj

Definition at line 11 of file MotionManager.cc.

8.110 MotionManager.h File Reference

8.110.1 Detailed Description

Describes [MotionManager](#), simplifies sharing of MotionCommand's and provides mutual exclusion to their access.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.13

State

Rel

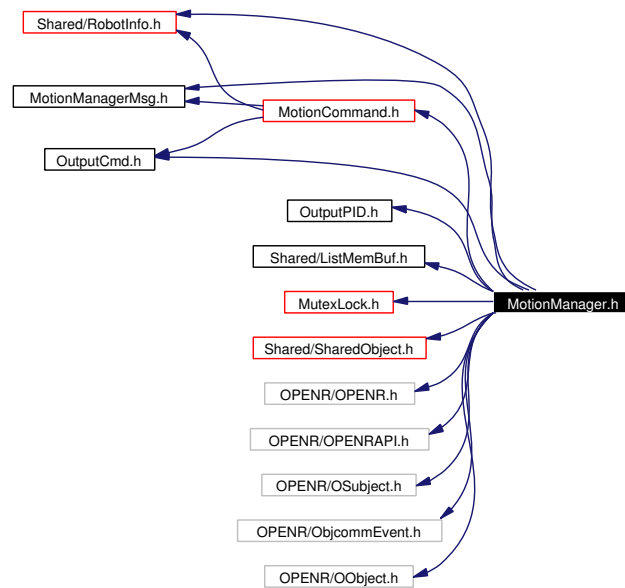
Date

2003/04/25 18:22:40

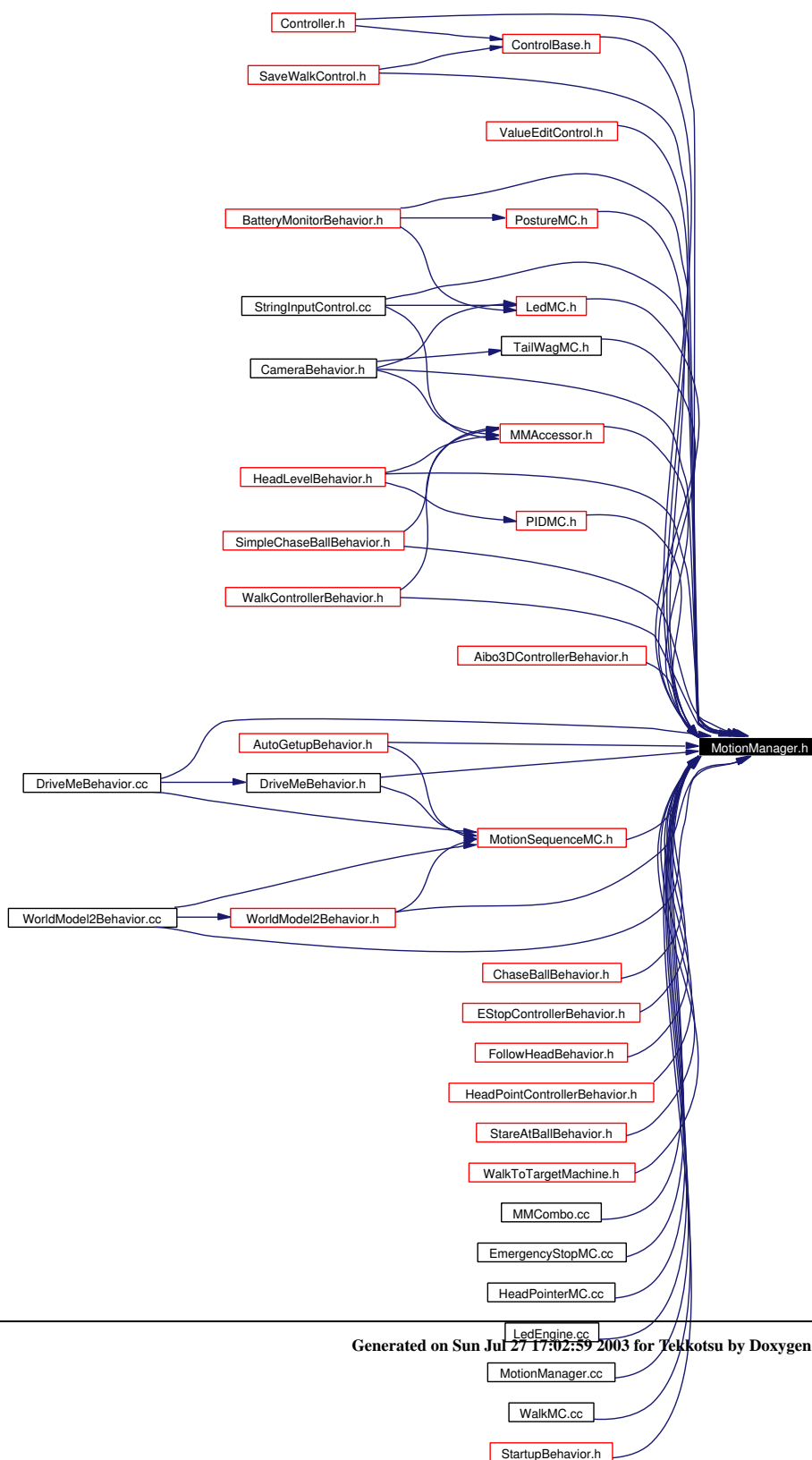
Definition in file [MotionManager.h](#).

```
#include "MotionCommand.h"
#include "OutputCmd.h"
#include "OutputPID.h"
#include "Shared/RobotInfo.h"
#include "Shared/ListMemBuf.h"
#include "Shared/MutexLock.h"
#include "Shared/SharedObject.h"
#include "MotionManagerMsg.h"
#include <OPENR/OPENR.h>
#include <OPENR/OPENRAPI.h>
#include <OPENR/OSubject.h>
#include <OPENR/ObjcommEvent.h>
#include <OPENR/OObject.h>
```

Include dependency graph for MotionManager.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [MotionManager](#)
The purpose of this class is to serialize access to the MotionCommands and simplify their sharing between memory spaces.
- struct [MotionManager.CommandEntry](#)
All the information we need to maintain about a [MotionCommand](#).
- class [MotionManager.OutputState](#)
holds the full requested value of an output
- struct [MotionManager.PIDUpdate](#)
used to request pids for a given joint

Variables

- [MotionManager](#) * [motman](#)
anyone who #includes [MotionManager.h](#) will be wanting to use the global motman... don't want multiple of these! created by MotoObj

8.110.2 Variable Documentation

8.110.2.1 [MotionManager](#)* [motman](#)

anyone who #includes [MotionManager.h](#) will be wanting to use the global motman... don't want multiple of these! created by MotoObj

Definition at line 228 of file MotionManager.h.

8.111 MotionManagerMsg.h File Reference

8.111.1 Detailed Description

Defines [MotionManagerMsg](#), a small header used by [MotionManager](#) for sending messages between processes.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

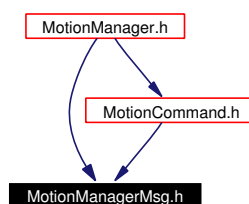
Rel

Date

2003/03/09 02:45:23

Definition in file [MotionManagerMsg.h](#).

This graph shows which files directly or indirectly include this file:



Compounds

- struct [MotionManagerMsg](#)

A small header that precedes data sent by [MotionManager](#) between processes.

8.112 motionReshapeKludge.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define `RESCALE_DX_A` 162.5885
- #define `RESCALE_DX_B` -32.0347
- #define `DX_MOT_THRESH` 55
- #define `RESCALE_DA_LEFT_A` $69.3368 * M_PI / 180$
- #define `RESCALE_DA_LEFT_B` $-28.3579 * M_PI / 180$
- #define `DA_LEFT_MOT_THRESH` 0.2
- #define `RESCALE_DA_RIGHT_A` $-68.4120 * M_PI / 180$
- #define `RESCALE_DA_RIGHT_B` $34.7600 * M_PI / 180$
- #define `DA_RIGHT_MOT_THRESH` -0.2
- #define `RESCALE_DX_A` 162.5885
- #define `RESCALE_DX_B` -32.0347
- #define `DX_MOT_THRESH` 55
- #define `RESCALE_DA_LEFT_A` $69.3368 * M_PI / 180$
- #define `RESCALE_DA_LEFT_B` $-28.3579 * M_PI / 180$
- #define `DA_LEFT_MOT_THRESH` 0.2
- #define `RESCALE_DA_RIGHT_A` $-68.4120 * M_PI / 180$
- #define `RESCALE_DA_RIGHT_B` $34.7600 * M_PI / 180$
- #define `DA_RIGHT_MOT_THRESH` -0.2

Functions

- `if (dx > DX_MOT_THRESH) dx=RESCALE_DX_A *(dx-DX_MOT_THRESH)/(180.0-DX_MOT_THRESH)`
- `else if (dx < -DX_MOT_THRESH) dx=RESCALE_DX_A *(dx+DX_MOT_THRESH)/(180.0-DX_MOT_THRESH)`
- `if (da > DA_LEFT_MOT_THRESH) da=RESCALE_DA_LEFT_A *(da-DA_LEFT_MOT_THRESH)/(1.008-DA_LEFT_MOT_THRESH)`

Variables

- else `dx` = 0.0
- else `da` = 0.0
- if((`dx`!=0.0)&&(da==0.0)) `time`=(`time` > 180?`time`-180 else if((`dx`==0.0)&&(da!=0.0)) `time`=(`time` > 100?`time`-100 els `time`) = (`time` > 140 ? `time` - 140 : `time` = 0)

8.112.1 Define Documentation

8.112.1.1 `#define DA_LEFT_MOT_THRESH 0.2`

Definition at line 64 of file motionReshapeKludge.h.

8.112.1.2 `#define DA_LEFT_MOT_THRESH 0.2`

Definition at line 64 of file motionReshapeKludge.h.

8.112.1.3 `#define DA_RIGHT_MOT_THRESH -0.2`

Definition at line 68 of file motionReshapeKludge.h.

8.112.1.4 `#define DA_RIGHT_MOT_THRESH -0.2`

Definition at line 68 of file motionReshapeKludge.h.

8.112.1.5 `#define DX_MOT_THRESH 55`

Definition at line 60 of file motionReshapeKludge.h.

8.112.1.6 `#define DX_MOT_THRESH 55`

Definition at line 60 of file motionReshapeKludge.h.

8.112.1.7 `#define RESCALE_DA_LEFT_A 69.3368*M_PI/180`

Definition at line 62 of file motionReshapeKludge.h.

8.112.1.8 #define RESCALE_DA_LEFT_A 69.3368*M_PI/180

Definition at line 62 of file motionReshapeKludge.h.

8.112.1.9 #define RESCALE_DA_LEFT_B -28.3579*M_PI/180

Definition at line 63 of file motionReshapeKludge.h.

8.112.1.10 #define RESCALE_DA_LEFT_B -28.3579*M_PI/180

Definition at line 63 of file motionReshapeKludge.h.

8.112.1.11 #define RESCALE_DA_RIGHT_A -68.4120*M_PI/180

Definition at line 66 of file motionReshapeKludge.h.

8.112.1.12 #define RESCALE_DA_RIGHT_A -68.4120*M_PI/180

Definition at line 66 of file motionReshapeKludge.h.

8.112.1.13 #define RESCALE_DA_RIGHT_B 34.7600*M_PI/180

Definition at line 67 of file motionReshapeKludge.h.

8.112.1.14 #define RESCALE_DA_RIGHT_B 34.7600*M_PI/180

Definition at line 67 of file motionReshapeKludge.h.

8.112.1.15 #define RESCALE_DX_A 162.5885

Definition at line 58 of file motionReshapeKludge.h.

8.112.1.16 #define RESCALE_DX_A 162.5885

Definition at line 58 of file motionReshapeKludge.h.

8.112.1.17 #define RESCALE_DX_B -32.0347

Definition at line 59 of file motionReshapeKludge.h.

8.112.1.18 #define RESCALE_DX_B -32.0347

Definition at line 59 of file motionReshapeKludge.h.

8.112.2 Function Documentation**8.112.2.1 if (da, DA_LEFT_MOT_THRESH)****8.112.2.2 else if ()****8.112.2.3 if (dx, DX_MOT_THRESH)****8.112.3 Variable Documentation****8.112.3.1 else da = 0.0**

Definition at line 83 of file motionReshapeKludge.h.

8.112.3.2 else dx = 0.0

Definition at line 75 of file motionReshapeKludge.h.

8.112.3.3 if ((dx != 0.0) && (da == 0.0)) time = (time > 180 ? time - 180 else if ((dx == 0.0) && (da != 0.0)) time = (time > 100 ? time - 100 els time) = (time > 140 ? time - 140 : time = 0)

Definition at line 101 of file motionReshapeKludge.h.

8.113 MotionSequenceMC.cc File Reference

8.113.1 Detailed Description

Implements [MotionSequence](#), handy little (or not so little) class for switching between a sequence of postures.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.8

State

Rel

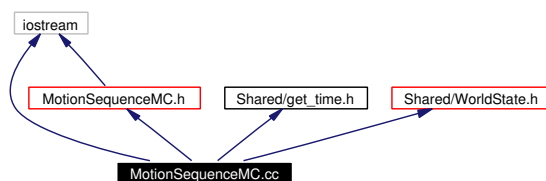
Date

2003/06/12 04:16:43

Definition in file [MotionSequenceMC.cc](#).

```
#include "MotionSequenceMC.h"  
#include "Shared/get_time.h"  
#include "Shared/WorldState.h"  
#include <iostream>
```

Include dependency graph for MotionSequenceMC.cc:



8.114 MotionSequenceMC.h File Reference

8.114.1 Detailed Description

Describes [MotionSequence](#) and defines [MotionSequenceMC](#), handy little (or not so little) classes for switching between a sequence of postures.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.9

State

Rel

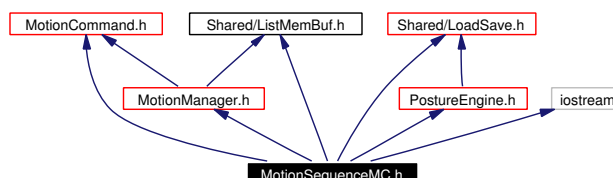
Date

2003/06/10 04:41:41

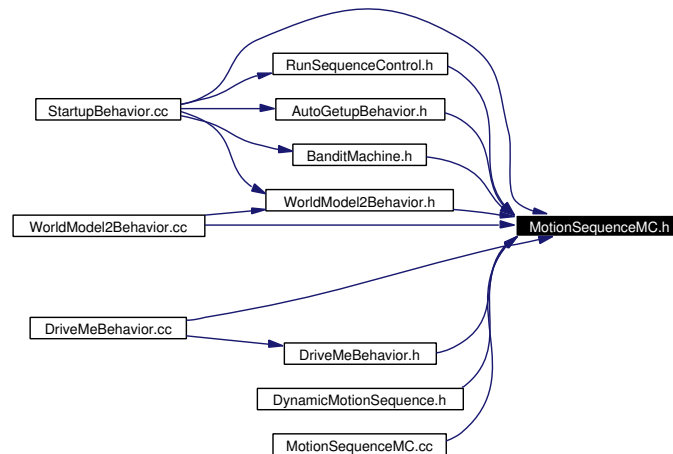
Definition in file [MotionSequenceMC.h](#).

```
#include "MotionCommand.h"
#include "MotionManager.h"
#include "Shared/LoadSave.h"
#include "Shared/ListMemBuf.h"
#include "PostureEngine.h"
#include <iostream>
```

Include dependency graph for MotionSequenceMC.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [MotionSequence](#)
A handy little (or not so little) class for switching between a sequence of postures.
- struct [MotionSequence.Move](#)
This struct holds all the information needed about a frame for a particular output.
- class [MotionSequenceMC](#)
Instantiates MotionSequences - when you want to make a new [MotionSequence](#), make one of these.

8.115 MutexLock.h File Reference

8.115.1 Detailed Description

Defines [MutexLock](#), a software only mutual exclusion lock.

Author:

ejt (Creator), Edward A. Lycklama, Vassos Hadzilacos (paper from which this was based)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.7

State

Rel

Date

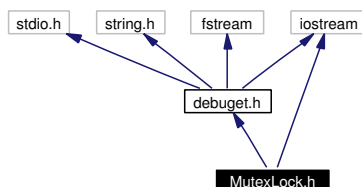
2003/06/12 23:41:40

Definition in file [MutexLock.h](#).

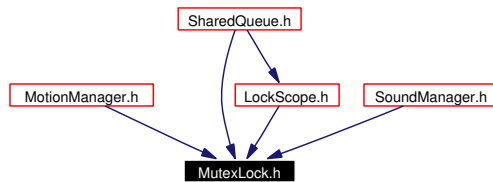
```
#include "debuget.h"
```

```
#include <iostream>
```

Include dependency graph for MutexLock.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [MutexLock](#)
A software only mutual exclusion lock.
- struct [MutexLock.door_t](#)
Holds per process shared info, one of these per process.

Defines

- `#define mutexdebugout(i, c) {}`
If you define this to do something more interesting, can use it to see what's going on in the locking process.

8.115.2 Define Documentation

8.115.2.1 `#define mutexdebugout(i, c) {}`

If you define this to do something more interesting, can use it to see what's going on in the locking process.

Definition at line 160 of file `MutexLock.h`.

8.116 NullControl.h File Reference

8.116.1 Detailed Description

Defines [NullControl](#), which does absolutely nothing (handy for fake items in a menu).

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.1

State

Rel

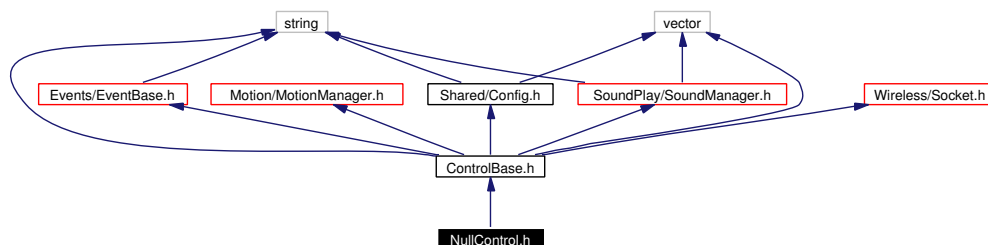
Date

2003/06/09 08:05:12

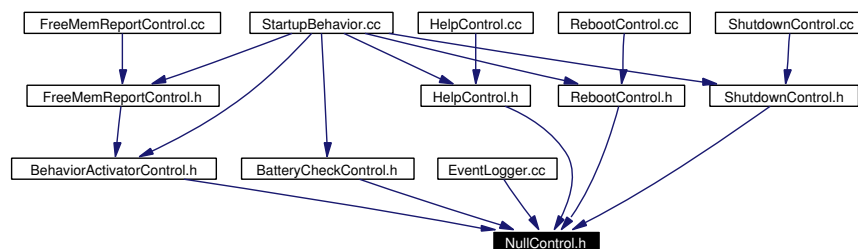
Definition in file [NullControl.h](#).

```
#include "ControlBase.h"
```

Include dependency graph for NullControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [NullControl](#)

when activated, this will return immediately (handy for fake items in a menu)

8.117 OutputCmd.h File Reference

8.117.1 Detailed Description

Describes [OutputCmd](#), holds information needed to control a single output.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

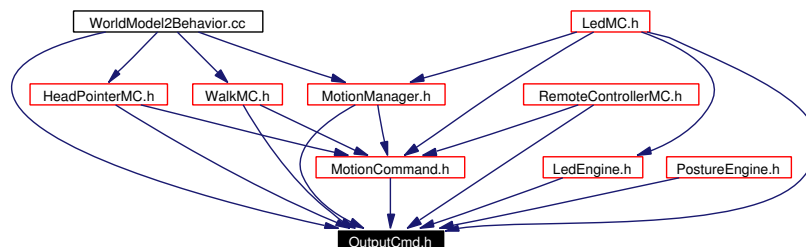
Rel

Date

2003/04/09 07:01:30

Definition in file [OutputCmd.h](#).

This graph shows which files directly or indirectly include this file:



Compounds

- class [OutputCmd](#)

This object holds all the information needed to control a single output.

8.118 OutputNode.h File Reference

8.118.1 Detailed Description

Defines [OutputNode](#), a very simple [StateNode](#) that outputs its name to a given ostream upon activation, handy for debugging.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.2

State

Rel

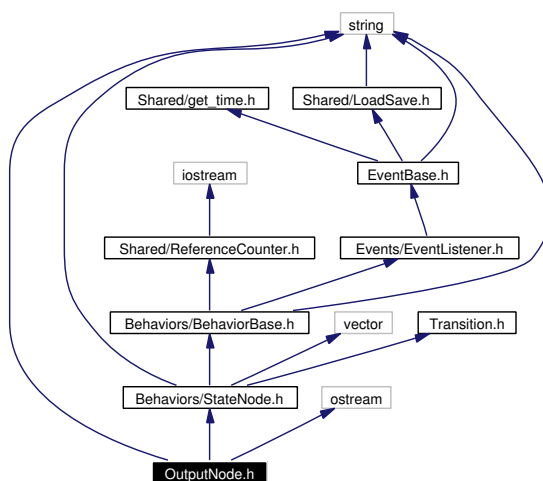
Date

2003/03/09 02:45:22

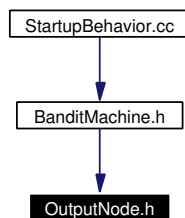
Definition in file [OutputNode.h](#).

```
#include "Behaviors/StateNode.h"
#include <string>
#include <ostream>
```

Include dependency graph for OutputNode.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [OutputNode](#)

A very simple [StateNode](#) that outputs its name to a given ostream upon activation, handy for debugging.

8.119 OutputPID.h File Reference

8.119.1 Detailed Description

Describes [OutputPID](#), holds information needed to control a single output.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

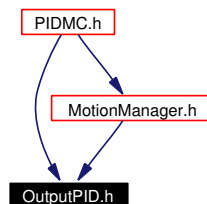
Rel

Date

2003/04/09 07:01:30

Definition in file [OutputPID.h](#).

This graph shows which files directly or indirectly include this file:



Compounds

- class [OutputPID](#)

This object holds all the information needed to control a single output.

8.120 Path.h File Reference

8.120.1 Detailed Description

Performs calculations regarding splines for path execution.

Author:

James R. Bruce (Creator)

```
=====
path.h : Template for building spline paths
-----
Copyright (C) 1999-2002 James R. Bruce
School of Computer Science, Carnegie Mellon University
-----

This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
=====
*
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.4

State

Rel

Date

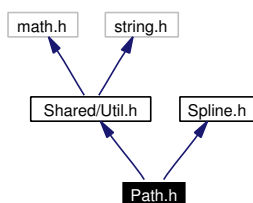
2003/01/23 18:14:04

Definition in file [Path.h](#).

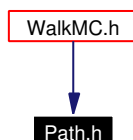
```
#include "Shared/Util.h"
```

```
#include "Spline.h"
```

Include dependency graph for Path.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [SplinePath](#)

Defines

- `#define SPATH SplinePath<point,fnum>`
- `#define SPATH_TEM template <class point,class fnum>`

8.120.2 Define Documentation

8.120.2.1 `#define SPATH SplinePath<point,fnum>`

Definition at line 22 of file Path.h.

8.120.2.2 `#define SPATH_TEM template <class point,class fnum>`

Definition at line 23 of file Path.h.

8.121 PIDMC.h File Reference

8.121.1 Detailed Description

Defines [PIDMC](#), a nice little [MotionCommand](#) for manually manipulating the PID values.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.9

State

Rel

Date

2003/06/12 23:41:40

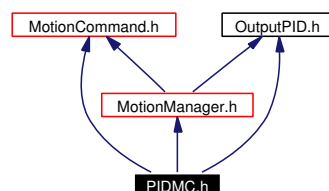
Definition in file [PIDMC.h](#).

```
#include "MotionCommand.h"
```

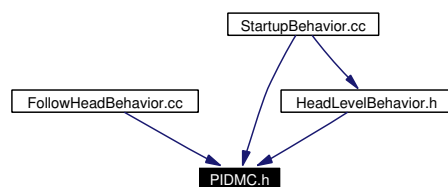
```
#include "MotionManager.h"
```

```
#include "OutputPID.h"
```

Include dependency graph for PIDMC.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [PIDMC](#)

A nice little [MotionCommand](#) for manually manipulating the PID values.

8.122 PlaySoundControl.h File Reference

8.122.1 Detailed Description

Defines [PlaySoundControl](#), which when activated, plays a sound selected from the memory stick.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.5

State

Rel

Date

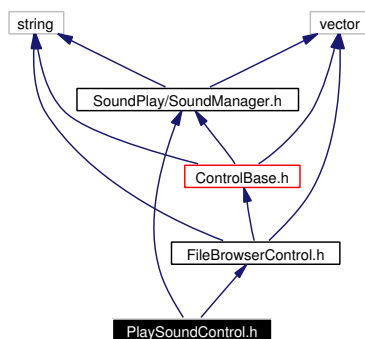
2003/06/10 00:53:48

Definition in file [PlaySoundControl.h](#).

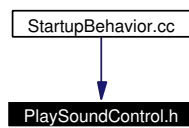
```
#include "FileBrowserControl.h"
```

```
#include "SoundPlay/SoundManager.h"
```

Include dependency graph for PlaySoundControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

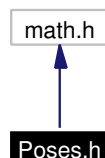
- class [PlaySoundControl](#)

Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...).

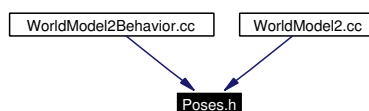
8.123 Poses.h File Reference

```
#include <math.h>
```

Include dependency graph for Poses.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define [SP_TOLERANCE](#) $5 * 180 / M_PI$
- #define [SP_LFR_JOINT](#) -0.2000
- #define [SP_LFR_SHLDR](#) 0.0600
- #define [SP_LFR_KNEE](#) 0.7500
- #define [SP_RFR_JOINT](#) -0.2000
- #define [SP_RFR_SHLDR](#) 0.0600
- #define [SP_RFR_KNEE](#) 0.7500
- #define [SP_LBK_JOINT](#) -0.4000
- #define [SP_LBK_SHLDR](#) 0.0600
- #define [SP_LBK_KNEE](#) 0.8500
- #define [SP_RBK_JOINT](#) -0.4000
- #define [SP_RBK_SHLDR](#) 0.0600
- #define [SP_RBK_KNEE](#) 0.8500
- #define [DA_TOLERANCE](#) $1.5 * 180 / M_PI$
- #define [DA_TILT](#) 0.0
- #define [DA_PAN](#) 0.0
- #define [DA_ROLL](#) 0.0

8.123.1 Define Documentation

8.123.1.1 `#define DA_PAN 0.0`

Definition at line 29 of file Poses.h.

8.123.1.2 `#define DA_ROLL 0.0`

Definition at line 30 of file Poses.h.

8.123.1.3 `#define DA_TILT 0.0`

Definition at line 28 of file Poses.h.

8.123.1.4 `#define DA_TOLERANCE 1.5*180/M_PI`

Definition at line 25 of file Poses.h.

8.123.1.5 `#define SP_LBK_JOINT -0.4000`

Definition at line 16 of file Poses.h.

8.123.1.6 `#define SP_LBK_KNEE 0.8500`

Definition at line 18 of file Poses.h.

8.123.1.7 `#define SP_LBK_SHLDR 0.0600`

Definition at line 17 of file Poses.h.

8.123.1.8 `#define SP_LFR_JOINT -0.2000`

Definition at line 10 of file Poses.h.

8.123.1.9 `#define SP_LFR_KNEE 0.7500`

Definition at line 12 of file Poses.h.

8.123.1.10 #define SP_LFR_SHLDR 0.0600

Definition at line 11 of file Poses.h.

8.123.1.11 #define SP_RBK_JOINT -0.4000

Definition at line 19 of file Poses.h.

8.123.1.12 #define SP_RBK_KNEE 0.8500

Definition at line 21 of file Poses.h.

8.123.1.13 #define SP_RBK_SHLDR 0.0600

Definition at line 20 of file Poses.h.

8.123.1.14 #define SP_RFR_JOINT -0.2000

Definition at line 13 of file Poses.h.

8.123.1.15 #define SP_RFR_KNEE 0.7500

Definition at line 15 of file Poses.h.

8.123.1.16 #define SP_RFR_SHLDR 0.0600

Definition at line 14 of file Poses.h.

8.123.1.17 #define SP_TOLERANCE 5*180/M_PI

Definition at line 7 of file Poses.h.

8.124 PostureEngine.cc File Reference

8.124.1 Detailed Description

Implements [PostureEngine](#), a base class for managing the values and weights of all the outputs.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.8

State

Rel

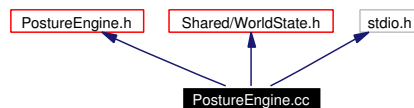
Date

2003/06/12 04:16:43

Definition in file [PostureEngine.cc](#).

```
#include "PostureEngine.h"  
#include "Shared/WorldState.h"  
#include <stdio.h>
```

Include dependency graph for PostureEngine.cc:



8.125 PostureEngine.h File Reference

8.125.1 Detailed Description

Describes [PostureEngine](#), a base class for managing the values and weights of all the outputs.

Todo

write a binary version of Load/Save commands for faster access

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

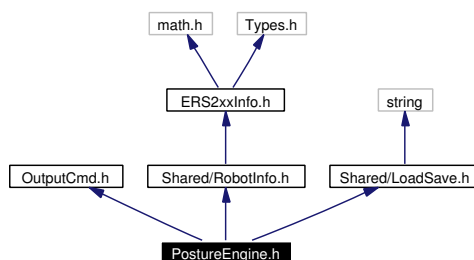
Date

2003/03/04 05:46:07

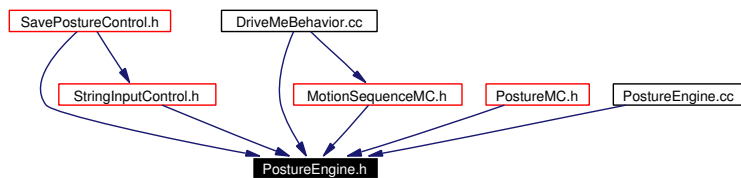
Definition in file [PostureEngine.h](#).

```
#include "OutputCmd.h"
#include "Shared/RobotInfo.h"
#include "Shared/LoadSave.h"
```

Include dependency graph for PostureEngine.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [PostureEngine](#)

A class for storing a set of positions and weights for all the outputs.

8.126 PostureMC.h File Reference

8.126.1 Detailed Description

Defines [PostureMC](#), a [MotionCommand](#) shell for [PostureEngine](#).

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

Date

2003/03/04 05:46:07

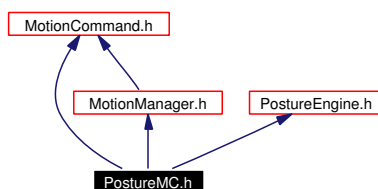
Definition in file [PostureMC.h](#).

```
#include "MotionCommand.h"
```

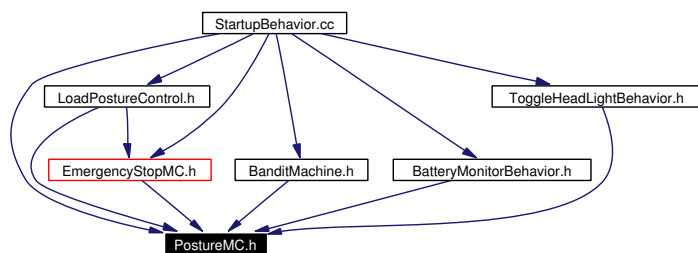
```
#include "PostureEngine.h"
```

```
#include "MotionManager.h"
```

Include dependency graph for PostureMC.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [PostureMC](#)
a [MotionCommand](#) shell for [PostureEngine](#)

8.127 ProcessID.cc File Reference

8.127.1 Detailed Description

Declares the static [ProcessID::ID](#), that's all.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

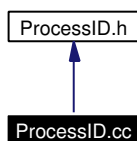
Date

2003/06/12 23:41:41

Definition in file [ProcessID.cc](#).

```
#include "ProcessID.h"
```

Include dependency graph for ProcessID.cc:



8.128 ProcessID.h File Reference

8.128.1 Detailed Description

Defines [ProcessID](#) - simple little global for checking which process is currently running, kind of. (see [ProcessID::getID\(\)](#)).

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

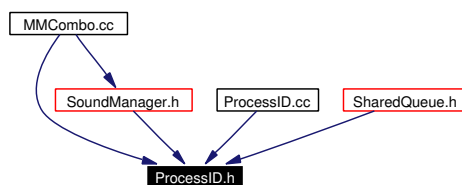
Rel

Date

2003/04/06 20:57:45

Definition in file [ProcessID.h](#).

This graph shows which files directly or indirectly include this file:



Compounds

- class [ProcessID](#)

this is a class instead of a namespace so i can limit write access of the ID value to the OObjects

8.129 Profiler.cc File Reference

8.129.1 Detailed Description

Implements [Profiler](#), which manages a hierarchy of timers for profiling time spent in code.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.9

State

Rel

Date

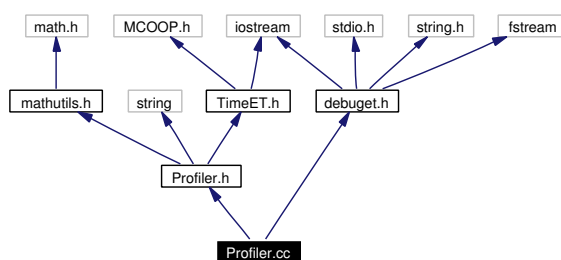
2003/04/09 01:12:24

Definition in file [Profiler.cc](#).

```
#include "Profiler.h"
```

```
#include "debuget.h"
```

Include dependency graph for Profiler.cc:



8.130 Profiler.h File Reference

8.130.1 Detailed Description

Describes [Profiler](#), which manages a hierarchy of timers for profiling time spent in code.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.8

State

Rel

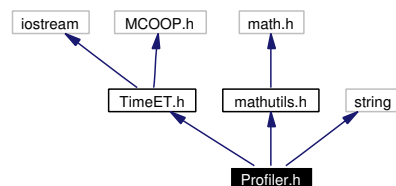
Date

2003/04/09 07:01:30

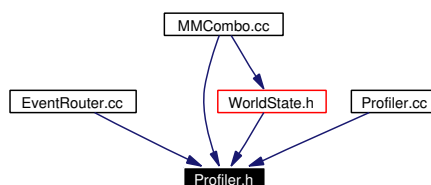
Definition in file [Profiler.h](#).

```
#include "TimeET.h"  
#include "mathutils.h"  
#include <string>
```

Include dependency graph for Profiler.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [Profiler](#)
Managers a hierarchy of timers for profiling time spent in code, gives microsecond resolution.
- struct [Profiler.SectionInfo](#)
holds all the information needed for book keeping for each timer
- class [Profiler.Timer](#)
Measures the time that this class exists, reports result to a profiler.
- class [ProfilerOfSize](#)
templated subclass allows compile-time flexibility of how much memory to use.

Defines

- `#define` [PROFSECTION](#)(NAME, PROF)
put this at the beginning of any function for which you wish to collect profiling information

8.130.2 Define Documentation

8.130.2.1 `#define` PROFSECTION(NAME, PROF)

Value:

```
static unsigned int _PROFSECTION_id=PROF.getNewID(NAME);\nProfiler::Timer timer(_PROFSECTION_id,&PROF);
```

put this at the beginning of any function for which you wish to collect profiling information

Uses a variable named `_PROFSECTION_id` to store a static ID number - don't redefine or modify that...

Parameters:

NAME the name of this section for reporting

PROF the actual profiler to use

Definition at line 14 of file Profiler.h.

8.131 ProfilerCheckControl.h File Reference

8.131.1 Detailed Description

Defines [ProfilerCheckControl](#), which causes the [WorldState::mainProfile](#) and [WorldState::motionProfile](#) to display reports to cout.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.3

State

Rel

Date

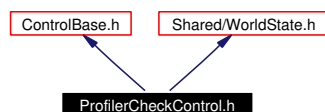
2003/06/10 00:53:48

Definition in file [ProfilerCheckControl.h](#).

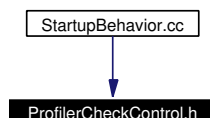
```
#include "ControlBase.h"
```

```
#include "Shared/WorldState.h"
```

Include dependency graph for ProfilerCheckControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [ProfilerCheckControl](#)
causes the [WorldState::mainProfile](#) and [WorldState::motionProfile](#) to display reports to cout

8.132 RebootControl.cc File Reference

8.132.1 Detailed Description

Implements [RebootControl](#), which causes the aibo to reboot (very short - one function separated out to limit recompile of the OPENR headers).

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

Date

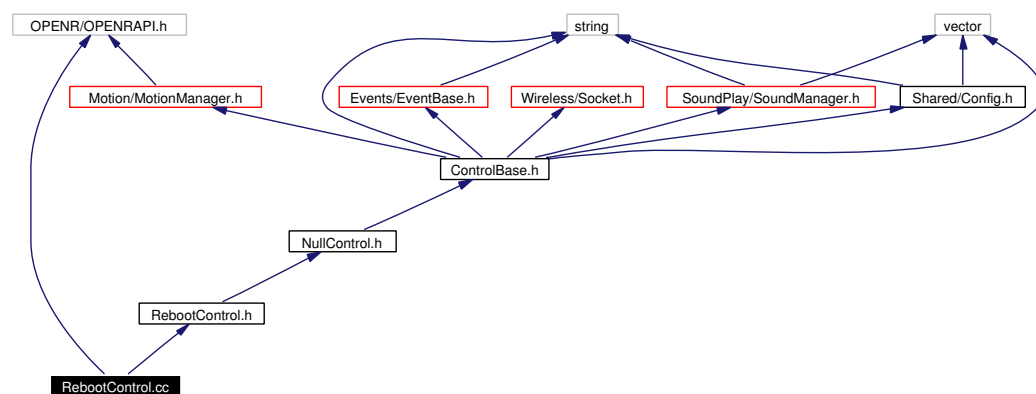
2003/06/12 23:41:36

Definition in file [RebootControl.cc](#).

```
#include "RebootControl.h"
```

```
#include <OPENR/OPENRAPI.h>
```

Include dependency graph for RebootControl.cc:



8.133 RebootControl.h File Reference

8.133.1 Detailed Description

Defines [RebootControl](#), which causes the aibo to reboot.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.1

State

Rel

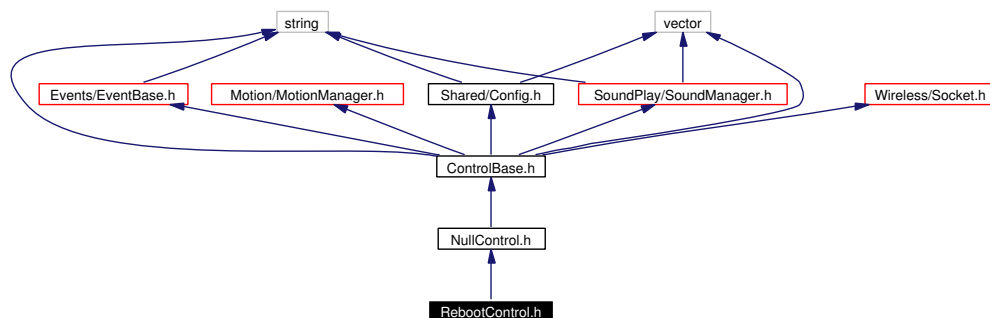
Date

2003/06/09 08:05:13

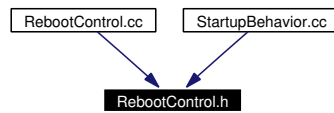
Definition in file [RebootControl.h](#).

```
#include "NullControl.h"
```

Include dependency graph for RebootControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [RebootControl](#)
when activated, this will cause the aibo to reboot

8.134 ReferenceCounter.h File Reference

8.134.1 Detailed Description

Defines the [ReferenceCounter](#) base class, which allows for automatic memory deallocation.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

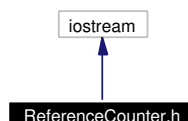
Date

2003/06/12 18:06:11

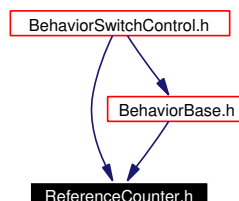
Definition in file [ReferenceCounter.h](#).

```
#include <iostream>
```

Include dependency graph for ReferenceCounter.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [ReferenceCounter](#)

Performs simple reference counting, will delete the object when removing the last reference.

8.135 RemoteControllerMC.h File Reference

8.135.1 Detailed Description

Describes [RemoteControllerMC](#), a class for various ways to control where the head is looking.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

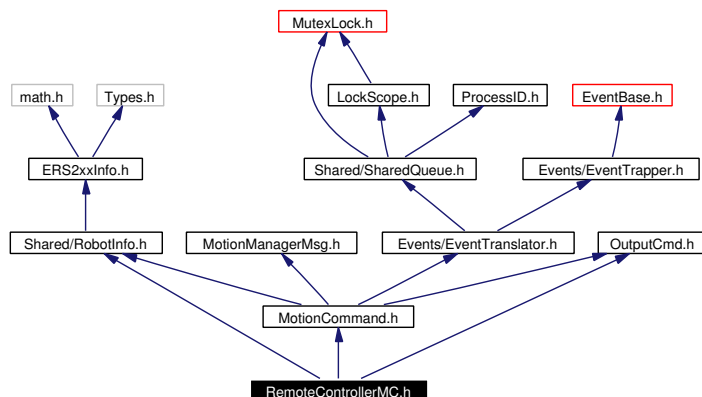
Date

2003/06/12 18:06:11

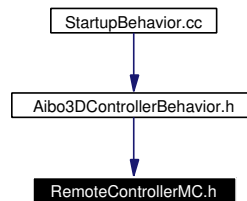
Definition in file [RemoteControllerMC.h](#).

```
#include "MotionCommand.h"
#include "OutputCmd.h"
#include "Shared/RobotInfo.h"
```

Include dependency graph for RemoteControllerMC.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [RemoteControllerMC](#)

This class is used for setting all `PIDJoints` to a certain set of values (not the gains, just the joint positions).

8.136 RemoteProcess.cc File Reference

8.136.1 Detailed Description

Describes [RemoteProcess](#), sample RemoteProcessingOPENR process.

Author:

alokl (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

Date

2003/06/12 23:41:40

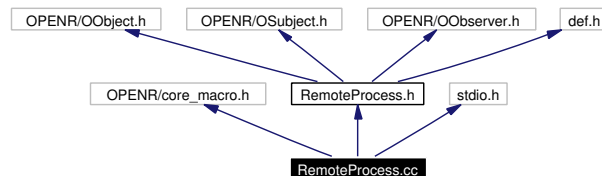
Definition in file [RemoteProcess.cc](#).

```
#include <OPENR/core_macro.h>
```

```
#include "RemoteProcess.h"
```

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

Include dependency graph for RemoteProcess.cc:



8.137 RemoteProcess.h File Reference

8.137.1 Detailed Description

Describes [RemoteProcess](#), sample RemoteProcessingOPENR process.

Author:

alokl (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.6

State

Rel

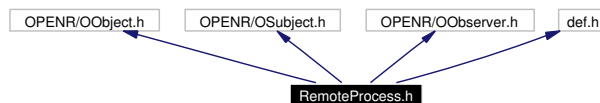
Date

2003/06/12 23:41:40

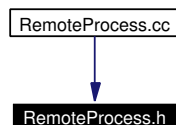
Definition in file [RemoteProcess.h](#).

```
#include <OPENR/OObject.h>
#include <OPENR/OSubject.h>
#include <OPENR/OObserver.h>
#include "def.h"
```

Include dependency graph for RemoteProcess.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [RemoteProcess](#)

Sample RemoteProcessingOPENR process.

8.138 RobotInfo.h File Reference

8.138.1 Detailed Description

Checks the define's to load the appropriate header and namespace.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.12

State

Rel

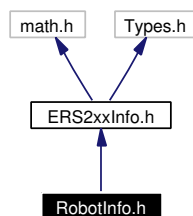
Date

2003/06/11 01:18:56

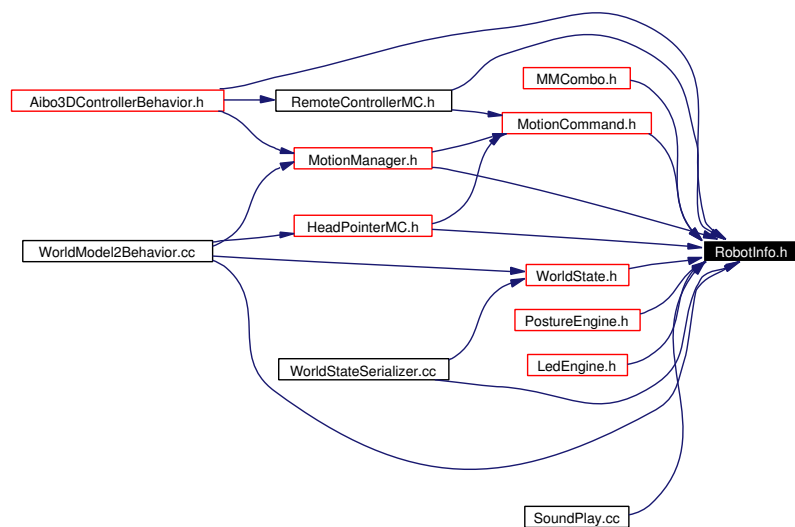
Definition in file [RobotInfo.h](#).

```
#include "ERS2xxInfo.h"
```

Include dependency graph for RobotInfo.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [RobotInfo](#)

8.139 RunSequenceControl.h File Reference

8.139.1 Detailed Description

Defines [RunSequenceControl](#), which when activated, loads and runs a motion sequence from a file name read from cin (stored in ms/data/motion).

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.9

State

Rel

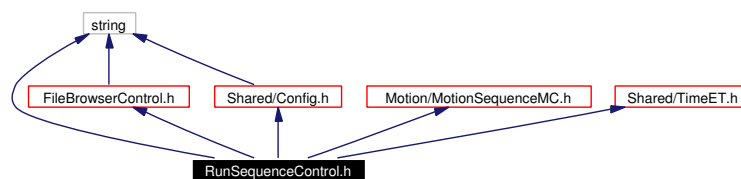
Date

2003/06/12 18:06:10

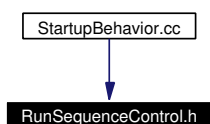
Definition in file [RunSequenceControl.h](#).

```
#include "FileBrowserControl.h"
#include "Motion/MotionSequenceMC.h"
#include "Shared/TimeET.h"
#include "Shared/Config.h"
#include <string>
```

Include dependency graph for RunSequenceControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [RunSequenceControl](#)

Upon activation, loads a position from a file name read from cin (stored in ms/data/motion...).

8.140 SavePostureControl.h File Reference

8.140.1 Detailed Description

Defines [SavePostureControl](#), which when activated, saves the current position to a file name read from user (stored in /ms/data/motion/...).

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.6

State

Rel

Date

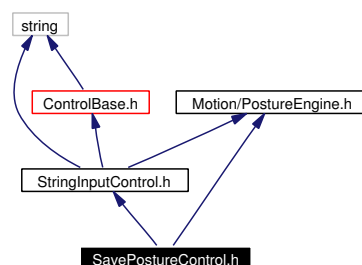
2003/06/12 18:06:10

Definition in file [SavePostureControl.h](#).

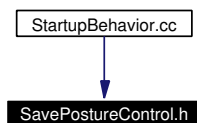
```
#include "StringInputControl.h"
```

```
#include "Motion/PostureEngine.h"
```

Include dependency graph for SavePostureControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [SavePostureControl](#)

Upon activation, saves the current position to a file name read from user (stored in /ms/data/motion/...).

8.141 SaveWalkControl.h File Reference

8.141.1 Detailed Description

Defines [SaveWalkControl](#), which when activated, saves walk parameters to a file specified from cin.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.4

State

Rel

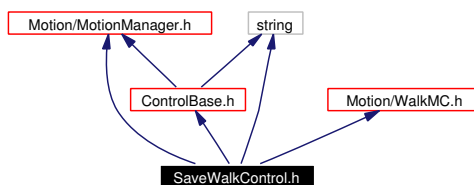
Date

2003/06/07 22:03:53

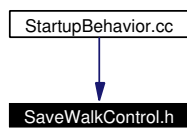
Definition in file [SaveWalkControl.h](#).

```
#include "ControlBase.h"
#include "Motion/MotionManager.h"
#include "Motion/WalkMC.h"
#include <string>
```

Include dependency graph for SaveWalkControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [SaveWalkControl](#)

When activated, saves walk parameters to a file specified from cin.

8.142 Serializer.h File Reference

8.142.1 Detailed Description

Defines the [Serializer](#) base class, which provides a default serializer for simple objects.

Author:

alokl (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.4

State

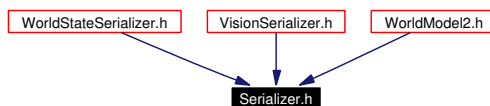
Rel

Date

2003/06/12 18:06:11

Definition in file [Serializer.h](#).

This graph shows which files directly or indirectly include this file:



Compounds

- class [Serializer](#)
provides a default serializer base class for simple objects

8.143 SharedObject.h File Reference

8.143.1 Detailed Description

Defines [SharedObject](#), a wrapper for objects in order to facilitate sending them between processes.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.1

State

Rel

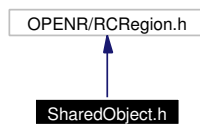
Date

2003/03/09 07:06:11

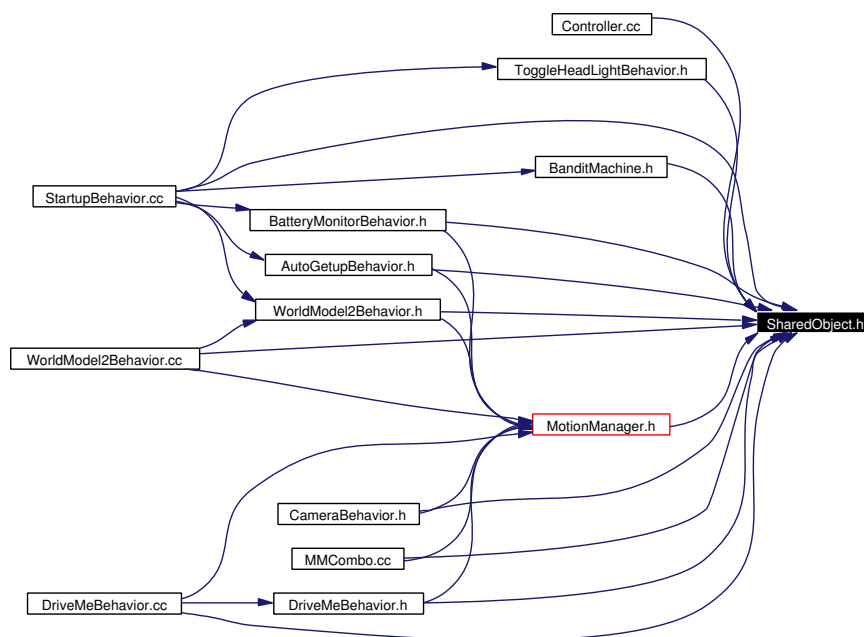
Definition in file [SharedObject.h](#).

```
#include <OPENR/RCRegion.h>
```

Include dependency graph for SharedObject.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [SharedObject](#)

This templated class allows convenient creation of any type of class wrapped in a shared memory region.

- class [SharedObjectBase](#)

It's nice to have a parent class of [SharedObject](#) (which is what you probably want to be reading) so that you can pass around the data structure without worrying about what type is inside the shared memory region.

8.144 SharedQueue.h File Reference

8.144.1 Detailed Description

Defines [SharedQueue](#), a shared memory message buffer for interprocess communication.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

Date

2003/06/12 23:41:41

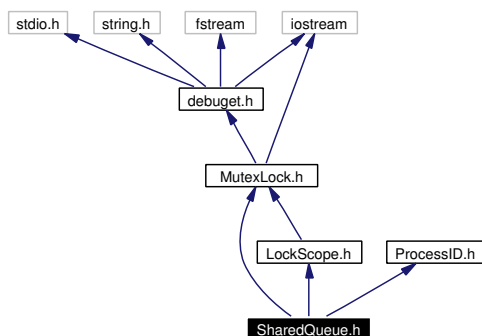
Definition in file [SharedQueue.h](#).

```
#include "MutexLock.h"
```

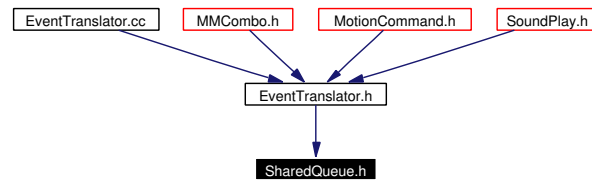
```
#include "ProcessID.h"
```

```
#include "LockScope.h"
```

Include dependency graph for SharedQueue.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [SharedQueue](#)
SharedQueue is a shared memory message buffer for interprocess communication.
- struct [SharedQueue.entry_t](#)
entry information

8.145 ShutdownControl.cc File Reference

8.145.1 Detailed Description

Implements [ShutdownControl](#), which causes the aibo to shutdown (very short - one function separated out to limit recompile of the OPENR headers).

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

Date

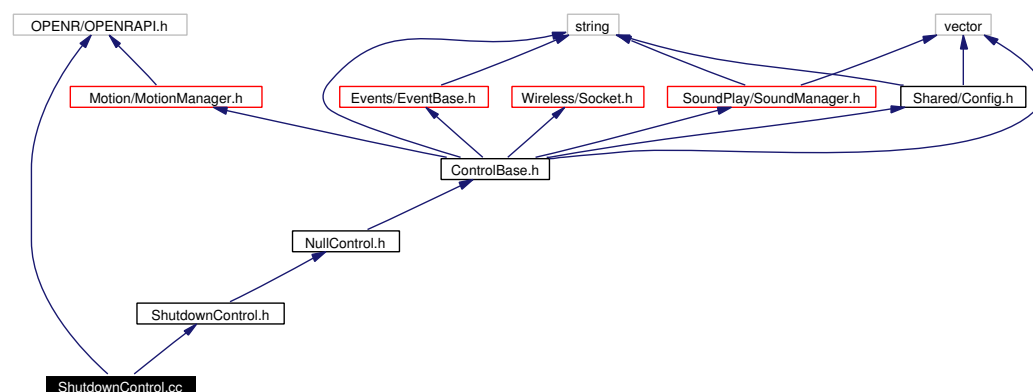
2003/06/12 23:41:36

Definition in file [ShutdownControl.cc](#).

```
#include "ShutdownControl.h"
```

```
#include <OPENR/OPENRAPI.h>
```

Include dependency graph for ShutdownControl.cc:



8.146 ShutdownControl.h File Reference

8.146.1 Detailed Description

Describes [ShutdownControl](#), which initiates the shutdown sequence.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

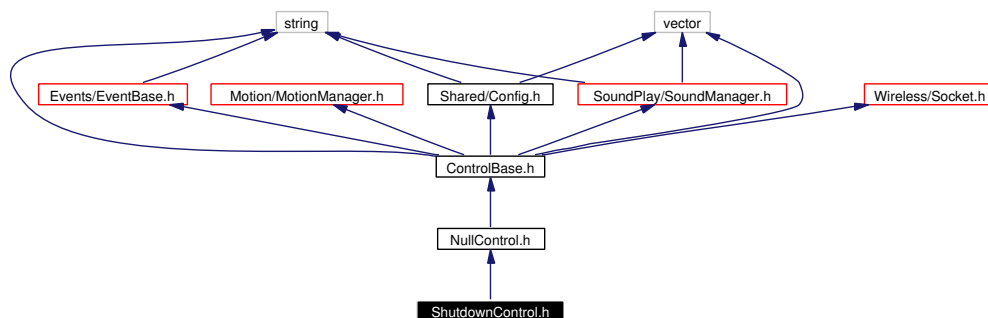
Date

2003/06/12 23:41:36

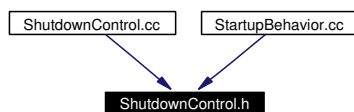
Definition in file [ShutdownControl.h](#).

```
#include "NullControl.h"
```

Include dependency graph for ShutdownControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [ShutdownControl](#)
when activated, this will cause the aibo to shut down

8.147 SimpleChaseBallBehavior.h File Reference

8.147.1 Detailed Description

Describes [SimpleChaseBallBehavior](#), which runs around after whatever the dog sees.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

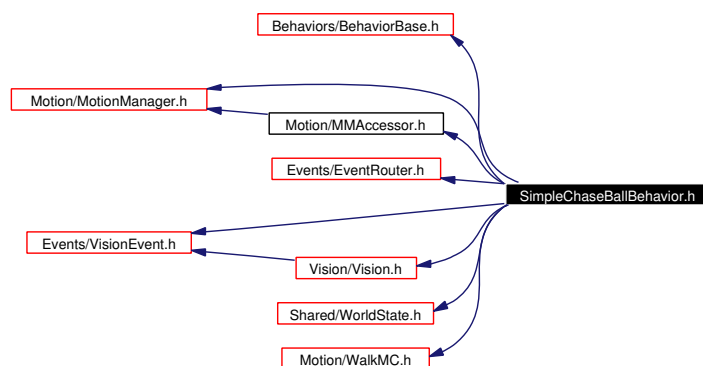
Date

2003/06/12 18:06:11

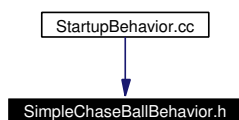
Definition in file [SimpleChaseBallBehavior.h](#).

```
#include "Behaviors/BehaviorBase.h"
#include "Motion/MotionManager.h"
#include "Motion/MMAccessor.h"
#include "Events/EventRouter.h"
#include "Events/VisionEvent.h"
#include "Shared/WorldState.h"
#include "Motion/WalkMC.h"
#include "Vision/Vision.h"
```

Include dependency graph for SimpleChaseBallBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [SimpleChaseBallBehavior](#)
A simple behavior to chase after any objects seen by the vision system.

8.148 SmoothCompareTrans.h File Reference

8.148.1 Detailed Description

Defines [SmoothCompareTrans](#), subclass of [CompareTrans](#), which provides monitoring of exponentially weighted averages to a threshold.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.4

State

Rel

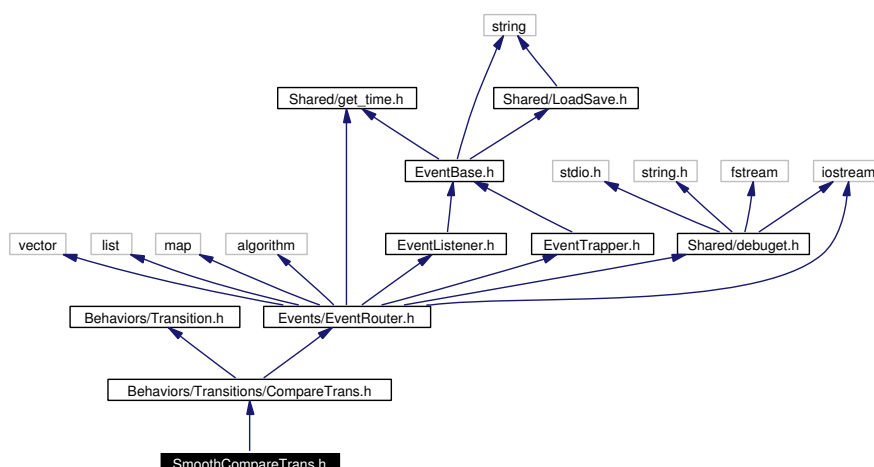
Date

2003/03/09 02:45:22

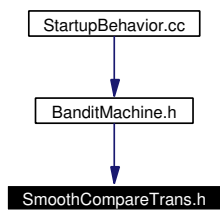
Definition in file [SmoothCompareTrans.h](#).

```
#include "Behaviors/Transitions/CompareTrans.h"
```

Include dependency graph for SmoothCompareTrans.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [SmoothCompareTrans](#)
A subclass of [CompareTrans](#), which provides monitoring of exponentially weighted averages to a threshold.

8.149 Socket.cc File Reference

8.149.1 Detailed Description

Implements Tekkotsu wireless [Socket](#) class, also sout and serr.

Author:

alokl (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.13

State

Rel

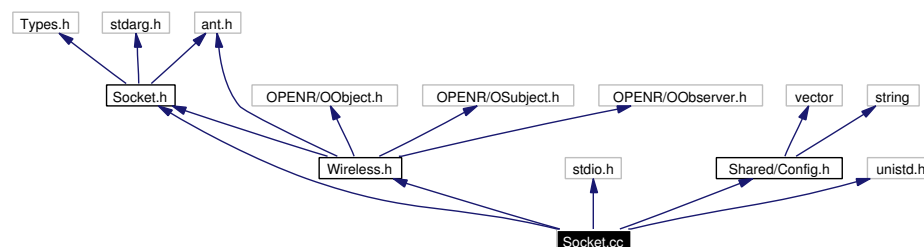
Date

2003/06/12 23:41:41

Definition in file [Socket.cc](#).

```
#include "Socket.h"
#include <stdio.h>
#include "Wireless.h"
#include "Shared/Config.h"
#include <unistd.h>
```

Include dependency graph for Socket.cc:



Variables

- `Socket * sout = NULL`
the standard tekkotsu in/out console (default port 10001)
- `Socket * serr = NULL`
the standard tekkotsu error output (default port 10002)

8.149.2 Variable Documentation

8.149.2.1 `Socket* serr = NULL`

the standard tekkotsu error output (default port 10002)

Definition at line 8 of file Socket.cc.

8.149.2.2 `Socket* sout = NULL`

the standard tekkotsu in/out console (default port 10001)

Definition at line 7 of file Socket.cc.

8.150 Socket.h File Reference

8.150.1 Detailed Description

Defines Tekkotsu wireless [Socket](#) class, also sout and serr.

Author:

alokl (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.12

State

Rel

Date

2003/06/13 03:23:05

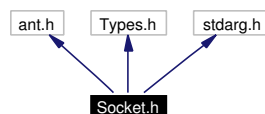
Definition in file [Socket.h](#).

```
#include <ant.h>
```

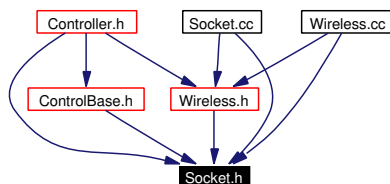
```
#include <Types.h>
```

```
#include <stdarg.h>
```

Include dependency graph for Socket.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [SocketNS](#)

Compounds

- class [Socket](#)
Tekkotsu wireless Socket class.

Variables

- [Socket](#) * [sout](#)
the standard tekkotsu in/out console (default port 10001)
- [Socket](#) * [serr](#)
the standard tekkotsu error output (default port 10002)

8.150.2 Variable Documentation

8.150.2.1 [Socket](#)* [serr](#)

the standard tekkotsu error output (default port 10002)

Definition at line 204 of file Socket.h.

8.150.2.2 [Socket](#)* [sout](#)

the standard tekkotsu in/out console (default port 10001)

Definition at line 203 of file Socket.h.

8.151 SoundManager.cc File Reference

8.151.1 Detailed Description

Implements [SoundManager](#), which provides sound effects and caching services, as well as mixing buffers for the [SoundPlay](#) process.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.13

State

Rel

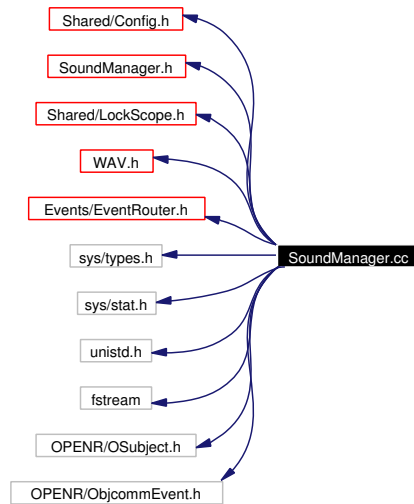
Date

2003/06/10 00:53:48

Definition in file [SoundManager.cc](#).

```
#include "Shared/Config.h"
#include "SoundManager.h"
#include "Shared/LockScope.h"
#include "WAV.h"
#include "Events/EventRouter.h"
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fstream>
#include <OPENR/OSubject.h>
#include <OPENR/ObjcommEvent.h>
```

Include dependency graph for SoundManager.cc:



Typedefs

- typedef [LockScope](#)< ProcessID::NumProcesses > [AutoLock](#)
for convenience when locking each of the functions

Variables

- [SoundManager](#) * [sndman](#) = NULL
lets you play a sound from anywhere in your code - just a one liner!

8.151.2 Typedef Documentation

8.151.2.1 typedef [LockScope](#)<ProcessID::NumProcesses> [AutoLock](#)

for convenience when locking each of the functions

Definition at line 17 of file SoundManager.cc.

8.151.3 Variable Documentation

8.151.3.1 [SoundManager](#)* [sndman](#) = NULL

lets you play a sound from anywhere in your code - just a one liner!

Definition at line 14 of file SoundManager.cc.

8.152 SoundManager.h File Reference

8.152.1 Detailed Description

Describes [SoundManager](#), which provides sound effects and caching services, as well as mixing buffers for the [SoundPlay](#) process.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.8

State

Rel

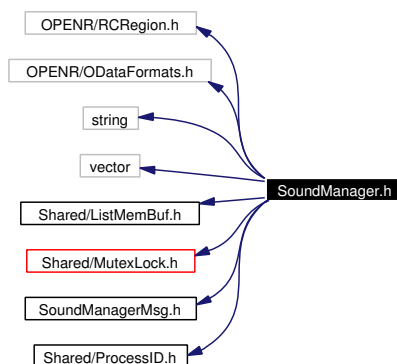
Date

2003/04/09 03:33:57

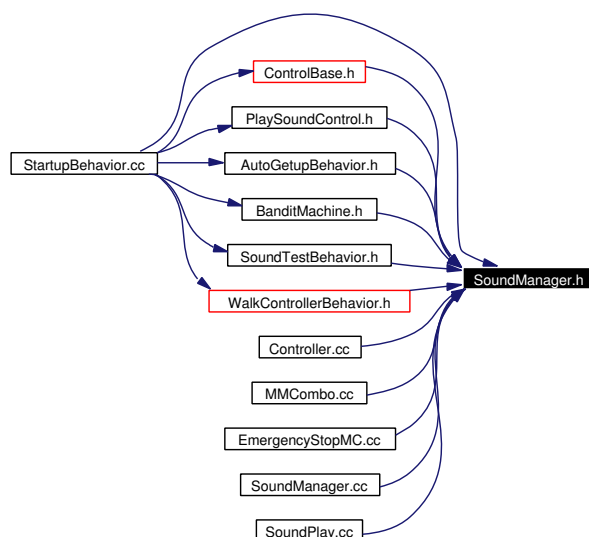
Definition in file [SoundManager.h](#).

```
#include <OPENR/RCRegion.h>
#include <OPENR/ODataFormats.h>
#include <string>
#include <vector>
#include "Shared/ListMemBuf.h"
#include "Shared/MutexLock.h"
#include "SoundManagerMsg.h"
#include "Shared/ProcessID.h"
```

Include dependency graph for SoundManager.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class `SoundManager`
Provides sound effects and caching services, as well as mixing buffers for the *Sound-Play* process.
- struct `SoundManager.PlayState`
Holds data about sounds currently being played.
- struct `SoundManager.SoundData`

Holds data about the loaded sounds.

Variables

- [SoundManager](#) * [sndman](#)

lets you play a sound from anywhere in your code - just a one liner!

8.152.2 Variable Documentation

8.152.2.1 [SoundManager](#)* [sndman](#)

lets you play a sound from anywhere in your code - just a one liner!

Definition at line 228 of file SoundManager.h.

8.153 SoundManagerMsg.h File Reference

8.153.1 Detailed Description

Defines [SoundManagerMsg](#), a small header used by [SoundManager](#) for sending messages between processes.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

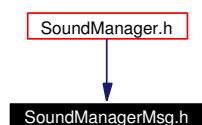
Rel

Date

2003/04/09 04:48:19

Definition in file [SoundManagerMsg.h](#).

This graph shows which files directly or indirectly include this file:



Compounds

- struct [SoundManagerMsg](#)

A small header that precedes data sent by [SoundManager](#) between processes.

8.154 SoundPlay.cc File Reference

8.154.1 Detailed Description

Implements the [SoundPlay](#) process (a.k.a. [OObject](#)), which is responsible for sending sound buffers to the system to play.

Author:

Sony (Creator)

This is basically the [SoundPlay](#) example from the Sony code, with a few modifications. Here's the license Sony provided with it:

Copyright 2002,2003 Sony Corporation

Permission to use, copy, modify, and redistribute this software for non-commercial use is hereby granted.

This software is provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of fitness for a particular purpose.

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.7

State

Rel

Date

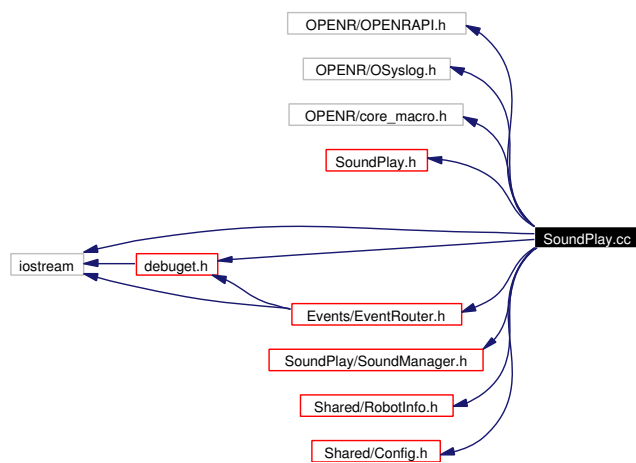
2003/06/12 23:41:41

Definition in file [SoundPlay.cc](#).

```
#include <OPENR/OPENRAPI.h>
#include <OPENR/OSyslog.h>
#include <OPENR/core_macro.h>
#include "SoundPlay.h"
#include <iostream>
#include "SoundPlay/SoundManager.h"
```

```
#include "Shared/RobotInfo.h"  
#include "Shared/Config.h"  
#include "Shared/debuget.h"  
#include "Events/EventRouter.h"
```

Include dependency graph for SoundPlay.cc:



8.155 SoundPlay.h File Reference

8.155.1 Detailed Description

Describes the [SoundPlay](#) process (a.k.a. [OObject](#)), which is responsible for sending sound buffers to the system to play.

Author:

Sony (Creator)

This is basically the [SoundPlay](#) example from the Sony code, with a few modifications. Here's the license Sony provided with it:

Copyright 2002,2003 Sony Corporation

Permission to use, copy, modify, and redistribute this software for non-commercial use is hereby granted.

This software is provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of fitness for a particular purpose.

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

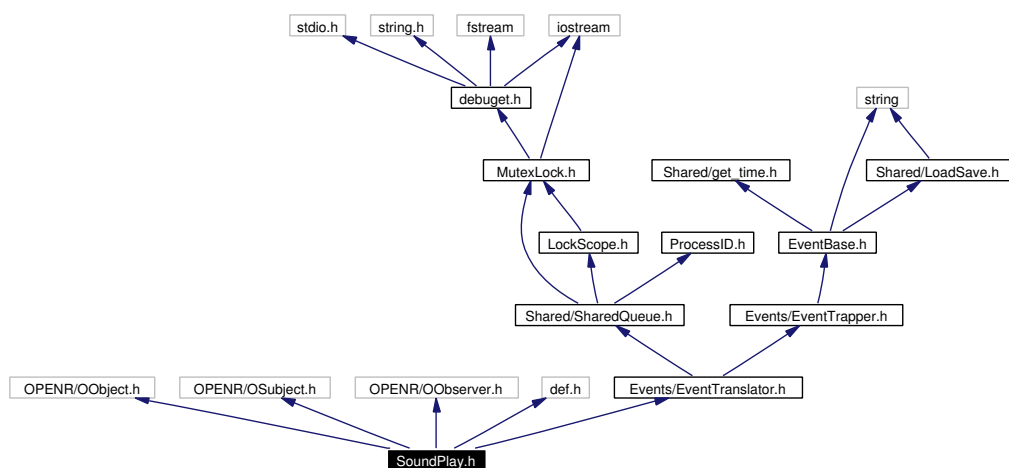
Date

2003/04/09 03:33:57

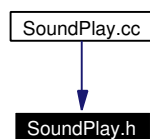
Definition in file [SoundPlay.h](#).

```
#include <OPENR/OObject.h>
#include <OPENR/OSubject.h>
#include <OPENR/OObserver.h>
#include "def.h"
#include "Events/EventTranslator.h"
```

Include dependency graph for SoundPlay.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [SoundPlay](#)

The process (a.k.a. [OObject](#)), which is responsible for sending sound buffers to the system to play.

8.156 SoundTestBehavior.h File Reference

8.156.1 Detailed Description

Defines the [SoundTestBehavior](#) demo, which allows you to experiment with playing sounds different ways.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.8

State

Rel

Date

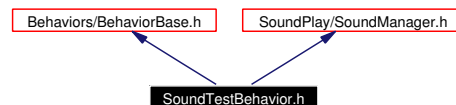
2003/06/05 17:03:15

Definition in file [SoundTestBehavior.h](#).

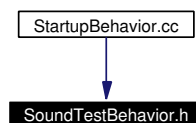
```
#include "Behaviors/BehaviorBase.h"
```

```
#include "SoundPlay/SoundManager.h"
```

Include dependency graph for SoundTestBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [SoundTestBehavior](#)
allows you to experiment with playing sounds different ways.

8.157 Spline.h File Reference

8.157.1 Detailed Description

Performs calculations regarding splines for path execution.

Author:

James R. Bruce (Creator)

```
=====
spline.h : Implementation of Uniform and Non-Uniform Hermite Splines
-----
Copyright (C) 1999-2002 James R. Bruce
School of Computer Science, Carnegie Mellon University
-----
This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
=====
*
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.4

State

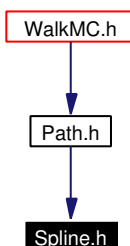
Rel

Date

2003/01/23 18:14:04

Definition in file [Spline.h](#).

This graph shows which files directly or indirectly include this file:



Compounds

- class [HermiteSplineSegment](#)
- class [NonUniformHermiteSplineSegment](#)

Defines

- #define [HSPLINE](#) [HermiteSplineSegment](#)<point,fnum>
- #define [NUHSPLINE](#) [NonUniformHermiteSplineSegment](#)<point,fnum>
- #define [HSPLINE_TEM](#) template <class point,class fnum>
- #define [NUHSPLINE_TEM](#) template <class point,class fnum>
- #define [EPS](#) 1E-6

8.157.2 Define Documentation

8.157.2.1 #define EPS 1E-6

Definition at line 29 of file Spline.h.

8.157.2.2 #define HSPLINE [HermiteSplineSegment](#)<point,fnum>

Definition at line 19 of file Spline.h.

8.157.2.3 #define HSPLINE_TEM template <class point,class fnum>

Definition at line 22 of file Spline.h.

8.157.2.4 #define NUHSPLINE [NonUniformHermiteSplineSegment](#)<point,fnum>

Definition at line 20 of file Spline.h.

8.157.2.5 `#define NUHSPLINE_TEM template <class point,class fnum>`

Definition at line 23 of file Spline.h.

8.158 StareAtBallBehavior.cc File Reference

8.158.1 Detailed Description

Implements [StareAtBallBehavior](#), which points the head at the ball.

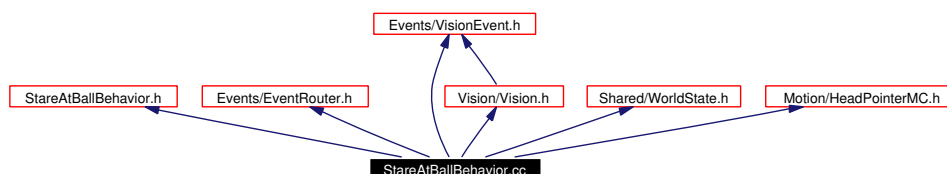
Author:

tss (Creator)

Definition in file [StareAtBallBehavior.cc](#).

```
#include "StareAtBallBehavior.h"
#include "Events/EventRouter.h"
#include "Events/VisionEvent.h"
#include "Shared/WorldState.h"
#include "Motion/HeadPointerMC.h"
#include "Vision/Vision.h"
```

Include dependency graph for StareAtBallBehavior.cc:



Functions

- double [DtoR](#) (double deg)
Converts degrees to radians.

8.158.2 Function Documentation

8.158.2.1 double DtoR (double *deg*) [inline]

Converts degrees to radians.

Definition at line 9 of file [StareAtBallBehavior.cc](#).

8.159 StareAtBallBehavior.h File Reference

8.159.1 Detailed Description

Describes [StareAtBallBehavior](#), which runs around after whatever the dog sees.

Author:

tss (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

Date

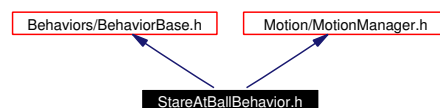
2003/06/05 17:03:15

Definition in file [StareAtBallBehavior.h](#).

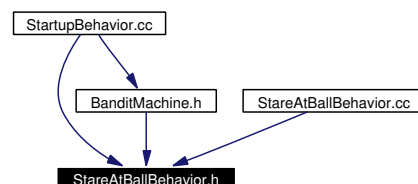
```
#include "Behaviors/BehaviorBase.h"
```

```
#include "Motion/MotionManager.h"
```

Include dependency graph for StareAtBallBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

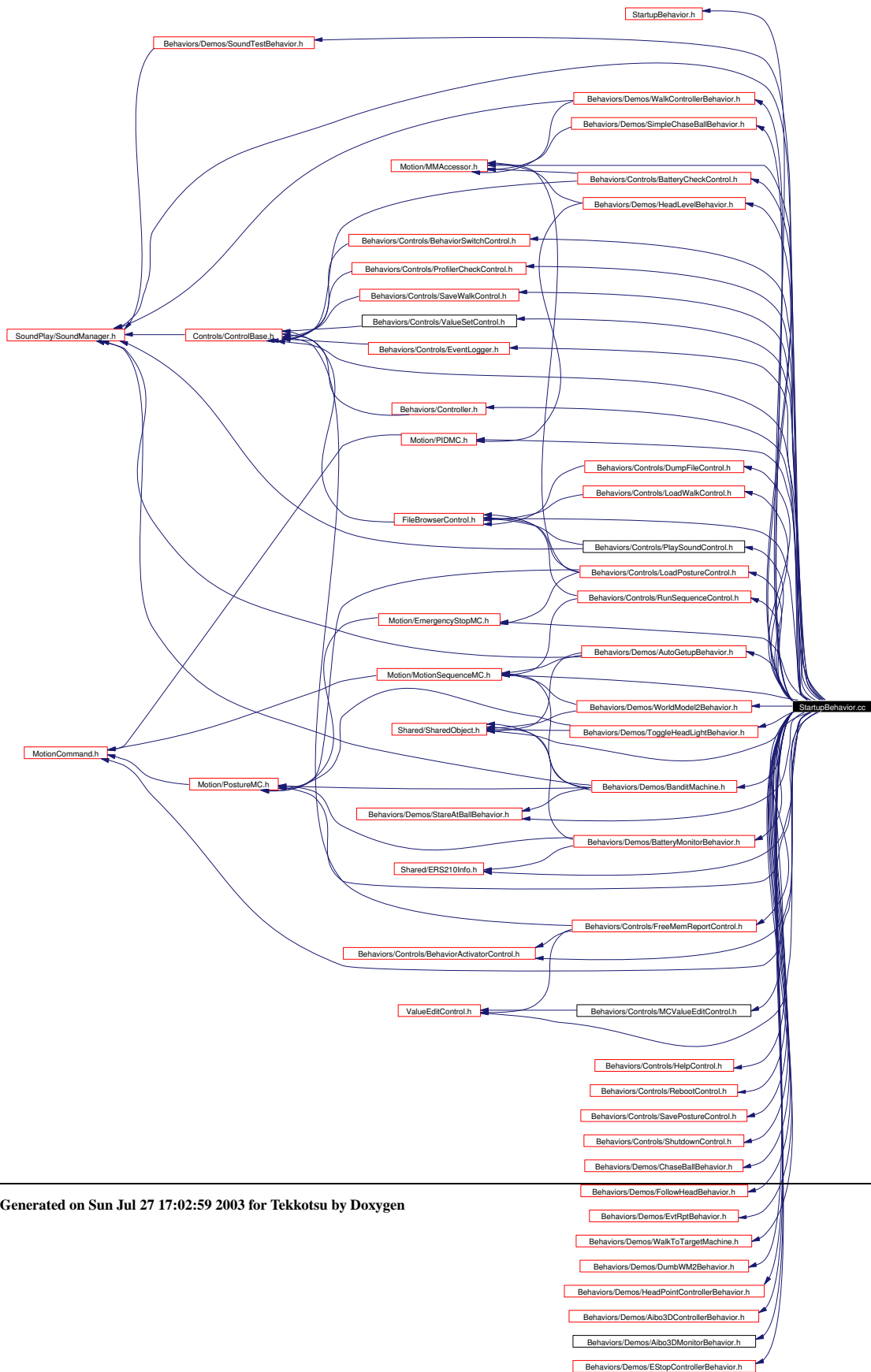
- class [StareAtBallBehavior](#)

A simple behavior to chase after any objects seen by the vision system.

8.160 StartupBehavior.cc File Reference

```
#include "StartupBehavior.h"
#include "Behaviors/Controller.h"
#include "Shared/SharedObject.h"
#include "Behaviors/Controls/BatteryCheckControl.h"
#include "Behaviors/Controls/BehaviorActivatorControl.h"
#include "Behaviors/Controls/BehaviorSwitchControl.h"
#include "Behaviors/Controls/ControlBase.h"
#include "Behaviors/Controls/DumpFileControl.h"
#include "Behaviors/Controls/FileBrowserControl.h"
#include "Behaviors/Controls/FreeMemReportControl.h"
#include "Behaviors/Controls/HelpControl.h"
#include "Behaviors/Controls/LoadPostureControl.h"
#include "Behaviors/Controls/LoadWalkControl.h"
#include "Behaviors/Controls/MCValueEditControl.h"
#include "Behaviors/Controls/PlaySoundControl.h"
#include "Behaviors/Controls/ProfilerCheckControl.h"
#include "Behaviors/Controls/RebootControl.h"
#include "Behaviors/Controls/RunSequenceControl.h"
#include "Behaviors/Controls/SavePostureControl.h"
#include "Behaviors/Controls/SaveWalkControl.h"
#include "Behaviors/Controls/ShutdownControl.h"
#include "Behaviors/Controls/ValueEditControl.h"
#include "Behaviors/Controls/ValueSetControl.h"
#include "Behaviors/Controls/EventLogger.h"
#include "Behaviors/Demos/AutoGetupBehavior.h"
#include "Behaviors/Demos/BatteryMonitorBehavior.h"
#include "Behaviors/Demos/ChaseBallBehavior.h"
#include "Behaviors/Demos/SimpleChaseBallBehavior.h"
#include "Behaviors/Demos/StareAtBallBehavior.h"
```

```
#include "Behaviors/Demos/FollowHeadBehavior.h"
#include "Behaviors/Demos/HeadLevelBehavior.h"
#include "Behaviors/Demos/EvtRptBehavior.h"
#include "Behaviors/Demos/WalkToTargetMachine.h"
#include "Behaviors/Demos/BanditMachine.h"
#include "Behaviors/Demos/WorldModel2Behavior.h"
#include "Behaviors/Demos/DumbWM2Behavior.h"
#include "Behaviors/Demos/SoundTestBehavior.h"
#include "Behaviors/Demos/ToggleHeadLightBehavior.h"
#include "Behaviors/Demos/WalkControllerBehavior.h"
#include "Behaviors/Demos/HeadPointControllerBehavior.h"
#include "Behaviors/Demos/Aibo3DControllerBehavior.h"
#include "Behaviors/Demos/Aibo3DMonitorBehavior.h"
#include "Behaviors/Demos/ESTopControllerBehavior.h"
#include "Motion/MotionCommand.h"
#include "Motion/PostureMC.h"
#include "Motion/EmergencyStopMC.h"
#include "Motion/PIDMC.h"
#include "Motion/MotionSequenceMC.h"
#include "Motion/MMAccessor.h"
#include "Shared/ERS210Info.h"
#include "SoundPlay/SoundManager.h"
Include dependency graph for StartupBehavior.cc:
```



Variables

- [StartupBehavior gStartup](#)
used to initialize the global [startupBehavior](#), used by [MMCombo](#)
- [BehaviorBase & startupBehavior = gStartup](#)
used by [MMCombo](#) as the init behavior

8.160.1 Variable Documentation

8.160.1.1 [StartupBehavior gStartup](#)

used to initialize the global [startupBehavior](#), used by [MMCombo](#)

Definition at line 59 of file StartupBehavior.cc.

8.160.1.2 [BehaviorBase& startupBehavior = gStartup](#)

used by [MMCombo](#) as the init behavior

Definition at line 60 of file StartupBehavior.cc.

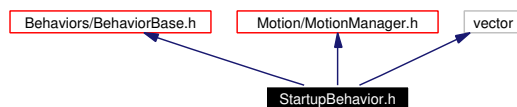
8.161 StartupBehavior.h File Reference

```
#include "Behaviors/BehaviorBase.h"
```

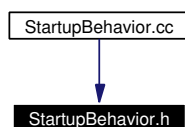
```
#include "Motion/MotionManager.h"
```

```
#include <vector>
```

Include dependency graph for StartupBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [StartupBehavior](#)

This Behavior is the only hardcoded behavior by the framework to start up once all the data structures and such are set up.

Variables

- [BehaviorBase](#) & [startupBehavior](#)

used by [MMCombo](#) as the init behavior

8.161.1 Variable Documentation

8.161.1.1 [BehaviorBase](#) & [startupBehavior](#)

used by [MMCombo](#) as the init behavior

Definition at line 37 of file StartupBehavior.h.

8.162 StateNode.cc File Reference

8.162.1 Detailed Description

Describes [StateNode](#), which is both a state machine controller as well as a node within a state machine itself.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

Date

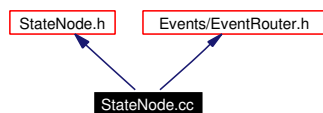
2003/06/05 17:03:09

Definition in file [StateNode.cc](#).

```
#include "StateNode.h"
```

```
#include "Events/EventRouter.h"
```

Include dependency graph for StateNode.cc:



8.163 StateNode.h File Reference

8.163.1 Detailed Description

Describes [StateNode](#), which is both a state machine controller as well as a node within a state machine itself.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

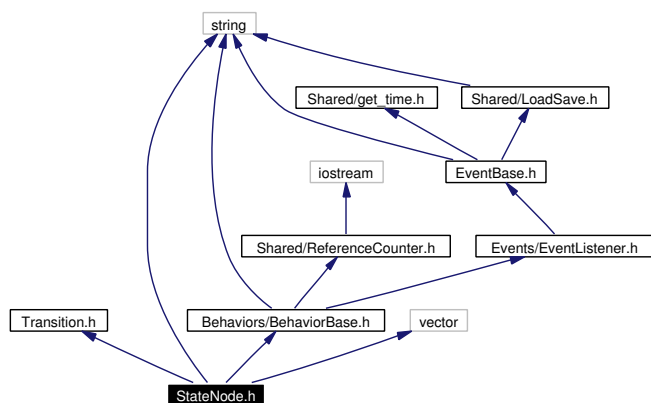
Date

2003/06/05 17:03:09

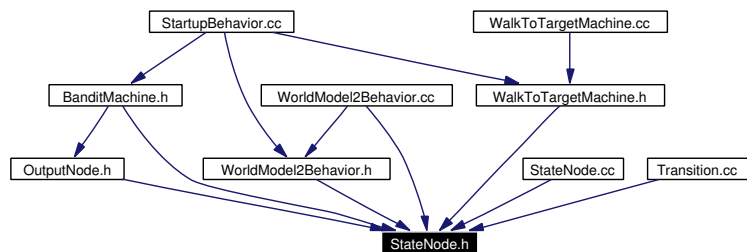
Definition in file [StateNode.h](#).

```
#include "Transition.h"
#include "Behaviors/BehaviorBase.h"
#include <vector>
#include <string>
```

Include dependency graph for StateNode.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [StateNode](#)

Recursive data structure - both a state machine controller as well as a node within a state machine itself.

8.164 StringInputControl.cc File Reference

8.164.1 Detailed Description

Implements [StringInputControl](#), which prompts for and stores a string from the user.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

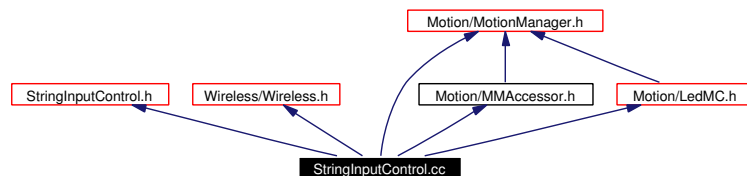
Date

2003/06/12 23:41:36

Definition in file [StringInputControl.cc](#).

```
#include "StringInputControl.h"
#include "Wireless/Wireless.h"
#include "Motion/MMAccessor.h"
#include "Motion/LedMC.h"
#include "Motion/MotionManager.h"
```

Include dependency graph for StringInputControl.cc:



8.165 StringInputControl.h File Reference

8.165.1 Detailed Description

Defines [StringInputControl](#), which prompts for and stores a string from the user.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

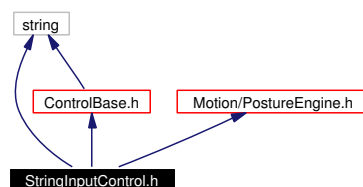
Date

2003/06/12 18:06:10

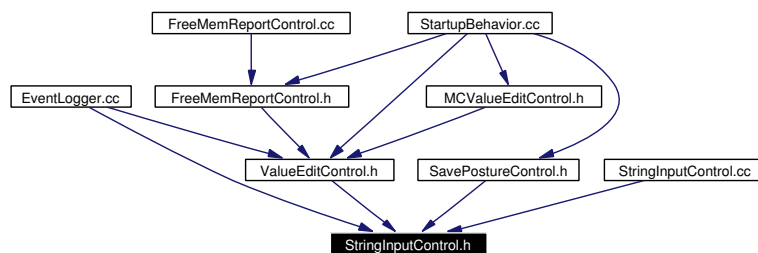
Definition in file [StringInputControl.h](#).

```
#include "ControlBase.h"
#include "Motion/PostureEngine.h"
#include <string>
```

Include dependency graph for StringInputControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [StringInputControl](#)

Upon activation, prompts the user for a string and stores it.

8.166 SystemUtility.h File Reference

8.166.1 Detailed Description

Wrappers for getting large memory regions from Aperios.

Author:

CMU RoboSoccer 2001-2002 (Creator)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
-----

This software is distributed under the GNU General Public License,
version 2.  If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA.  This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
-----

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed.  In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

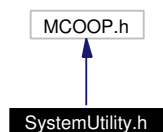
Date

2003/01/24 00:37:39

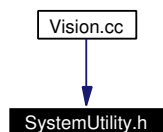
Definition in file [SystemUtility.h](#).

```
#include <MCOOP.h>
```

Include dependency graph for SystemUtility.h:



This graph shows which files directly or indirectly include this file:



Functions

- `template<class T> T * NewLarge (T **dst, int count)`
- `template<class T> void DeleteLarge (T *dst)`

8.166.2 Function Documentation

8.166.2.1 `template<class T> void DeleteLarge (T * dst)`

Definition at line 46 of file SystemUtility.h.

8.166.2.2 `template<class T> T* NewLarge (T ** dst, int count)`

Definition at line 36 of file SystemUtility.h.

8.167 TailWagMC.h File Reference

8.167.1 Detailed Description

Defines [TailWagMC](#), which will wag the tail on a ERS-210 robot.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.4

State

Rel

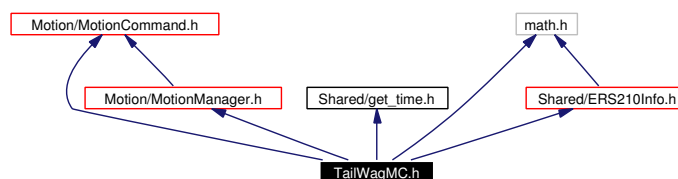
Date

2003/07/09 00:10:57

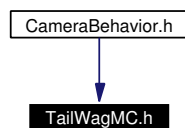
Definition in file [TailWagMC.h](#).

```
#include "Motion/MotionCommand.h"
#include "Motion/MotionManager.h"
#include "Shared/get_time.h"
#include "math.h"
#include "Shared/ERS210Info.h"
```

Include dependency graph for TailWagMC.h:



This graph shows which files directly or indirectly include this file:



Compounds

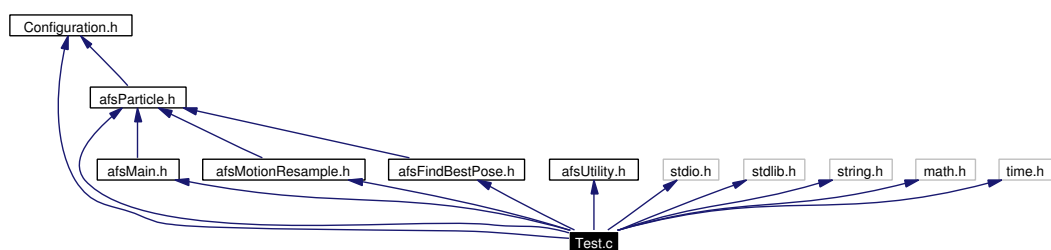
- class [TailWagMC](#)

A simple motion command for wagging the tail - you can specify period, magnitude, and tilt.

8.168 Test.c File Reference

```
#include "afsMain.h"
#include "afsParticle.h"
#include "afsMotionResample.h"
#include "afsFindBestPose.h"
#include "afsUtility.h"
#include "Configuration.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
```

Include dependency graph for Test.c:



Functions

- int [main](#) (int argc, char **argv)

Variables

- [afsParticle](#) * [Particles](#)
- [afsXY](#) [landmarks](#) [AFS_NUM_LANDMARKS]

8.168.1 Function Documentation

8.168.1.1 `int main (int argc, char ** argv)`

Definition at line 25 of file Test.c.

References AFS_MEASURE_VARIANCE, AFS_NUM_LANDMARKS, AFS_NUM_PARTICLES, afsApplyBestPose(), afsCertainty(), afsDistribute(), afsFindBestPose(), afsInit(), afsMeasurement(), afsMotion(), afsMotionResample(), afsParticleInit(), afsResample(), afsSetLandmark(), afsWhatsUp(), find_dtheta(), _afsParticle::landmarks, landmarks, _afsLandmarkLoc::mean, normRand(), Particles, _afsParticle::pose, _afsLandmarkLoc::state, _afsPose::theta, time, _afsLandmarkLoc::variance, _afsPose::x, _afsXY::x, _afsPose::y, and _afsXY::y.

8.168.2 Variable Documentation

8.168.2.1 `afsXY landmarks[AFS_NUM_LANDMARKS]`

Definition at line 22 of file Test.c.

8.168.2.2 `afsParticle* Particles`

Definition at line 20 of file Test.c.

8.169 TextMsgEvent.h File Reference

8.169.1 Detailed Description

Defines [TextMsgEvent](#), which extends [EventBase](#) to also include actual message text.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.7

State

Rel

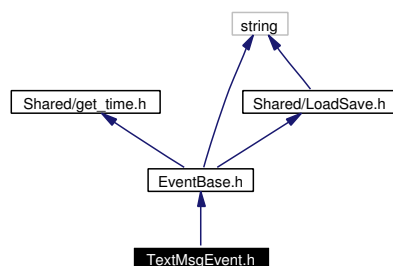
Date

2003/06/12 23:41:40

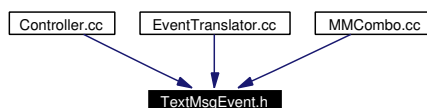
Definition in file [TextMsgEvent.h](#).

```
#include "EventBase.h"
```

Include dependency graph for TextMsgEvent.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [TextMsgEvent](#)

Extends [EventBase](#) to also include actual message text.

8.170 TimeET.cc File Reference

8.170.1 Detailed Description

Implements [TimeET](#), a nice class for handling time values with high precision (but all that's in the .cc is implementation of struct timezone [TimeET::tz](#)).

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

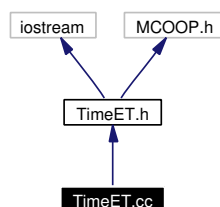
Date

2003/01/09 02:02:59

Definition in file [TimeET.cc](#).

```
#include "TimeET.h"
```

Include dependency graph for TimeET.cc:



8.171 TimeET.h File Reference

8.171.1 Detailed Description

Describes [TimeET](#), a nice class for handling time values with high precision.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

Date

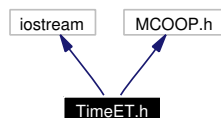
2003/04/18 05:15:11

Definition in file [TimeET.h](#).

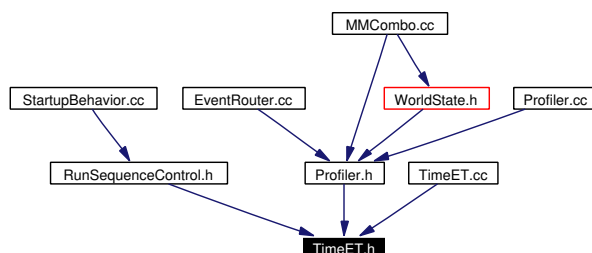
```
#include <iostream>
```

```
#include <MCOOP.h>
```

Include dependency graph for TimeET.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class `TimeET`
a nice class for handling time values with high precision
- struct `timeval`
would be defined by system - we redefine the same structure in case we're compiling for Aperios
- struct `timezone`
would be defined by system - we redefine the same structure in case we're compiling for Aperios

Functions

- `std::ostream & operator<< (std::ostream &o, const TimeET &t)`
lets the class be displayed easily

8.171.2 Function Documentation

8.171.2.1 `TimeET` operator+ (long *t1*, const `TimeET` &*t2*) [inline]

for doing doing math with time

Definition at line 137 of file TimeET.h.

8.171.2.2 `TimeET` operator- (double *t1*, const `TimeET` &*t2*) [inline]

for doing doing math with time

Definition at line 138 of file TimeET.h.

8.171.2.3 `std::ostream& operator<< (std::ostream & o, const TimeET & t)` [inline]

lets the class be displayed easily

Definition at line 142 of file TimeET.h.

References TimeET::tv, timeval::tv_sec, and timeval::tv_usec.

8.172 TimeOutTrans.h File Reference

8.172.1 Detailed Description

Defines [TimeOutTrans](#), which causes a transition after a specified amount of time has passed.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

Date

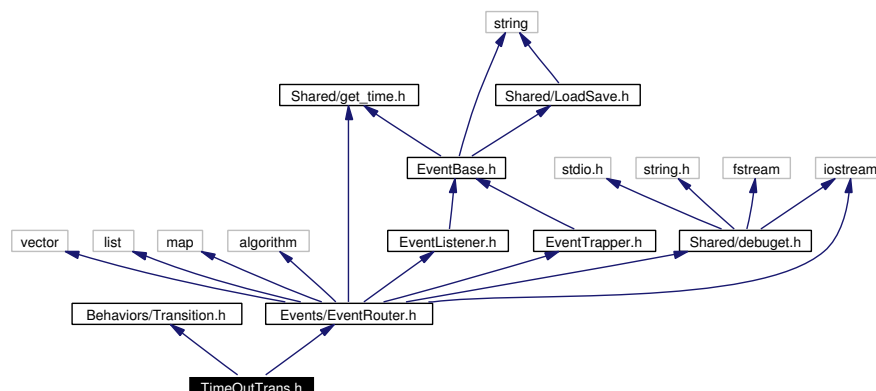
2003/03/09 02:45:22

Definition in file [TimeOutTrans.h](#).

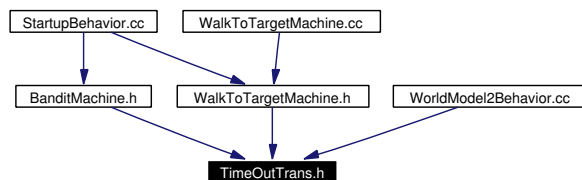
```
#include "Behaviors/Transition.h"
```

```
#include "Events/EventRouter.h"
```

Include dependency graph for TimeOutTrans.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [TimeOutTrans](#)
causes a transition after a specified amount of time has passed

8.173 ToggleHeadLightBehavior.h File Reference

8.173.1 Detailed Description

Defines [ToggleHeadLightBehavior](#), which will open or close the head light on an ERS-220.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.1

State

Rel

Date

2003/06/28 18:03:57

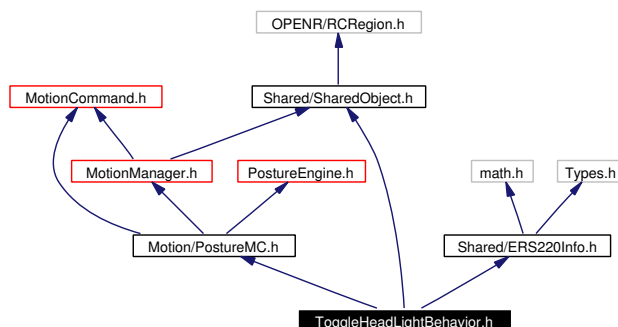
Definition in file [ToggleHeadLightBehavior.h](#).

```
#include "Shared/SharedObject.h"
```

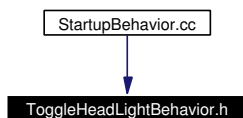
```
#include "Motion/PostureMC.h"
```

```
#include "Shared/ERS220Info.h"
```

Include dependency graph for ToggleHeadLightBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [ToggleHeadLightBehavior](#)
opens or closes the head light on an ERS-220

8.174 Transition.cc File Reference

8.174.1 Detailed Description

Implements [Transition](#), represents a transition between StateNodes.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.1

State

Rel

Date

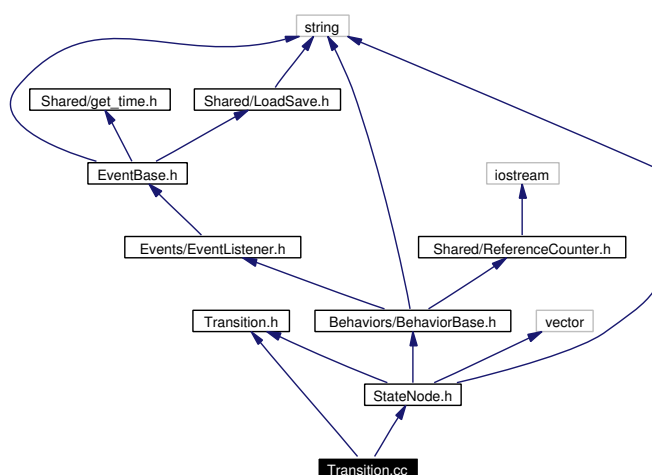
2003/03/01 20:53:26

Definition in file [Transition.cc](#).

```
#include "Transition.h"
```

```
#include "StateNode.h"
```

Include dependency graph for Transition.cc:



8.175 Transition.h File Reference

8.175.1 Detailed Description

Describes [Transition](#), represents a transition between StateNodes.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.1

State

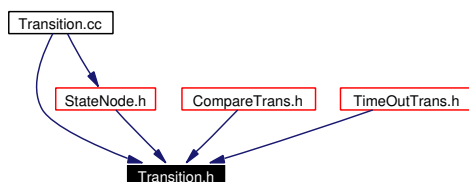
Rel

Date

2003/03/01 20:53:26

Definition in file [Transition.h](#).

This graph shows which files directly or indirectly include this file:



Compounds

- class [Transition](#)

Represents a transition between StateNodes.

8.176 Util.h File Reference

8.176.1 Detailed Description

Numerical Utilities.

Author:

James R. Bruce (Creator)

```
=====
Util.h
-----
Numerical utility functions
-----
Copyright 1999, 2000, 2001 James R. Bruce
School of Computer Science, Carnegie Mellon University
-----
This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License as
published by the Free Software Foundation; either version 2 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to: Free Software Foundation,
Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
=====
*
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.4

State

Rel

Date

2003/01/23 18:14:11

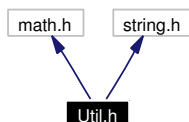
Definition in file [Util.h](#).

```
#include <math.h>
```

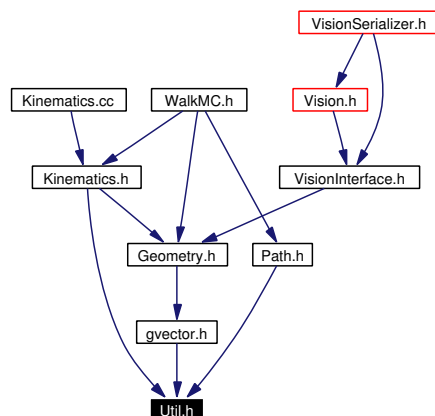


```
#include <string.h>
```

Include dependency graph for Util.h:



This graph shows which files directly or indirectly include this file:



Functions

- `template<class num1, class num2> num1 bound (num1 x, num2 low, num2 high)`
- `template<class num> num max3 (num a, num b, num c)`
- `template<class num> num min3 (num a, num b, num c)`
- `template<class num> void sort (num &a, num &b, num &c)`
- `template<class real> real fmodt (real x, real m)`
- `template<class num1, class num2> num1 setbits (num1 val, num2 bits)`
- `template<class num1, class num2> num1 clearbits (num1 val, num2 bits)`
- `template<class real> real saw (real t)`
- `template<class data> int mcopy (data *dest, data *src, int num)`
- `template<class data> data mset (data *dest, data val, int num)`
- `template<class data> void mzero (data &d)`
- `template<class node> int list_length (node *list)`
- `double norm_angle (double angle)`

- void [angle_wrap](#) (double &angle)
- double [avg_angle](#) (double left, double right)
- double [atan2a](#) (double y, double x)
- double [atan2b](#) (double y, double x)
- template<class num> int [sign](#) (num n)
- double [gaussian_with_min](#) (double x, double min_value)
- double [gaussian_prob](#) (double x, double mean, double sigma)

Variables

- const double [TODEG](#) = (180.0 / M_PI)
- const double [gaussian_constant](#) = sqrt(2*M_PI)

8.176.2 Function Documentation

8.176.2.1 void [angle_wrap](#) (double &*angle*) [inline]

Definition at line 177 of file Util.h.

8.176.2.2 double [atan2a](#) (double *y*, double *x*) [inline]

Definition at line 199 of file Util.h.

8.176.2.3 double [atan2b](#) (double *y*, double *x*) [inline]

Definition at line 209 of file Util.h.

8.176.2.4 double [avg_angle](#) (double *left*, double *right*) [inline]

Definition at line 185 of file Util.h.

8.176.2.5 template<class num1, class num2> num1 [bound](#) (num1 *x*, num2 *low*, num2 *high*) [inline]

Definition at line 61 of file Util.h.

8.176.2.6 template<class num1, class num2> num1 [clearbits](#) (num1 *val*, num2 *bits*) [inline]

Definition at line 111 of file Util.h.

8.176.2.7 `template<class real> real fmodt (real x, real m)`

Definition at line 97 of file Util.h.

8.176.2.8 `double gaussian_prob (double x, double mean, double sigma)`
`[inline]`

Definition at line 241 of file Util.h.

References `gaussian_constant`.

8.176.2.9 `double gaussian_with_min (double x, double min_value)` `[inline]`

Definition at line 227 of file Util.h.

8.176.2.10 `template<class node> int list_length (node * list)`

Definition at line 151 of file Util.h.

8.176.2.11 `template<class num> num max3 (num a, num b, num c)`
`[inline]`

Definition at line 69 of file Util.h.

8.176.2.12 `template<class data> int mcopy (data * dest, data * src, int num)`
`[inline]`

Definition at line 124 of file Util.h.

8.176.2.13 `template<class num> num min3 (num a, num b, num c)`
`[inline]`

Definition at line 79 of file Util.h.

8.176.2.14 `template<class data> data mset (data * dest, data val, int num)`
`[inline]`

Definition at line 134 of file Util.h.

8.176.2.15 `template<class data> void mzero (data & d)` [inline]

Definition at line 144 of file Util.h.

8.176.2.16 `double norm_angle (double angle)` [inline]

Definition at line 166 of file Util.h.

8.176.2.17 `template<class real> real saw (real t)`

Definition at line 117 of file Util.h.

8.176.2.18 `template<class num1, class num2> num1 setbits (num1 val, num2 bits)` [inline]

Definition at line 105 of file Util.h.

8.176.2.19 `template<class num> int sign (num n)` [inline]

Definition at line 219 of file Util.h.

8.176.2.20 `template<class num> void sort (num & a, num & b, num & c)`
[inline]

Definition at line 89 of file Util.h.

8.176.3 Variable Documentation

8.176.3.1 `const double gaussian_constant = sqrt(2*M_PI)`

Definition at line 239 of file Util.h.

8.176.3.2 `const double TODEG = (180.0 / M_PI)`

Definition at line 31 of file Util.h.

8.177 ValueEditControl.h File Reference

8.177.1 Detailed Description

Defines [ValueEditControl](#) class, which will allow modification of a value through a pointer.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.8

State

Rel

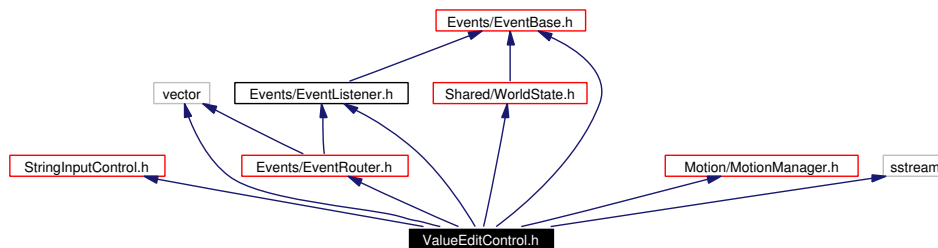
Date

2003/06/09 20:10:17

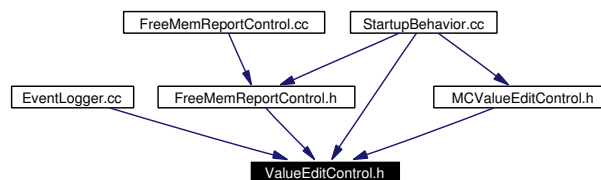
Definition in file [ValueEditControl.h](#).

```
#include "StringInputControl.h"
#include "Events/EventListener.h"
#include "Events/EventBase.h"
#include <vector>
#include "Motion/MotionManager.h"
#include "Events/EventRouter.h"
#include "Shared/WorldState.h"
#include <sstream>
```

Include dependency graph for ValueEditControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [ValueEditControl](#)
allows real-time modification of a value through a pointer

8.178 ValueSetControl.h File Reference

8.178.1 Detailed Description

Defines [ValueSetControl](#) class, which will assign a value through a pointer upon activation.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.1

State

Rel

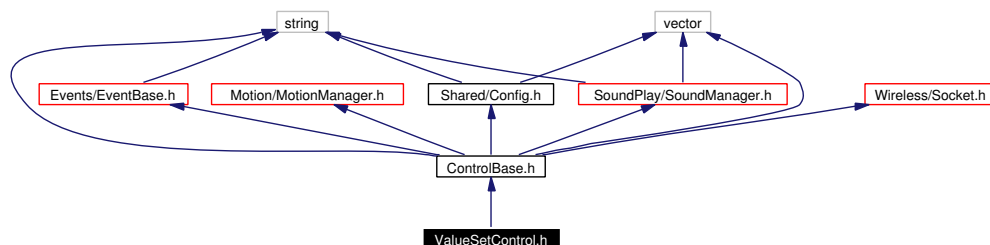
Date

2003/03/01 20:53:29

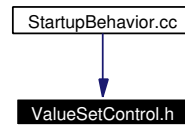
Definition in file [ValueSetControl.h](#).

```
#include "ControlBase.h"
```

Include dependency graph for ValueSetControl.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [ValueSetControl](#)

Upon activation, this control will set the target pointer to the specified value.

8.179 Vision.cc File Reference

8.179.1 Detailed Description

Does majority of vision processing.

Author:

CMU RoboSoccer 2001-2002 (Creator)
alokl (ported)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
=====

This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
=====

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed. In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.11

State

Rel

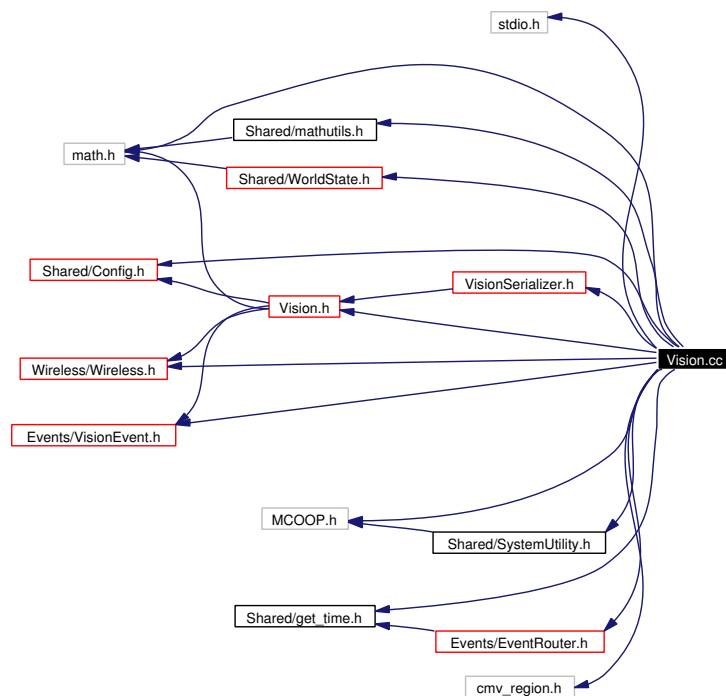
Date

2003/06/12 23:41:41

Definition in file [Vision.cc](#).

```
#include <stdio.h>
#include <math.h>
#include <MCOOP.h>
#include "Shared/Config.h"
#include "Shared/mathutils.h"
#include "Events/EventRouter.h"
#include "cmv_region.h"
#include "Vision.h"
#include "Shared/WorldState.h"
#include "Events/VisionEvent.h"
#include "Shared/get_time.h"
#include "Shared/SystemUtility.h"
#include "VisionSerializer.h"
#include "Wireless/Wireless.h"
```

Include dependency graph for Vision.cc:



Namespaces

- namespace [VisionInterface](#)

Functions

- double [pct_from_mean](#) (double *a*, double *b*)
- int [WritePPM](#) (char *filename, rgb *img, int width, int height)

Variables

- [Vision](#) * [vision](#) = NULL

8.179.2 Function Documentation

8.179.2.1 double pct_from_mean (double *a*, double *b*) [inline]

Definition at line 294 of file Vision.cc.

8.179.2.2 `int WritePPM (char * filename, rgb * img, int width, int height)`

Definition at line 1004 of file Vision.cc.

8.179.3 **Variable Documentation****8.179.3.1** `Vision* vision = NULL`

Definition at line 18 of file Vision.cc.

8.180 Vision.h File Reference

8.180.1 Detailed Description

Does majority of vision processing.

Author:

CMU RoboSoccer 2001-2002 (Creator)
alokl (ported)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
-----

This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
-----

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed. In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

alokl

Name

tekkotsu-1.4.1

Revision

1.7

State

Rel

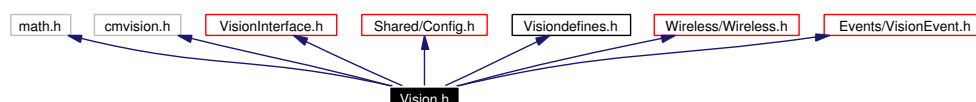
Date

2003/03/27 04:25:04

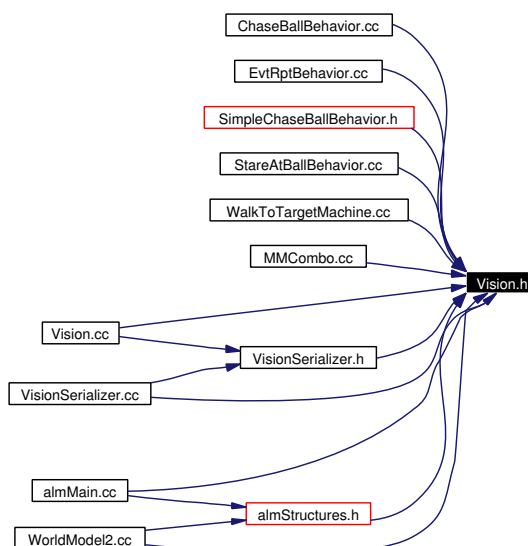
Definition in file [Vision.h](#).

```
#include <math.h>
#include "cmvision.h"
#include "VisionInterface.h"
#include "Shared/Config.h"
#include "Visiondefines.h"
#include "Wireless/Wireless.h"
#include "Events/VisionEvent.h"
```

Include dependency graph for Vision.h:



This graph shows which files directly or indirectly include this file:



Compounds

- struct [Marker](#)

- class [Vision](#)
- struct [VisionEventSpec](#)
- struct [VisionObjectInfo](#)

Defines

- `#define` [MAX_TMAPS](#) 16
- `#define` [MIN_EXP_REGION_SIZE](#) 16
- `#define` [MIN_EXP_RUN_LENGTH](#) 8

Typedefs

- typedef uchar [pixel](#)
- typedef uchar [cmap_t](#)
- typedef [CMVision::image](#)< uchar > [image](#)
- typedef [CMVision::color_class_state](#) [color_class_state](#)
- typedef [CMVision::run](#)< [cmap_t](#) > [run](#)
- typedef [CMVision::region](#) [region](#)

Variables

- const int [bits_y](#) = 4
- const int [bits_u](#) = 6
- const int [bits_v](#) = 6
- const double [FocalDist](#) = (176.0/2)/tan(HorzFOV/2)
- const double [FocalDistV](#) = (144.0/2)/tan(VertFOV/2)
- const double [YPixelSize](#) = ([FocalDistV](#)/[FocalDist](#))
- const double [RobotArea](#) = 13711
- [Vision](#) * [vision](#)

8.180.2 Define Documentation

8.180.2.1 `#define` [MAX_TMAPS](#) 16

Definition at line 44 of file [Vision.h](#).

8.180.2.2 `#define` [MIN_EXP_REGION_SIZE](#) 16

Definition at line 71 of file [Vision.h](#).

8.180.2.3 #define MIN_EXP_RUN_LENGTH 8

Definition at line 72 of file Vision.h.

8.180.3 Typedef Documentation**8.180.3.1 typedef uchar [cmap_t](#)**

Definition at line 48 of file Vision.h.

8.180.3.2 typedef [CMVision::color_class_state](#) color_class_state

Definition at line 50 of file Vision.h.

8.180.3.3 typedef [CMVision::image](#)<uchar> [image](#)

Definition at line 49 of file Vision.h.

8.180.3.4 typedef uchar [pixel](#)

Definition at line 47 of file Vision.h.

8.180.3.5 typedef [CMVision::region](#) region

Definition at line 52 of file Vision.h.

8.180.3.6 typedef [CMVision::run](#)<[cmap_t](#)> [run](#)

Definition at line 51 of file Vision.h.

8.180.4 Variable Documentation**8.180.4.1 const int [bits_u](#) = 6**

Definition at line 55 of file Vision.h.

8.180.4.2 const int [bits_v](#) = 6

Definition at line 56 of file Vision.h.

8.180.4.3 `const int bits_y = 4`

Definition at line 54 of file Vision.h.

8.180.4.4 `const double FocalDist = (176.0/2)/tan(HorzFOV/2) [static]`

Definition at line 64 of file Vision.h.

8.180.4.5 `const double FocalDistV = (144.0/2)/tan(VertFOV/2) [static]`

Definition at line 65 of file Vision.h.

8.180.4.6 `const double RobotArea = 13711 [static]`

Definition at line 69 of file Vision.h.

8.180.4.7 `Vision* vision`

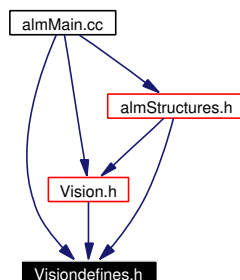
Definition at line 219 of file Vision.h.

8.180.4.8 `const double YPixelSize = (FocalDistV/FocalDist) [static]`

Definition at line 67 of file Vision.h.

8.181 Visiondefines.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define `MAX_COLORS` 11
- #define `COLOR_BACKGROUND` 0
- #define `COLOR_BLACK` 0
- #define `COLOR_BLUE` 1
- #define `COLOR_GREEN` 2
- #define `COLOR_ORANGE` 3
- #define `COLOR_BGREEN` 4
- #define `COLOR_PURPLE` 5
- #define `COLOR_RED` 6
- #define `COLOR_PINK` 7
- #define `COLOR_YELLOW` 8
- #define `COLOR_GRAY` 9
- #define `COLOR_SKIN` 10
- #define `NUM_VEVENTS` 6

8.181.1 Define Documentation

8.181.1.1 #define `COLOR_BACKGROUND` 0

Definition at line 6 of file Visiondefines.h.

8.181.1.2 #define `COLOR_BGREEN` 4

Definition at line 11 of file Visiondefines.h.

8.181.1.3 #define COLOR_BLACK 0

Definition at line 7 of file Visiondefines.h.

8.181.1.4 #define COLOR_BLUE 1

Definition at line 8 of file Visiondefines.h.

8.181.1.5 #define COLOR_GRAY 9

Definition at line 16 of file Visiondefines.h.

8.181.1.6 #define COLOR_GREEN 2

Definition at line 9 of file Visiondefines.h.

8.181.1.7 #define COLOR_ORANGE 3

Definition at line 10 of file Visiondefines.h.

8.181.1.8 #define COLOR_PINK 7

Definition at line 14 of file Visiondefines.h.

8.181.1.9 #define COLOR_PURPLE 5

Definition at line 12 of file Visiondefines.h.

8.181.1.10 #define COLOR_RED 6

Definition at line 13 of file Visiondefines.h.

8.181.1.11 #define COLOR_SKIN 10

Definition at line 17 of file Visiondefines.h.

8.181.1.12 #define COLOR_YELLOW 8

Definition at line 15 of file Visiondefines.h.

8.181.1.13 #define MAX_COLORS 11

Definition at line 4 of file Visiondefines.h.

8.181.1.14 #define NUM_VEVENTS 6

Definition at line 19 of file Visiondefines.h.

8.182 VisionEvent.h File Reference

8.182.1 Detailed Description

Provides information about objects recognized in the camera image.

Author:

alokl (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.7

State

Rel

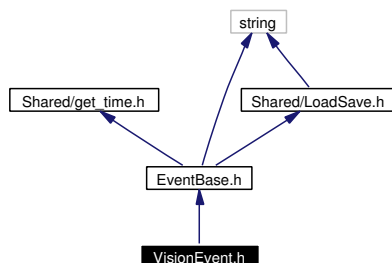
Date

2003/06/12 23:41:40

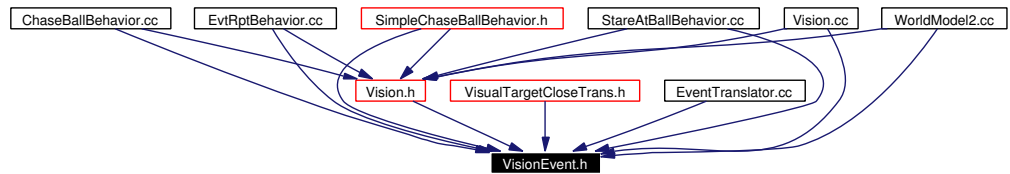
Definition in file [VisionEvent.h](#).

```
#include "EventBase.h"
```

Include dependency graph for VisionEvent.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [VisionEventNS](#)

Compounds

- class [VisionEvent](#)

Extends [EventBase](#) to also include location in the visual field and distance (though distance is not implimented yet).

8.183 VisionInterface.h File Reference

8.183.1 Detailed Description

Interfaces between CMVision and the [Vision](#) module.

Author:

CMU RoboSoccer 2001-2002 (Creator)

alokl (ported)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
=====

This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
=====

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed. In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

tss

Name

tekkotsu-1.4.1

Revision

1.4

State

Rel

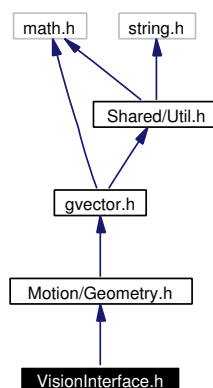
Date

2003/02/13 20:11:07

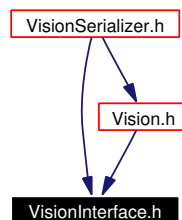
Definition in file [VisionInterface.h](#).

```
#include "Motion/Geometry.h"
```

Include dependency graph for VisionInterface.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [VisionInterface](#)

8.184 VisionSerializer.cc File Reference

8.184.1 Detailed Description

Implements [VisionSerializer](#), which encodes and transmits camera images.

Author:

CMU RoboSoccer 2001-2002 (Creator)

alokl (Ported)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
=====

This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
=====

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed. In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.5

State

Rel

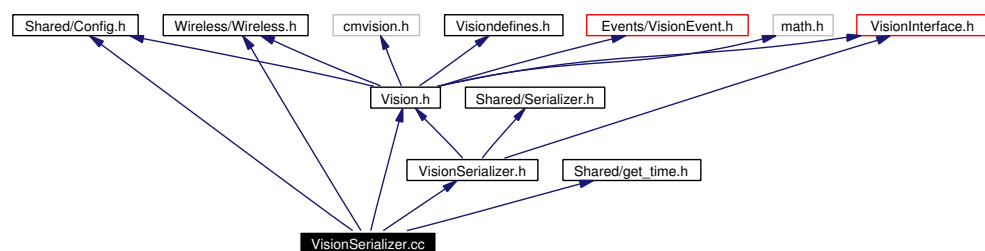
Date

2003/06/12 23:41:41

Definition in file [VisionSerializer.cc](#).

```
#include "Vision.h"  
#include "VisionSerializer.h"  
#include "Wireless/Wireless.h"  
#include "Shared/Config.h"  
#include "Shared/get_time.h"
```

Include dependency graph for VisionSerializer.cc:



8.185 VisionSerializer.h File Reference

8.185.1 Detailed Description

Describes [VisionSerializer](#), which encodes and transmits camera images.

Author:

CMU RoboSoccer 2001-2002 (Creator)

alokl (Ported)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
=====

This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
=====

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed. In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

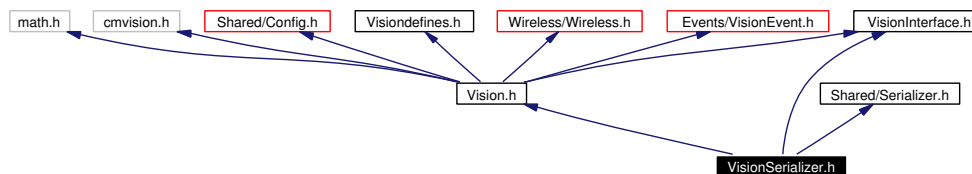
Date

2003/06/12 23:41:41

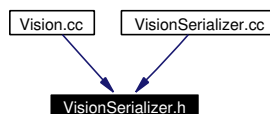
Definition in file [VisionSerializer.h](#).

```
#include "Vision.h"  
#include "VisionInterface.h"  
#include "Shared/Serializer.h"
```

Include dependency graph for VisionSerializer.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [VisionSerializer](#)
Encodes and transmits camera images.

8.186 VisualTargetCloseTrans.h File Reference

8.186.1 Detailed Description

Defines [VisualTargetCloseTrans](#), which causes a transition when a visual object is "close".

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.1

State

Rel

Date

2003/03/01 20:53:32

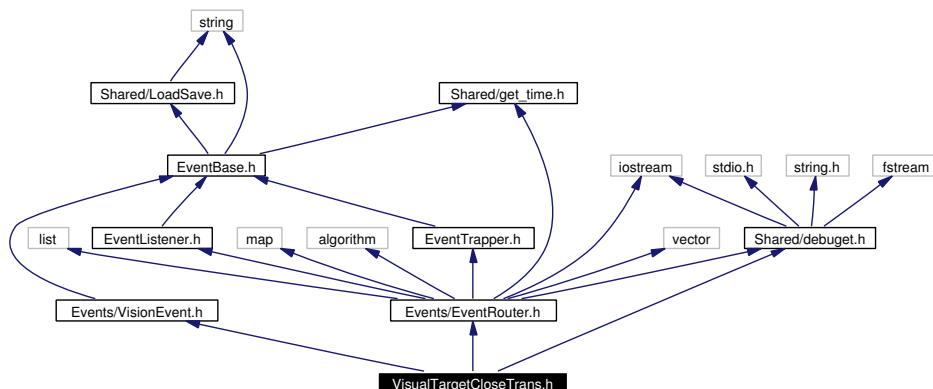
Definition in file [VisualTargetCloseTrans.h](#).

```
#include "Events/EventRouter.h"
```

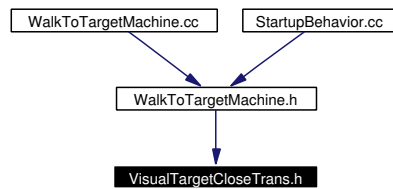
```
#include "Events/VisionEvent.h"
```

```
#include "Shared/debuget.h"
```

Include dependency graph for VisualTargetCloseTrans.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [VisualTargetCloseTrans](#)
causes a transition when a visual object is "close"

8.187 WalkControllerBehavior.cc File Reference

8.187.1 Detailed Description

Implements [WalkControllerBehavior](#), listens to mecha control commands coming in from the command port for remotely controlling the walk.

Author:

tss (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.1

State

Rel

Date

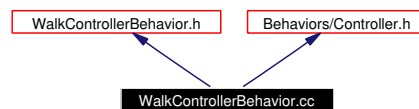
2003/07/07 01:00:08

Definition in file [WalkControllerBehavior.cc](#).

```
#include "WalkControllerBehavior.h"
```

```
#include "Behaviors/Controller.h"
```

Include dependency graph for WalkControllerBehavior.cc:



8.188 WalkControllerBehavior.h File Reference

8.188.1 Detailed Description

Describes [WalkControllerBehavior](#), listens to control commands coming in from the command port for remotely controlling the walk.

Author:

tss (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.1

State

Rel

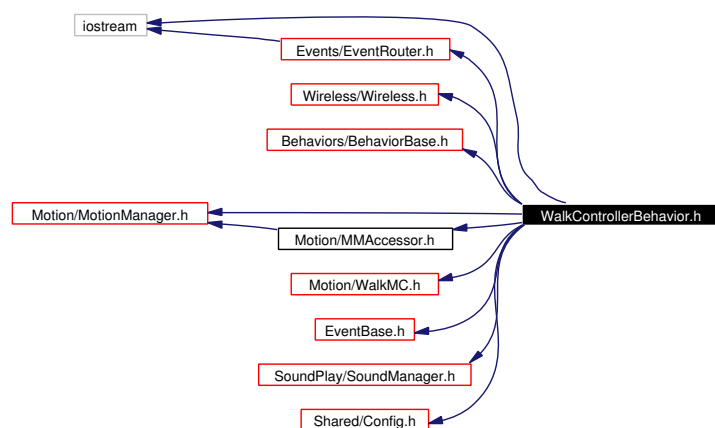
Date

2003/07/07 01:00:08

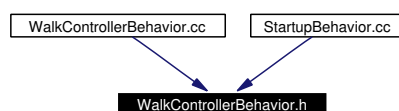
Definition in file [WalkControllerBehavior.h](#).

```
#include <iostream>
#include "Wireless/Wireless.h"
#include "Behaviors/BehaviorBase.h"
#include "Motion/MotionManager.h"
#include "Motion/WalkMC.h"
#include "Motion/MMAccessor.h"
#include "Events/EventRouter.h"
#include "Events/EventBase.h"
#include "SoundPlay/SoundManager.h"
#include "Shared/Config.h"
```

Include dependency graph for WalkControllerBehavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [WalkControllerBehavior](#)

Listens to control commands coming in from the command port for remotely controlling the walk.

8.189 WalkMC.cc File Reference

8.189.1 Detailed Description

Implements [WalkMC](#), a [MotionCommand](#) for walking around.

Author:

CMU RoboSoccer 2001-2002 (Creator)

ejt (ported)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
=====
```

```
-----
This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
-----
```

```
-----
Additionally licensed to Sony Corporation under the following terms:
```

```
-----
This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed. In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.13

State

Rel

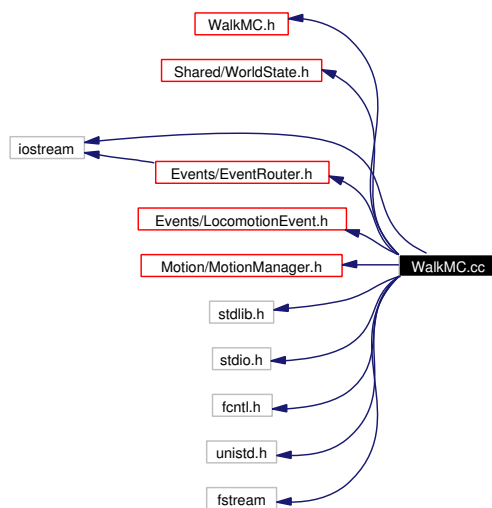
Date

2003/05/01 17:19:29

Definition in file [WalkMC.cc](#).

```
#include "WalkMC.h"
#include "Shared/WorldState.h"
#include "Events/EventRouter.h"
#include "Events/LocomotionEvent.h"
#include "Motion/MotionManager.h"
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <fcntl.h>
#include <unistd.h>
#include <fstream>
```

Include dependency graph for WalkMC.cc:



Defines

- #define [BOUND_MOTION](#)

Functions

- unsigned int [checksum](#) (const char *data, int num)

computes a file checksum

- `template<class data> int read_file (data *arr, int num, const char *filename)`
reads a set of walk parameters from a file
- `template<class data> int save_file (data *arr, int num, const char *filename)`
saves the current walk parameters to a file

8.189.2 Define Documentation

8.189.2.1 `#define BOUND_MOTION`

Definition at line 48 of file WalkMC.cc.

8.189.3 Function Documentation

8.189.3.1 `unsigned int checksum (const char * data, int num)`

computes a file checksum

Definition at line 237 of file WalkMC.cc.

8.189.3.2 `template<class data> int read_file (data * arr, int num, const char * filename)`

reads a set of walk parameters from a file

Definition at line 252 of file WalkMC.cc.

8.189.3.3 `template<class data> int save_file (data * arr, int num, const char * filename)`

saves the current walk parameters to a file

Definition at line 274 of file WalkMC.cc.

8.190 WalkMC.h File Reference

8.190.1 Detailed Description

Describes [WalkMC](#), a [MotionCommand](#) for walking around.

Author:

CMU RoboSoccer 2001-2002 (Creator)
ejt (ported)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
=====

This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
=====

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed. In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.8

State

Rel

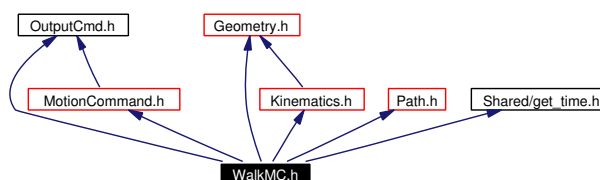
Date

2003/04/09 03:33:57

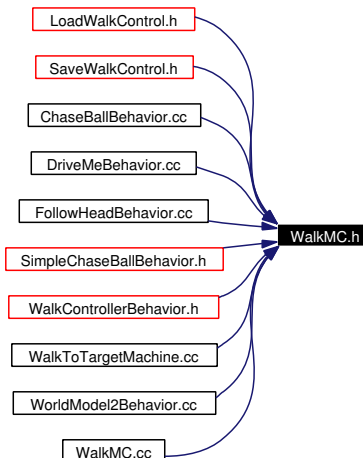
Definition in file [WalkMC.h](#).

```
#include "MotionCommand.h"
#include "Geometry.h"
#include "Kinematics.h"
#include "Path.h"
#include "Shared/get_time.h"
#include "OutputCmd.h"
```

Include dependency graph for WalkMC.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [WalkMC](#)

A nice walking class from Carnegie Mellon University's 2001 Robosoccer team, modified to fit this framework, see their [license](#).

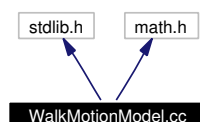
- struct [WalkMC.LegParam](#)
holds parameters about how to move each leg
- struct [WalkMC.LegWalkState](#)
holds state about each leg's path
- struct [WalkMC.WalkParam](#)
holds more general parameters about the walk

8.191 WalkMotionModel.cc File Reference

```
#include <stdlib.h>
```

```
#include <math.h>
```

Include dependency graph for WalkMotionModel.cc:



Functions

- void [WalkMotionModel](#) (double *x, double *y, double *theta, double [dx](#), double dy, double [da](#), unsigned int [time](#))

8.191.1 Function Documentation

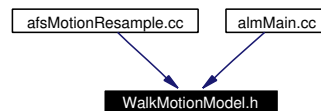
8.191.1.1 void [WalkMotionModel](#) (double *x, double *y, double *theta, double *dx*, double *dy*, double *da*, unsigned int *time*)

Definition at line 15 of file WalkMotionModel.cc.

References [da](#), [dx](#), and [time](#).

8.192 WalkMotionModel.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void [WalkMotionModel](#) (double *x, double *y, double *theta, double dx, double dy, double da, unsigned int time)

8.192.1 Function Documentation

8.192.1.1 void [WalkMotionModel](#) (double *x, double *y, double *theta, double dx, double dy, double da, unsigned int time)

Definition at line 15 of file WalkMotionModel.cc.

References da, dx, and time.

8.193 WalkToTargetMachine.cc File Reference

8.193.1 Detailed Description

Implements [WalkToTargetMachine](#), a state machine for walking towards a visual target.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1_4_1

Revision

1.2

State

Rel

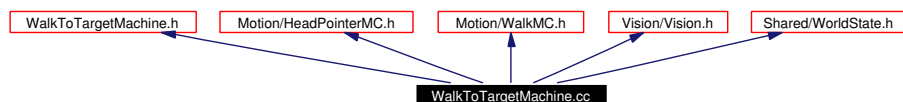
Date

2003/03/09 02:45:21

Definition in file [WalkToTargetMachine.cc](#).

```
#include "WalkToTargetMachine.h"
#include "Motion/HeadPointerMC.h"
#include "Motion/WalkMC.h"
#include "Vision/Vision.h"
#include "Shared/WorldState.h"
```

Include dependency graph for WalkToTargetMachine.cc:



Functions

- double [DtoR](#) (double deg)
Converts degrees to radians.

8.193.2 Function Documentation

8.193.2.1 double DtoR (double *deg*) [inline]

Converts degrees to radians.

Definition at line 8 of file WalkToTargetMachine.cc.

8.194 WalkToTargetMachine.h File Reference

8.194.1 Detailed Description

Describes [WalkToTargetMachine](#), a state machine for walking towards a visual target.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

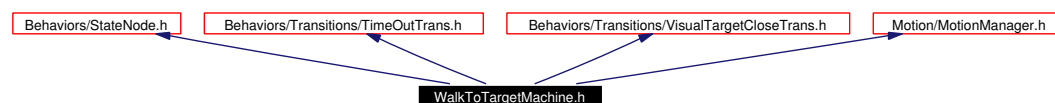
Date

2003/06/05 17:03:15

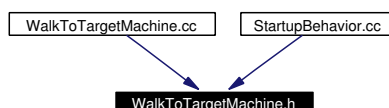
Definition in file [WalkToTargetMachine.h](#).

```
#include "Behaviors/StateNode.h"
#include "Behaviors/Transitions/TimeOutTrans.h"
#include "Behaviors/Transitions/VisualTargetCloseTrans.h"
#include "Motion/MotionManager.h"
```

Include dependency graph for WalkToTargetMachine.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [WalkToTargetMachine](#)
a state machine for walking towards a visual target

8.195 WAV.cc File Reference

8.195.1 Detailed Description

Allows you to load [WAV](#) files from the memory stick.

Author:

Sony (Creator)

This file is from the [SoundPlay](#) example from the Sony sample code, with a few if any modifications. Here's the license Sony provided with it:

Copyright 2002,2003 Sony Corporation

Permission to use, copy, modify, and redistribute this software for non-commercial use is hereby granted.

This software is provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of fitness for a particular purpose.

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.3

State

Rel

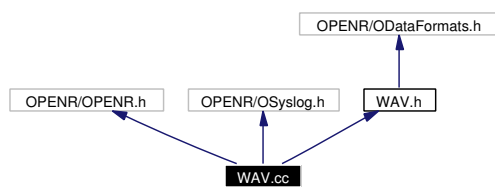
Date

2003/06/12 23:41:41

Definition in file [WAV.cc](#).

```
#include <OPENR/OPENR.h>
#include <OPENR/OSyslog.h>
#include "WAV.h"
```

Include dependency graph for WAV.cc:



8.196 WAV.h File Reference

8.196.1 Detailed Description

Allows you to load [WAV](#) files from the memory stick.

Author:

Sony (Creator)

This file is from the [SoundPlay](#) example from the Sony sample code, with a few if any modifications. Here's the license Sony provided with it:

Copyright 2002,2003 Sony Corporation

Permission to use, copy, modify, and redistribute this software for non-commercial use is hereby granted.

This software is provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of fitness for a particular purpose.

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.2

State

Rel

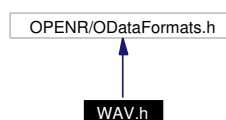
Date

2003/06/12 23:41:41

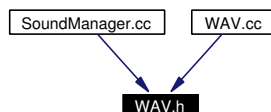
Definition in file [WAV.h](#).

```
#include <OPENR/ODataFormats.h>
```

Include dependency graph for WAV.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [WAV](#)

Enumerations

- enum [WAVEError](#) {
 [WAV_SUCCESS](#), [WAV_FAIL](#), [WAV_NOT_RIFF](#), [WAV_NOT_WAV](#),
 [WAV_FORMAT_NOT_SUPPORTED](#), [WAV_CHANNEL_NOT_SUPPORTED](#),
 [WAV_SAMPLINGRATE_NOT_SUPPORTED](#), [WAV_BITSPERSAMPLE_NOT_SUPPORTED](#),
 [WAV_SIZE_NOT_ENOUGH](#) }

8.196.2 Enumeration Type Documentation

8.196.2.1 enum [WAVEError](#)

Enumeration values:

[WAV_SUCCESS](#)
[WAV_FAIL](#)
[WAV_NOT_RIFF](#)
[WAV_NOT_WAV](#)
[WAV_FORMAT_NOT_SUPPORTED](#)
[WAV_CHANNEL_NOT_SUPPORTED](#)
[WAV_SAMPLINGRATE_NOT_SUPPORTED](#)
[WAV_BITSPERSAMPLE_NOT_SUPPORTED](#)
[WAV_SIZE_NOT_ENOUGH](#)

Definition at line 29 of file WAV.h.

8.197 Wireless.cc File Reference

8.197.1 Detailed Description

Interacts with the system to provide networking services.

Author:

alokl (Creator)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
-----

This software is distributed under the GNU General Public License,
version 2.  If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA.  This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
-----

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed.  In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.7

State

Rel

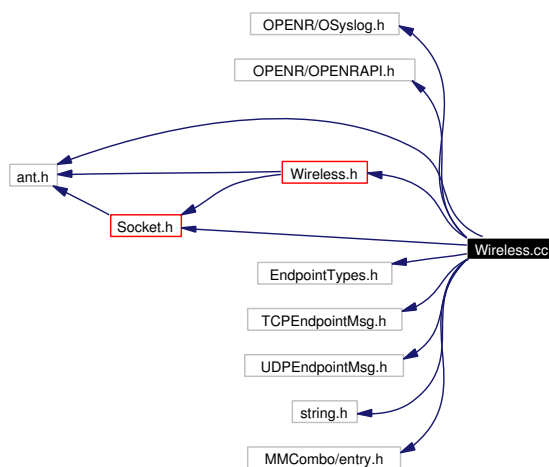
Date

2003/06/12 23:41:41

Definition in file [Wireless.cc](#).

```
#include <OPENR/OSyslog.h>
#include <OPENR/OPENRAPI.h>
#include <ant.h>
#include <EndpointTypes.h>
#include <TCPEndpointMsg.h>
#include <UDPEndpointMsg.h>
#include <string.h>
#include "Wireless.h"
#include "Socket.h"
#include "MMCombo/entry.h"
```

Include dependency graph for Wireless.cc:



Variables

- `Wireless * wireless = NULL`

the global wireless object - you'll want to make your function calls on this

8.197.2 Variable Documentation

8.197.2.1 `Wireless*` `wireless` = NULL

the global wireless object - you'll want to make your function calls on this

Definition at line 15 of file Wireless.cc.

8.198 Wireless.h File Reference

8.198.1 Detailed Description

Interacts with the system to provide networking services.

Author:

alokl (Creator)

```
=====
CMPack'02 Source Code Release for OPEN-R SDK v1.0
Copyright (C) 2002 Multirobot Lab [Project Head: Manuela Veloso]
School of Computer Science, Carnegie Mellon University
-----

This software is distributed under the GNU General Public License,
version 2. If you do not have a copy of this licence, visit
www.gnu.org, or write: Free Software Foundation, 59 Temple Place,
Suite 330 Boston, MA 02111-1307 USA. This program is distributed
in the hope that it will be useful, but WITHOUT ANY WARRANTY,
including MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
-----

Additionally licensed to Sony Corporation under the following terms:

This software is provided by the copyright holders AS IS and any
express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed. In no event shall authors be liable for
any direct, indirect, incidental, special, exemplary, or consequential
damages (including, but not limited to, procurement of substitute
goods or services; loss of use, data, or profits; or business
interruption) however caused and on any theory of liability, whether
in contract, strict liability, or tort (including negligence or
otherwise) arising in any way out of the use of this software, even if
advised of the possibility of such damage.
=====
```

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.12

State

Rel

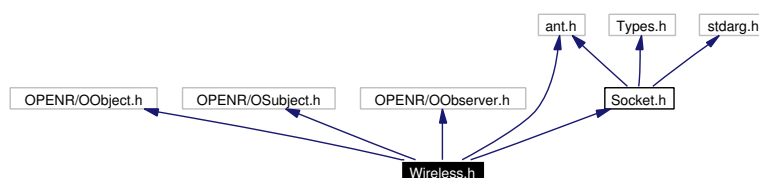
Date

2003/06/12 23:41:41

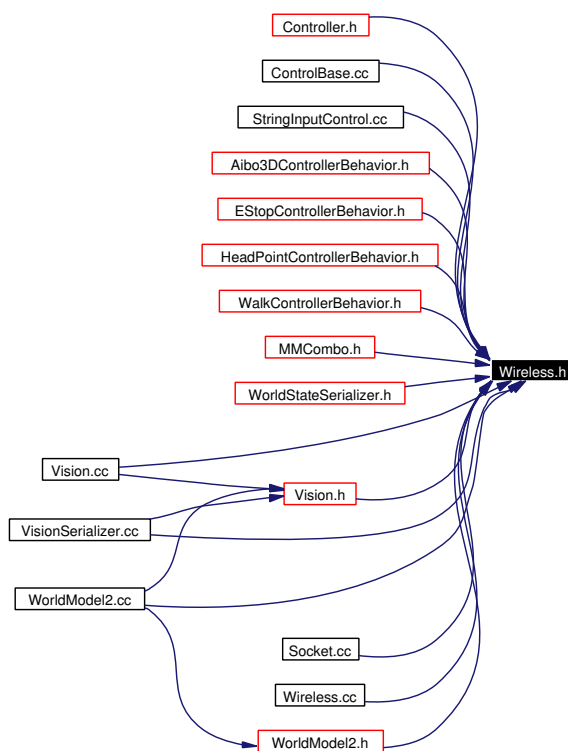
Definition in file [Wireless.h](#).

```
#include <OPENR/OObject.h>
#include <OPENR/OSubject.h>
#include <OPENR/OObserver.h>
#include <ant.h>
#include "Socket.h"
```

Include dependency graph for Wireless.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [Wireless](#)
Tekkotsu wireless class.

Variables

- [Wireless](#) * [wireless](#)
the global wireless object - you'll want to make your function calls on this

8.198.2 Variable Documentation

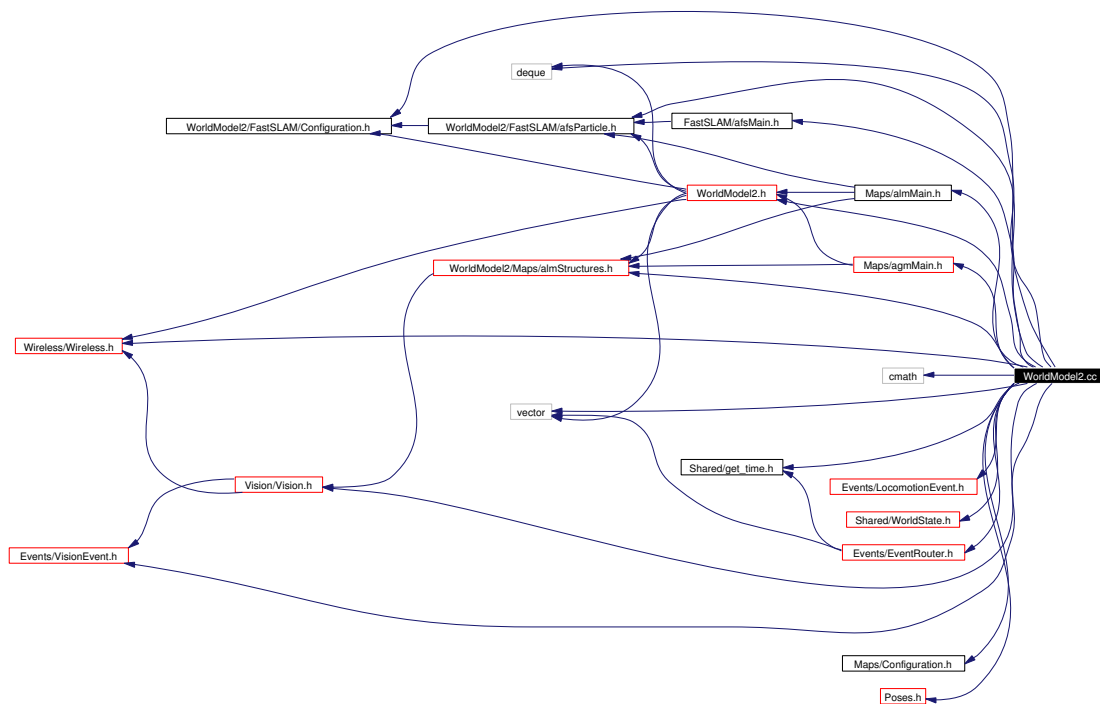
8.198.2.1 [Wireless](#)* [wireless](#)

the global wireless object - you'll want to make your function calls on this
Definition at line 135 of file Wireless.h.

8.199 WorldModel2.cc File Reference

```
#include "WorldModel2.h"
#include <cmath>
#include <vector>
#include <deque>
#include "Events/EventRouter.h"
#include "Events/VisionEvent.h"
#include "Events/LocomotionEvent.h"
#include "Shared/WorldState.h"
#include "Vision/Vision.h"
#include "Wireless/Wireless.h"
#include "FastSLAM/afsMain.h"
#include "Maps/almMain.h"
#include "Maps/agmMain.h"
#include "Maps/almStructures.h"
#include "FastSLAM/Configuration.h"
#include "FastSLAM/afsParticle.h"
#include "Maps/Configuration.h"
#include "Poses.h"
#include "Shared/get_time.h"
```

Include dependency graph for WorldModel2.cc:



Defines

- #define [UNLESS](#)(item) if(!(item))
- #define [TIMER_SID_GPA](#) 0
- #define [TIMER_SID_SRL](#) 1

Functions

- bool [aiiboIsErect](#) ()
- bool [aiiboStaresDeadAhead](#) ()
- bool [aiiboIsLevelHeaded](#) ()

8.199.1 Define Documentation

8.199.1.1 #define [TIMER_SID_GPA](#) 0

Definition at line 53 of file WorldModel2.cc.

8.199.1.2 #define TIMER_SID_SRL 1

Definition at line 54 of file WorldModel2.cc.

8.199.1.3 #define UNLESS(item) if(!(item))

Definition at line 47 of file WorldModel2.cc.

8.199.2 Function Documentation**8.199.2.1 bool aiboIsErect ()**

Determine whether Aibo is in an erect stature, allowing us to do measurements. TODO: replace with a real motion model

Definition at line 682 of file WorldModel2.cc.

References ERS210Info::ElevatorOffset, ERS210Info::KneeOffset, ERS210Info::LBkLegOffset, ERS210Info::LFrLegOffset, WorldState::outputs, ERS210Info::RBkLegOffset, ERS210Info::RFrLegOffset, ERS210Info::RotatorOffset, SP_LFR_JOINT, SP_LFR_KNEE, SP_LFR_SHLDR, SP_RFR_JOINT, SP_RFR_KNEE, SP_RFR_SHLDR, SP_TOLERANCE, and state.

8.199.2.2 bool aiboIsLevelHeaded ()

Determine whether Aibo is keeping its head level (i.e. no tilt or roll). Needed for FastSLAM at the moment.

Definition at line 735 of file WorldModel2.cc.

References DA_ROLL, DA_TILT, DA_TOLERANCE, ERS210Info::HeadOffset, WorldState::outputs, ERS210Info::RollOffset, state, and ERS210Info::TiltOffset.

8.199.2.3 bool aiboStaresDeadAhead ()

Determine whether Aibo is looking dead ahead. Needed for ground plane assumption, for the moment.

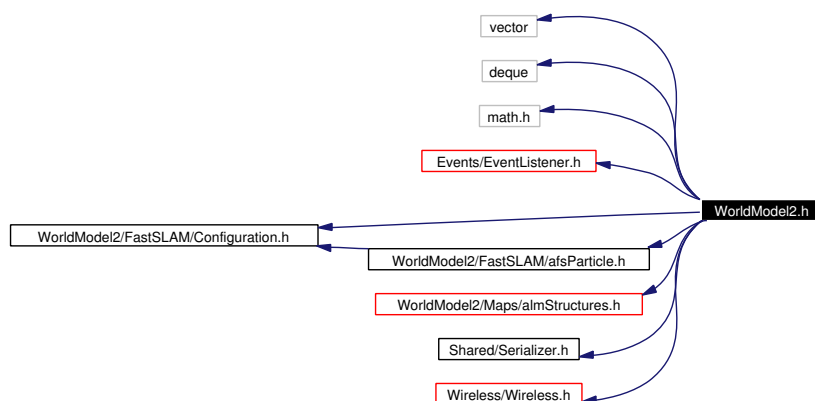
Definition at line 719 of file WorldModel2.cc.

References DA_PAN, DA_ROLL, DA_TILT, DA_TOLERANCE, ERS210Info::HeadOffset, WorldState::outputs, ERS210Info::PanOffset, ERS210Info::RollOffset, state, and ERS210Info::TiltOffset.

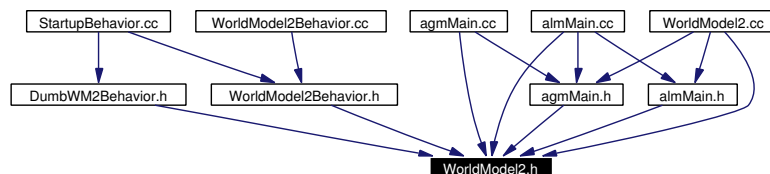
8.200 WorldModel2.h File Reference

```
#include <vector>
#include <deque>
#include <math.h>
#include "Events/EventListener.h"
#include "WorldModel2/FastSLAM/Configuration.h"
#include "WorldModel2/FastSLAM/afsParticle.h"
#include "WorldModel2/Maps/almStructures.h"
#include "Shared/Serializer.h"
#include "Wireless/Wireless.h"
```

Include dependency graph for WorldModel2.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [WM2Kludge](#)

Compounds

- struct [_FastSLAM_update](#)
- struct [MotionRequest](#)

Structure for containing motion requests.

- class [WorldModel2](#)

Tekkotsu's localization and mapping system (IN ACTIVE DEVELOPMENT).

Typedefs

- typedef [_FastSLAM_update](#) [FastSLAM_update](#)
- typedef std::vector< [MotionRequest](#) > [MRvector](#)

A vector for motion requests.

- typedef std::deque< [FastSLAM_update](#) > [FSUdeque](#)

A deque for FastSLAM updates.

Functions

- bool [aiboIsErect](#) ()
- bool [aiboStaresDeadAhead](#) ()
- bool [aiboIsLevelHeaded](#) ()

8.200.1 Typedef Documentation

8.200.1.1 typedef struct [_FastSLAM_update](#) [FastSLAM_update](#)

Structure for keeping track of FastSLAM system updates for eventual evaluation. See the documentation on the LazyFastSLAM kludge to see why we need to keep this data around at times.

8.200.1.2 typedef std::deque<[FastSLAM_update](#)> [FSUdeque](#)

A deque for FastSLAM updates.

Definition at line 104 of file WorldModel2.h.

8.200.1.3 typedef std::vector<MotionRequest> MRvector

A vector for motion requests.

Definition at line 101 of file WorldModel2.h.

8.200.2 Function Documentation

8.200.2.1 bool aiboIsErect ()

Determine whether Aibo is in an erect stature, allowing us to do measurements. TODO: replace with a real motion model

Definition at line 682 of file WorldModel2.cc.

References ERS210Info::ElevatorOffset, ERS210Info::KneeOffset, ERS210Info::LBkLegOffset, ERS210Info::LFrLegOffset, WorldState::outputs, ERS210Info::RBkLegOffset, ERS210Info::RFrLegOffset, ERS210Info::RotatorOffset, SP_LFR_JOINT, SP_LFR_KNEE, SP_LFR_SHLDR, SP_RFR_JOINT, SP_RFR_KNEE, SP_RFR_SHLDR, SP_TOLERANCE, and state.

8.200.2.2 bool aiboIsLevelHeaded ()

Determine whether Aibo is keeping its head level (i.e. no tilt or roll). Needed for FastSLAM at the moment.

Definition at line 735 of file WorldModel2.cc.

References DA_ROLL, DA_TILT, DA_TOLERANCE, ERS210Info::HeadOffset, WorldState::outputs, ERS210Info::RollOffset, state, and ERS210Info::TiltOffset.

8.200.2.3 bool aiboStaresDeadAhead ()

Determine whether Aibo is looking dead ahead. Needed for ground plane assumption, for the moment.

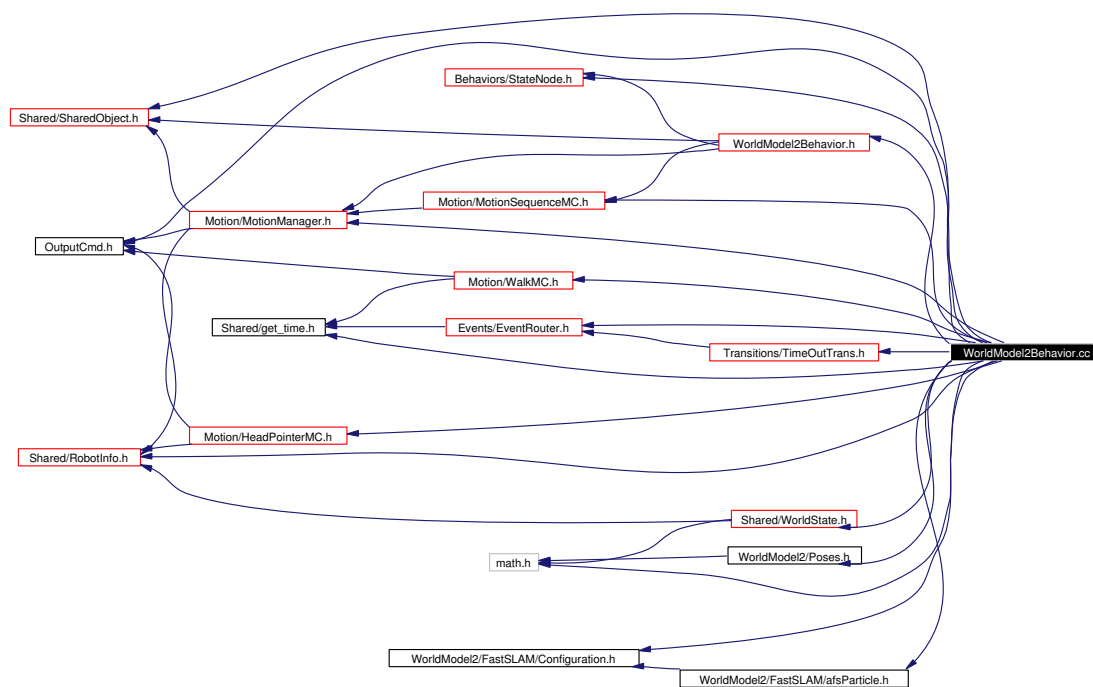
Definition at line 719 of file WorldModel2.cc.

References DA_PAN, DA_ROLL, DA_TILT, DA_TOLERANCE, ERS210Info::HeadOffset, WorldState::outputs, ERS210Info::PanOffset, ERS210Info::RollOffset, state, and ERS210Info::TiltOffset.

8.201 WorldModel2Behavior.cc File Reference

```
#include "WorldModel2Behavior.h"
#include "StateNode.h"
#include "Transitions/TimeOutTrans.h"
#include "Motion/MotionManager.h"
#include "Motion/WalkMC.h"
#include "Motion/HeadPointerMC.h"
#include "Motion/MotionSequenceMC.h"
#include "Motion/OutputCmd.h"
#include "Shared/SharedObject.h"
#include "Events/EventRouter.h"
#include "Shared/WorldState.h"
#include "Shared/RobotInfo.h"
#include "Shared/get_time.h"
#include "WorldModel2/FastSLAM/afsParticle.h"
#include "WorldModel2/FastSLAM/Configuration.h"
#include "WorldModel2/Poses.h"
#include <math.h>
```

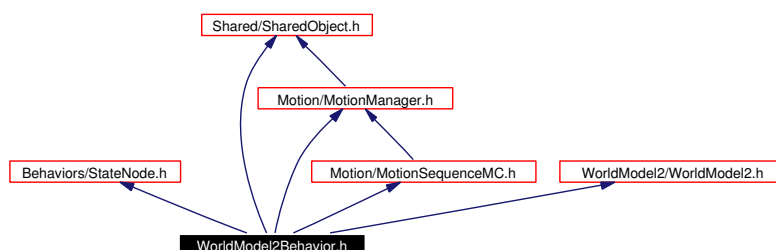
Include dependency graph for WorldModel2Behavior.cc:



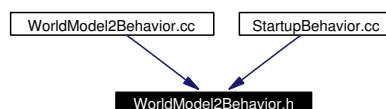
8.202 WorldModel2Behavior.h File Reference

```
#include "Behaviors/StateNode.h"
#include "Motion/MotionManager.h"
#include "WorldModel2/WorldModel2.h"
#include "Motion/MotionSequenceMC.h"
#include "Shared/SharedObject.h"
```

Include dependency graph for WorldModel2Behavior.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [WorldModel2Behavior](#)
Implements a stateful behavior specifically designed to feed data to the second World-Model.
- struct [WorldModel2Behavior.GawkNode](#)
This one wags the head around.
- struct [WorldModel2Behavior.WaitNode](#)
This one lets the AIBO have a reflective pause.
- struct [WorldModel2Behavior.WalkNode](#)

This one does a random walk.

8.203 WorldState.cc File Reference

8.203.1 Detailed Description

Implements [WorldState](#), maintains information about the robot's environment, namely sensors and power status.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.15

State

Rel

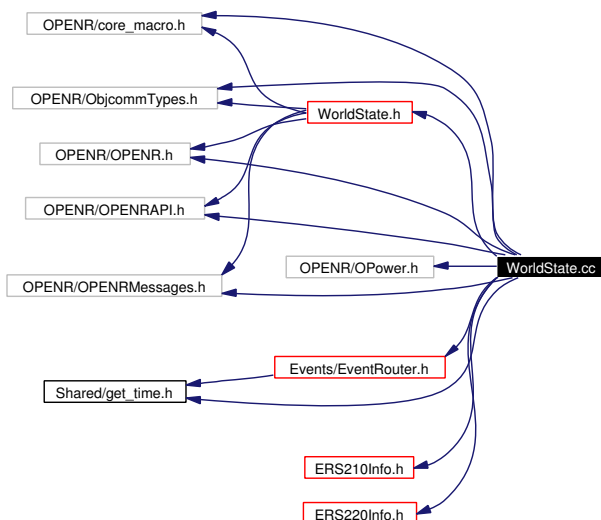
Date

2003/06/12 04:16:44

Definition in file [WorldState.cc](#).

```
#include <OPENR/core_macro.h>
#include <OPENR/ObjcommTypes.h>
#include <OPENR/OPENR.h>
#include <OPENR/OPENRAPI.h>
#include <OPENR/OPENRMessages.h>
#include <OPENR/OPower.h>
#include "WorldState.h"
#include "Shared/get_time.h"
#include "Events/EventRouter.h"
#include "ERS210Info.h"
#include "ERS220Info.h"
```

Include dependency graph for WorldState.cc:



Defines

- #define **GETD**(cpc) (((float)sensor.GetData(cpc) → frame[0].value) / 1.0E6f)
value from OPEN-R, converted from micro in int to base in float
- #define **GETB**(cpc) ((bool)sensor.GetData(cpc) → frame[0].value)
value from OPEN-R, as bool
- #define **GETSIG**(cpc) ((word)sensor.GetData(cpc) → frame[0].signal)
signal from OPEN-R as word
- #define **GETDUTY**(cpc) ((float)((OJointValue*)&sensor.GetData(cpc) → frame[0]) → pwmDuty/512.0f)
duty cycle from OPEN-R as float; -1 (full reverse) to 0 (idle) to 1 (full forward)

Variables

- **WorldState** * state = NULL
the global state object, is a shared memory region, created by MainObject

8.203.2 Define Documentation

8.203.2.1 `#define GETB(cpc) ((bool)sensor.GetData(cpc) → frame[0].value)`

value from OPEN-R, as bool

Definition at line 17 of file WorldState.cc.

8.203.2.2 `#define GETD(cpc) (((float)sensor.GetData(cpc) → frame[0].value) / 1.0E6f)`

value from OPEN-R, converted from micro in int to base in float

Definition at line 16 of file WorldState.cc.

8.203.2.3 `#define GETDUTY(cpc) ((float)((OJointValue*)&sensor.GetData(cpc) → frame[0]) → pwmDuty/512.0f)`

duty cycle from OPEN-R as float; -1 (full reverse) to 0 (idle) to 1 (full forward)

Definition at line 19 of file WorldState.cc.

8.203.2.4 `#define GETSIG(cpc) ((word)sensor.GetData(cpc) → frame[0].signal)`

signal from OPEN-R as word

Definition at line 18 of file WorldState.cc.

8.203.3 Variable Documentation

8.203.3.1 `WorldState* state = NULL`

the global state object, is a shared memory region, created by MainObject

Definition at line 24 of file WorldState.cc.

8.204 WorldState.h File Reference

8.204.1 Detailed Description

Describes [WorldState](#), maintains information about the robot's environment, namely sensors and power status.

Author:

ejt (Creator)

Author

ejt

Name

tekkotsu-1.4.1

Revision

1.19

State

Exp

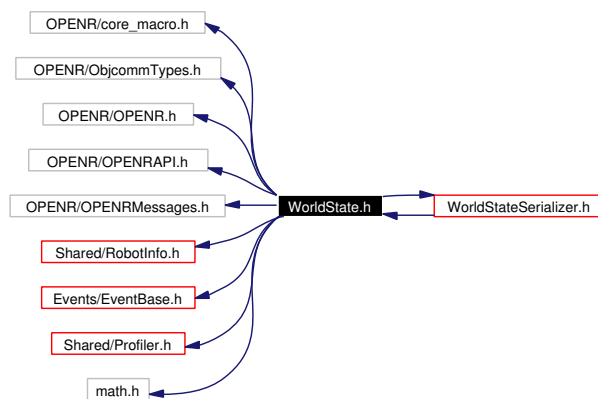
Date

2003/07/27 19:11:27

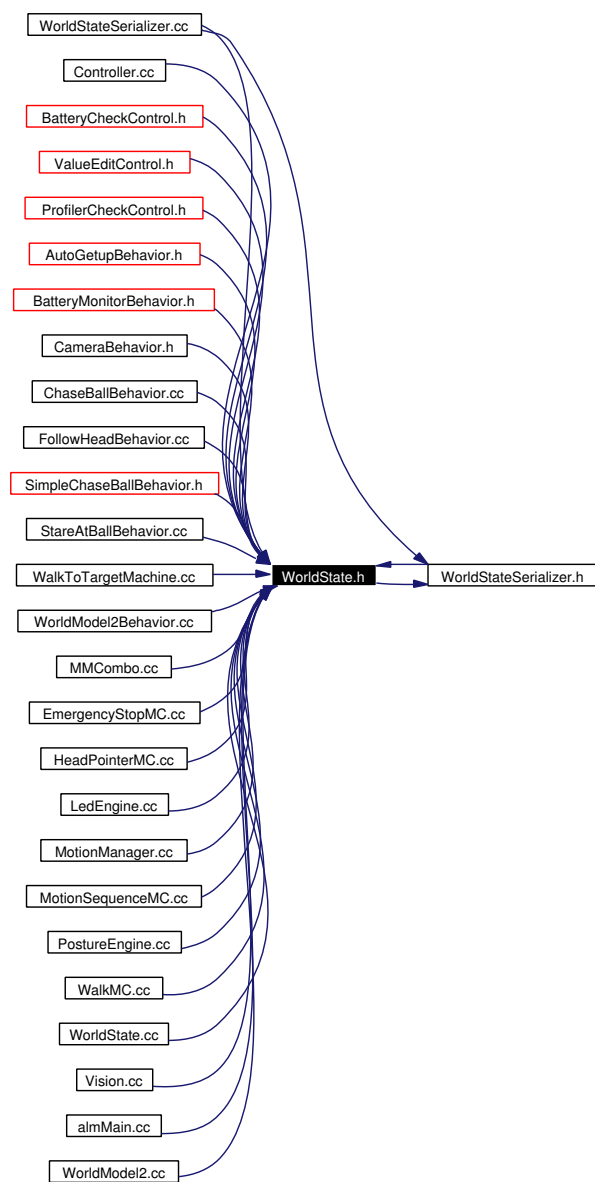
Definition in file [WorldState.h](#).

```
#include <OPENR/core_macro.h>
#include <OPENR/ObjcommTypes.h>
#include <OPENR/OPENR.h>
#include <OPENR/OPENRAPI.h>
#include <OPENR/OPENRMessages.h>
#include "Shared/RobotInfo.h"
#include "Events/EventBase.h"
#include "Shared/Profiler.h"
#include <math.h>
#include "WorldStateSerializer.h"
```

Include dependency graph for WorldState.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [ButtonSourceID](#)
- namespace [PowerSourceID](#)
- namespace [SensorSourceID](#)

Compounds

- class [WorldState](#)

The state of the robot and its environment.

Variables

- [WorldState](#) * [state](#)

the global state object, is a shared memory region, created by MainObject

8.204.2 Variable Documentation

8.204.2.1 [WorldState](#)* [state](#)

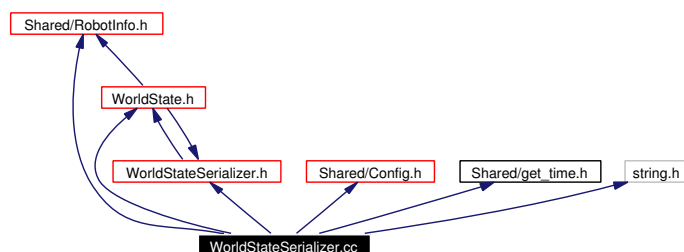
the global state object, is a shared memory region, created by MainObject

Definition at line 210 of file WorldState.h.

8.205 WorldStateSerializer.cc File Reference

```
#include "WorldState.h"
#include "WorldStateSerializer.h"
#include "Shared/Config.h"
#include "Shared/get_time.h"
#include <string.h>
#include "RobotInfo.h"
```

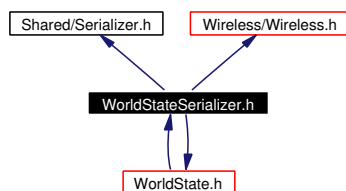
Include dependency graph for WorldStateSerializer.cc:



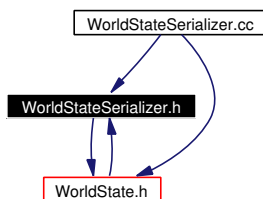
8.206 WorldStateSerializer.h File Reference

```
#include "WorldState.h"
#include "Shared/Serializer.h"
#include "Wireless/Wireless.h"
```

Include dependency graph for WorldStateSerializer.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class [WorldStateSerializer](#)

Chapter 9

Tekkotsu Page Documentation

9.1 Todo List

Class [BodyPosition](#) get rid of this

Class [ControlBase](#) ControlBase's should use [ReferenceCounter](#) so memory management is not an issue

Member [EventManager::processTimers\(\)](#) handle recursive calls

Member [HeadPointerMC::convFromBodyRelative\(RobotInfo::TPROffset_t i, double v, CoordFrame_t mode\) const](#)
this is perhaps a bit amateurish - could be more accurate

Member [HeadPointerMC::convToBodyRelative\(RobotInfo::TPROffset_t i, double v, CoordFrame_t mode\) const](#)
this is perhaps a bit amateurish - could be more accurate

Member [LedMC::isAlive\(\)](#) let's make this smarter so you can flash the LED's and have it autoprune

Member [MotionCommand::interpolate\(double a, double b, double x\)](#) - replace with a more fancy spline based thing?

Member [MotionCommand::interpolate\(float a, float b, float x\)](#) - replace with a more fancy spline based thing?

Member `MotionManager::setOutput(const MotionCommand *caller, unsigned int output, const OutputPI`
should be able to set background pid

Member `MutexLock::lock(int id)` - I'd like to not use a loop here

Member `PostureEngine::PostureEngine(const char *filename)` might want to make a library of common positions so they don't have to be loaded repeatedly from memstick

Member `PostureEngine::avgdiff(const PostureEngine &pe) const` create a version which looks at weights? This doesn't use them.

Member `PostureEngine::diff(const PostureEngine &pe) const` create a version which looks at weights? This doesn't use them.

Member `PostureEngine::maxdiff(const PostureEngine &pe) const` create a version which looks at weights? This doesn't use them.

Member `ProfilerCheckControl::activate(MotionManager::MC_ID display, Socket *)`
make the leds flash

Class `SoundManager` Volume control, variable playback speed, support more wav file formats (all go together)
Add functions to hand out regions to be filled out to avoid copying into the buffer.

Member `SoundManager::LoadFile(const char *name)` this does one more copy than it really needs to

Member `TimeET::Set()` not getting timeofday on OPEN-R, is time since boot instead...

Class `ValueEditControl< T >` needs some work to really be useful again

Member `ValueSetControl::activate(MotionManager::MC_ID display)` make the leds flash

Member **VisualTargetCloseTrans::processEvent**(const **EventBase** &e) need to activate if it's "close"

Member **WalkMC::load**(const char *pfile) use **LoadSave**)

Member **WalkMC::save**(const char *pfile) const use **LoadSave**)

Member **WorldState::read**(OSensorFrameVectorData &sensor, **EventRouter** *er) change to use most recent instead of oldest - is a buffer!

File **LoadSave.h** detect appropriate byte ordering for other platforms

File **PostureEngine.h** write a binary version of Load/Save commands for faster access

9.2 Test List

Member `MMAccessor::checkout()` can this be a `dynamic_cast`?

Member `MotionManager::removeMotion(MC_ID mcid)` do i do this right, or does it leak memory?

Index

- ~Aibo3DControllerBehavior
 - Aibo3DControllerBehavior, [160](#)
- ~AutoGetupBehavior
 - AutoGetupBehavior, [172](#)
- ~BanditMachine
 - BanditMachine, [177](#)
- ~BatteryCheckControl
 - BatteryCheckControl, [214](#)
- ~BatteryMonitorBehavior
 - BatteryMonitorBehavior, [219](#)
- ~BehaviorActivatorControl
 - BehaviorActivatorControl, [226](#)
- ~BehaviorBase
 - BehaviorBase, [230](#)
- ~BehaviorGroup
 - BehaviorSwitchControl-
Base::BehaviorGroup, [250](#)
- ~BehaviorSwitchActivatorControl
 - BehaviorSwitchActivatorControl, [235](#)
- ~BehaviorSwitchControl
 - BehaviorSwitchControl, [240](#)
- ~BehaviorSwitchControlBase
 - BehaviorSwitchControlBase, [246](#)
- ~CameraBehavior
 - CameraBehavior, [256](#)
- ~ChaseBallBehavior
 - ChaseBallBehavior, [262](#)
- ~Config
 - Config, [272](#)
- ~ControlBase
 - ControlBase, [297](#)
- ~Controller
 - Controller, [313](#)
- ~DriveMeBehavior
 - DriveMeBehavior, [329](#)
- ~DumbWM2Behavior
 - DumbWM2Behavior, [333](#)
- ~DumpFileControl
 - DumpFileControl, [337](#)
- ~DynamicMotionSequence
 - DynamicMotionSequence, [340](#)
- ~EStopControllerBehavior
 - EStopControllerBehavior, [354](#)
- ~EmergencyStopMC
 - EmergencyStopMC, [347](#)
- ~EventBase
 - EventBase, [363](#)
- ~EventListener
 - EventListener, [372](#)
- ~EventRouter
 - EventRouter, [383](#)
- ~EventTrapper
 - EventTrapper, [410](#)
- ~EvtRptBehavior
 - EvtRptBehavior, [413](#)
- ~FollowHeadBehavior
 - FollowHeadBehavior, [426](#)
- ~FreeMemReportControl
 - FreeMemReportControl, [432](#)
- ~HeadLevelBehavior
 - HeadLevelBehavior, [437](#)
- ~HeadPointControllerBehavior
 - HeadPointControllerBehavior, [442](#)
- ~HeadPointerMC
 - HeadPointerMC, [450](#)
- ~LedEngine
 - LedEngine, [481](#)
- ~LedMC
 - LedMC, [493](#)
- ~LoadPostureControl
 - LoadPostureControl, [512](#)

- ~LoadSave
 - LoadSave, [520](#)
- ~LoadWalkControl
 - LoadWalkControl, [538](#)
- ~LockScope
 - LockScope, [539](#)
- ~MCValueEditControl
 - MCValueEditControl, [548](#)
- ~MMAccessor
 - MMAccessor, [552](#)
- ~MMCombo
 - MMCombo, [561](#)
- ~MotionCommand
 - MotionCommand, [574](#)
- ~MotionManagerMsg
 - MotionManagerMsg, [610](#)
- ~MotionSequence
 - MotionSequence, [621](#)
- ~MotionSequenceMC
 - MotionSequenceMC, [637](#)
- ~PIDMC
 - PIDMC, [673](#)
- ~PlaySoundControl
 - PlaySoundControl, [678](#)
- ~PostureEngine
 - PostureEngine, [682](#)
- ~PostureMC
 - PostureMC, [692](#)
- ~PressNode
 - BanditMachine::PressNode, [184](#)
- ~ProfilerCheckControl
 - ProfilerCheckControl, [719](#)
- ~ReferenceCounter
 - ReferenceCounter, [726](#)
- ~RemoteControllerMC
 - RemoteControllerMC, [729](#)
- ~RemoteProcess
 - RemoteProcess, [733](#)
- ~SaveWalkControl
 - SaveWalkControl, [743](#)
- ~SimpleChaseBallBehavior
 - SimpleChaseBallBehavior, [764](#)
- ~Socket
 - Socket, [774](#)
- ~SoundManagerMsg
 - SoundManagerMsg, [807](#)
- ~SoundPlay
 - SoundPlay, [813](#)
- ~StareAtBallBehavior
 - StareAtBallBehavior, [827](#)
- ~StartupBehavior
 - StartupBehavior, [831](#)
- ~StateNode
 - StateNode, [838](#)
- ~TailWagMC
 - TailWagMC, [851](#)
- ~Timer
 - Profiler::Timer, [716](#)
- ~Transition
 - Transition, [876](#)
- ~ValueEditControl
 - ValueEditControl, [881](#)
- ~ValueSetControl
 - ValueSetControl, [888](#)
- ~Vision
 - Vision, [906](#)
- ~WAV
 - WAV, [971](#)
- ~WaitNode
 - BanditMachine::WaitNode, [187](#)
- ~WalkControllerBehavior
 - WalkControllerBehavior, [938](#)
- ~WalkNode
 - WorldModel2Behavior::WalkNode, [1008](#)
- ~Wireless
 - Wireless, [977](#)
- ~WorldModel2
 - WorldModel2, [988](#)
- ~WorldModel2Behavior
 - WorldModel2Behavior, [997](#)
- ~basic_netbuf
 - basic_netbuf, [203](#)
- _FastSLAM_update
 - LANDMARK, [152](#)
 - MOTION, [152](#)
- _FastSLAM_update, [152](#)
- _FastSLAM_update
 - da, [153](#)
 - dx, [153](#)
 - dy, [153](#)
 - time, [153](#)

- type, 153
- ._MMaccID
 - MotionManager, 596
- ._RI_RAD_FLAG
 - ERS210Info.h, 1129
 - ERS220Info.h, 1131
 - ERS2xxInfo.h, 1133
- ._afsLandmarkLoc
 - PRIMED, 140
 - PRIMING, 140
- ._afsLandmarkLoc, 139
- ._afsLandmarkLoc
 - mean, 140
 - priming, 140
 - state, 140
 - variance, 140
 - x, 140
 - xy, 140
 - y, 141
- ._afsLastObservation, 142
- ._afsLastObservation
 - empty, 142
 - theta, 142
 - x, 142
 - y, 143
- ._afsParticle, 144
- ._afsParticle
 - gotweight, 144
 - landmarks, 144
 - pose, 144
 - weight, 144
- ._afsPose, 146
- ._afsPose
 - theta, 146
 - x, 146
 - y, 146
- ._afsRRP, 148
- ._afsRRP
 - distance, 148
 - theta, 148
- ._afsXY, 149
- ._afsXY
 - x, 149
 - y, 149
- ._cenX
 - VisionEvent, 923
- ._cenY
 - VisionEvent, 923
- ._distance
 - VisionEvent, 923
- ._dm_cell, 150
 - cluster, 150
 - color, 150
 - confidence, 150
 - depth, 151
- ._extractFilename
 - debuget.h, 1112
- ._hm_cell, 154
 - cluster, 154
 - color, 154
 - confidence, 154
 - height, 155
 - trav, 155
- ._id
 - Profiler::Timer, 717
- ._parent
 - Profiler::Timer, 717
- ._prof
 - Profiler::Timer, 717
- ._property
 - VisionEvent, 923
- ._t
 - Profiler::Timer, 717
- ._text
 - TextMsgEvent, 858
- ._token
 - TextMsgEvent, 858
- a
 - HermiteSplineSegment, 461
 - LocomotionEvent, 544
 - NonUniformHermiteSplineSegment, 653
- abs_t
 - mathutils, 121
- accID_t
 - MotionManager, 586
- activate
 - BatteryCheckControl, 214
 - BehaviorActivatorControl, 227
 - BehaviorSwitchActivatorControl, 236

- BehaviorSwitchControlBase, [247](#)
- ControlBase, [298](#)
- Controller, [313](#)
- FileBrowserControl, [420](#)
- HelpControl, [458](#)
- NullControl, [655](#)
- ProfilerCheckControl, [720](#)
- RebootControl, [724](#)
- SaveWalkControl, [743](#)
- ShutdownControl, [762](#)
- StringInputControl, [846](#)
- Transition, [876](#)
- ValueEditControl, [881](#)
- ValueSetControl, [888](#)
- activateETID
 - EventBase, [362](#)
- active
 - EmergencyStopMC, [350](#)
 - HeadPointerMC, [455](#)
 - RemoteControllerMC, [730](#)
 - SoundPlay, [816](#)
 - TailWagMC, [854](#)
- activeBack
 - ListMemBuf, [507](#)
- activeBegin
 - ListMemBuf, [507](#)
- add
 - SoundManagerMsg, [807](#)
 - SplinePath, [824](#)
- addCopy
 - ValueEditControl, [881](#)
- addListener
 - EventRouter, [384](#)
- addMapping
 - EventRouter::EventMapper, [396](#)
- addMotion
 - MotionManager, [587](#), [588](#)
 - MotionManagerMsg, [610](#)
- addNode
 - StateNode, [838](#)
- AddReference
 - ReferenceCounter, [726](#)
- addRunLevel
 - MMCombo, [561](#)
- addTimer
 - EventRouter, [384](#), [385](#)
- addToHistHorizStrip
 - Vision, [907](#)
- addToHistVertStrip
 - Vision, [907](#)
- addTransition
 - StateNode, [838](#)
- addTrapper
 - EventRouter, [385](#), [386](#)
- AFS_COVARIANCE_FUDGE
 - FastSLAM/Configuration.h, [1094](#)
- AFS_MAX_TRIANG_ANGLE
 - FastSLAM/Configuration.h, [1094](#)
- AFS_MEASURE_VARIANCE
 - FastSLAM/Configuration.h, [1094](#)
- AFS_MIN_TRIANG_ANGLE
 - FastSLAM/Configuration.h, [1095](#)
- AFS_NUM_LANDMARKS
 - FastSLAM/Configuration.h, [1095](#)
- AFS_NUM_PARTICLES
 - FastSLAM/Configuration.h, [1095](#)
- AFS_PERTURB_DA_MEAN
 - FastSLAM/Configuration.h, [1095](#)
- AFS_PERTURB_DA_VARIANCE
 - FastSLAM/Configuration.h, [1095](#)
- AFS_PERTURB_DX_MEAN
 - FastSLAM/Configuration.h, [1095](#)
- AFS_PERTURB_DX_VARIANCE
 - FastSLAM/Configuration.h, [1095](#)
- AFS_PERTURB_DY_MEAN
 - FastSLAM/Configuration.h, [1095](#)
- AFS_PERTURB_DY_VARIANCE
 - FastSLAM/Configuration.h, [1095](#)
- AFS_VARIANCE_MULTIPLIER
 - FastSLAM/Configuration.h, [1095](#)
- afsApplyBestPose
 - afsFindBestPose.cc, [1024](#)
 - afsFindBestPose.h, [1027](#)
- afsCertainty
 - afsMain.cc, [1030](#)
 - afsMain.h, [1032](#)
- afsDistribute
 - afsMain.cc, [1030](#)
 - afsMain.h, [1033](#)
- afsFindBestPose
 - afsFindBestPose.cc, [1024](#)
 - afsFindBestPose.h, [1027](#)

- afsFindBestPose.cc, [1023](#)
- afsFindBestPose.cc
 - afsApplyBestPose, [1024](#)
 - afsFindBestPose, [1024](#)
 - afsGuessState, [1024](#)
 - afsParticleError, [1024](#)
- afsFindBestPose.h, [1026](#)
- afsFindBestPose.h
 - afsApplyBestPose, [1027](#)
 - afsFindBestPose, [1027](#)
 - afsGuessState, [1027](#)
 - afsParticleError, [1028](#)
 - afsRRP, [1027](#)
 - afsXY, [1027](#)
- afsGuessState
 - afsFindBestPose.cc, [1024](#)
 - afsFindBestPose.h, [1027](#)
- afsInit
 - afsMain.cc, [1030](#)
 - afsMain.h, [1033](#)
- afsInvadeFSData
 - afsMain.cc, [1030](#)
 - afsMain.h, [1033](#)
- afsLandmarkLoc
 - afsParticle.h, [1042](#)
- afsLastObservation
 - afsParticle.h, [1042](#)
- afsMain.cc, [1029](#)
- afsMain.cc
 - afsCertainty, [1030](#)
 - afsDistribute, [1030](#)
 - afsInit, [1030](#)
 - afsInvadeFSData, [1030](#)
 - afsMeasurement, [1030](#)
 - afsMotion, [1030](#)
 - afsResample, [1030](#)
 - afsSetLandmark, [1031](#)
 - afsWhatsUp, [1031](#)
 - newParticles, [1031](#)
 - Particles, [1031](#)
 - ParticleSets, [1031](#)
 - Weights, [1031](#)
- afsMain.h, [1032](#)
- afsMain.h
 - afsCertainty, [1032](#)
 - afsDistribute, [1033](#)
- afsInit, [1033](#)
- afsInvadeFSData, [1033](#)
- afsMeasurement, [1033](#)
- afsMotion, [1033](#)
- afsResample, [1033](#)
- afsSetLandmark, [1033](#)
- afsWhatsUp, [1034](#)
- afsMeasurement
 - afsMain.cc, [1030](#)
 - afsMain.h, [1033](#)
- afsMeasurementUpdate
 - afsMeasurementUpdate.cc, [1035](#)
 - afsMeasurementUpdate.h, [1037](#)
- afsMeasurementUpdate.cc, [1035](#)
- afsMeasurementUpdate.cc
 - afsMeasurementUpdate, [1035](#)
 - doKalmanUpdate, [1035](#)
 - doPriming, [1036](#)
- afsMeasurementUpdate.h, [1037](#)
- afsMeasurementUpdate.h
 - afsMeasurementUpdate, [1037](#)
- afsMotion
 - afsMain.cc, [1030](#)
 - afsMain.h, [1033](#)
- afsMotionResample
 - afsMotionResample.cc, [1038](#)
 - afsMotionResample.h, [1039](#)
- afsMotionResample.cc, [1038](#)
- afsMotionResample.cc
 - afsMotionResample, [1038](#)
- afsMotionResample.h, [1039](#)
- afsMotionResample.h
 - afsMotionResample, [1039](#)
- afsParticle
 - afsParticle.h, [1042](#)
- afsParticle.cc, [1040](#)
- afsParticle.cc
 - afsParticleCopy, [1040](#)
 - afsParticleInit, [1040](#)
- afsParticle.h, [1041](#)
- afsParticle.h
 - afsLandmarkLoc, [1042](#)
 - afsLastObservation, [1042](#)
 - afsParticle, [1042](#)
 - afsParticleCopy, [1043](#)
 - afsParticleInit, [1043](#)

- afsPose, 1042
- afsParticleCopy
 - afsParticle.cc, 1040
 - afsParticle.h, 1043
- afsParticleError
 - afsFindBestPose.cc, 1024
 - afsFindBestPose.h, 1028
- afsParticleInit
 - afsParticle.cc, 1040
 - afsParticle.h, 1043
- afsPose
 - afsParticle.h, 1042
- afsResample
 - afsMain.cc, 1030
 - afsMain.h, 1033
- afsRRP
 - afsFindBestPose.h, 1027
- afsSetLandmark
 - afsMain.cc, 1031
 - afsMain.h, 1033
- afsTriangulator
 - afsTriangulator.cc, 1044
 - afsTriangulator.h, 1045
- afsTriangulator.cc, 1044
- afsTriangulator.cc
 - afsTriangulator, 1044
 - triangulate, 1044
- afsTriangulator.h, 1045
- afsTriangulator.h
 - afsTriangulator, 1045
- afsUtility.cc, 1046
- afsUtility.cc
 - find_dtheta, 1046
 - normRand, 1046
- afsUtility.h, 1047
- afsUtility.h
 - find_dtheta, 1047
 - normRand, 1047
- afsWhatsUp
 - afsMain.cc, 1031
 - afsMain.h, 1034
- afsXY
 - afsFindBestPose.h, 1027
- Age
 - TimeET, 863
- AGM, 156
- carryOver, 156
 - decay, 156
 - dump, 156
 - genRequests, 156
 - getGM, 157
 - init, 157
 - WorldModel2, 157
- AGM_BOTTOM
 - Maps/Configuration.h, 1098
- AGM_H_SIZE
 - Maps/Configuration.h, 1098
- AGM_LEFT
 - Maps/Configuration.h, 1098
- AGM_NUMCLUSTERS
 - Maps/Configuration.h, 1098
- AGM_RIGHT
 - Maps/Configuration.h, 1098
- AGM_TOP
 - Maps/Configuration.h, 1098
- AGM_V_SIZE
 - Maps/Configuration.h, 1098
- agmMain.cc, 1048
- agmMain.cc
 - GM, 1048
- agmMain.h, 1049
- AHEAD
 - World-
 - Model2Behavior::WalkNode, 1008
- aibo3d_port
 - Config::main_config, 279
- Aibo3DControllerBehavior, 158
 - Aibo3DControllerBehavior, 160
- Aibo3DControllerBehavior
 - ~Aibo3DControllerBehavior, 160
 - Aibo3DControllerBehavior, 160
- cmdsock, 162
- DoStart, 160
- DoStop, 160
- fbuf, 162
- getClassDescription, 161
- getGUIType, 161
- getName, 161
- getPort, 161
- operator=, 162

- pos, [162](#)
- rcontrol_id, [162](#)
- registerData, [162](#)
- val, [163](#)
- aibo3dControllerBehavior
 - Aibo3DControllerBehavior.h, [1052](#)
- Aibo3DControllerBehavior.h, [1050](#)
- Aibo3DControllerBehavior.h
 - aibo3dControllerBehavior, [1052](#)
 - aibo3dcontrollercmd_callback, [1052](#)
- aibo3dcontrollercmd_callback
 - Aibo3DControllerBehavior.h, [1052](#)
- Aibo3DMonitorBehavior, [164](#)
 - Aibo3DMonitorBehavior, [165](#)
- Aibo3DMonitorBehavior
 - Aibo3DMonitorBehavior, [165](#)
 - getClassDescription, [165](#)
 - getGUIType, [165](#)
 - getName, [165](#)
 - getPort, [165](#)
- Aibo3DMonitorBehavior.h, [1053](#)
- AIBO_CAM_H_FOV
 - Maps/Configuration.h, [1098](#)
- AIBO_CAM_V_FOV
 - Maps/Configuration.h, [1099](#)
- AIBO_HEAD_LENGTH
 - Maps/Configuration.h, [1099](#)
- AIBO_IR_CAL_MULTIPLIER
 - Maps/Configuration.h, [1099](#)
- AIBO_IR_CAL_OFFSET
 - Maps/Configuration.h, [1099](#)
- AIBO_MAX_BUMP
 - Maps/Configuration.h, [1099](#)
- AIBO_MIN_CLEARANCE
 - Maps/Configuration.h, [1099](#)
- AIBO_NECK_HEIGHT
 - Maps/Configuration.h, [1099](#)
- AIBO_TILT_PIVOT_HEIGHT
 - Maps/Configuration.h, [1099](#)
- AIBO_TURN_PIVOT_DIST
 - Maps/Configuration.h, [1099](#)
- aiboIsErect
 - WorldModel2.cc, [1428](#)
 - WorldModel2.h, [1431](#)
- aiboIsLevelHeaded
 - WorldModel2.cc, [1428](#)
 - WorldModel2.h, [1431](#)
- aiboStaresDeadAhead
 - WorldModel2.cc, [1428](#)
 - WorldModel2.h, [1431](#)
- aiEGID
 - EventBase, [361](#)
- air
 - WalkMC::LegWalkState, [960](#)
- airpath
 - WalkMC::LegWalkState, [960](#)
- allevents
 - EventRouter::EventMapper, [399](#)
- AllLEDMask
 - ERS210Info, [42](#)
 - ERS220Info, [70](#)
 - ERS2xxInfo, [99](#)
- ALM, [167](#)
 - dumpDM, [167](#)
 - dumpHM, [167](#)
 - genRequests, [167](#)
 - getDM, [168](#)
 - getHM, [168](#)
 - init, [168](#)
 - move, [168](#)
 - nukeAndPaveCurrentMap, [168](#)
 - registerDepth, [169](#)
 - registerGround, [169](#)
 - stampHM, [169](#)
 - WorldModel2, [170](#)
- ALM_DM_BOTTOM
 - Maps/Configuration.h, [1099](#)
- ALM_DM_H_SIZE
 - Maps/Configuration.h, [1100](#)
- ALM_DM_LEFT
 - Maps/Configuration.h, [1100](#)
- ALM_DM_NUMCLUSTERS
 - Maps/Configuration.h, [1100](#)
- ALM_DM_RIGHT
 - Maps/Configuration.h, [1100](#)
- ALM_DM_TAX
 - Maps/Configuration.h, [1100](#)
- ALM_DM_TOP
 - Maps/Configuration.h, [1100](#)

- ALM_DM_V_SIZE
 - Maps/Configuration.h, [1100](#)
- ALM_GM_TAX
 - Maps/Configuration.h, [1100](#)
- ALM_GPA_CONFIDENCE
 - Maps/Configuration.h, [1100](#)
- ALM_HM_NUMCLUSTERS
 - Maps/Configuration.h, [1100](#)
- ALM_HM_RADIUS
 - Maps/Configuration.h, [1101](#)
- ALM_HM_SIZE
 - Maps/Configuration.h, [1101](#)
- ALM_HM_TAX
 - Maps/Configuration.h, [1101](#)
- ALM_IR_SPLAT_STDDEV
 - Maps/Configuration.h, [1101](#)
- almMain.cc, [1054](#)
- almMain.cc
 - DM, [1055](#)
 - DMs, [1055](#)
 - HM, [1055](#)
 - HMs, [1055](#)
 - IROORDIST, [1055](#)
 - SQRT_2_PI, [1055](#)
- almMain.h, [1056](#)
- almStructures.h, [1057](#)
- almStructures.h
 - colortype, [1058](#)
 - dm_cell, [1058](#)
 - hm_cell, [1059](#)
- almUtility.cc, [1060](#)
- almUtility.cc
 - angles2dm_index, [1060](#)
 - dm_index2angles, [1060](#)
 - gm_index2xy, [1060](#)
 - head_range2xyz, [1061](#)
 - hm_index2xy, [1061](#)
 - neck_range2xyz, [1061](#)
 - xy2gm_index, [1061](#)
 - xy2hm_index, [1061](#)
 - xyz2neck_range, [1061](#)
- almUtility.h, [1063](#)
- almUtility.h
 - angles2dm_index, [1063](#)
 - dm_index2angles, [1063](#)
 - gm_index2xy, [1063](#)
 - head_range2xyz, [1064](#)
 - hm_index2xy, [1064](#)
 - neck_range2xyz, [1064](#)
 - xy2gm_index, [1064](#)
 - xy2hm_index, [1064](#)
 - xyz2neck_range, [1064](#)
- alreadyGotBoth
 - Controller, [319](#)
- altitude
 - MotionRequest, [612](#)
- alwaysGenerateStatus
 - WorldState, [1016](#)
- AM_KMEANS_ITERATIONS
 - Maps/Configuration.h, [1101](#)
- amp
 - LedEngine::LEDInfo, [489](#)
- angle
 - BodyPosition, [253](#)
 - GVector::vector2d, [892](#)
- angle_delta
 - WalkMC, [954](#)
- angle_wrap
 - Util.h, [1372](#)
- angles2dm_index
 - almUtility.cc, [1060](#)
 - almUtility.h, [1063](#)
- ASSERT
 - debuget.h, [1111](#)
- ASSERTFATAL
 - debuget.h, [1111](#)
- ASSERTRET
 - debuget.h, [1111](#)
- ASSERTRETVAL
 - debuget.h, [1111](#)
- atan2a
 - Util.h, [1372](#)
- atan2b
 - Util.h, [1372](#)
- audioEGID
 - EventBase, [361](#)
- AutoGetupBehavior, [171](#)
 - AutoGetupBehavior, [172](#)
- AutoGetupBehavior
 - ~AutoGetupBehavior, [172](#)
 - AutoGetupBehavior, [172](#)
 - back, [174](#)

- DoStart, [173](#)
- DoStop, [173](#)
- gamma, [174](#)
- getClassDescription, [173](#)
- getName, [173](#)
- processEvent, [173](#)
- sensitivity, [174](#)
- side, [174](#)
- waiting, [174](#)
- AutoGetupBehavior.h, [1066](#)
- AutoLock
 - SharedQueue, [756](#)
 - SoundManager.cc, [1323](#)
- autoprun
 - MotionCommand, [579](#)
- avg
 - SmoothCompareTrans, [768](#)
- avg_angle
 - Util.h, [1372](#)
- avgdiff
 - PostureEngine, [682](#)
- azalt
 - MotionRequest, [612](#)
- azimuth
 - MotionRequest, [613](#)
- b
 - BanditMachine::DecideNode, [182](#)
 - BanditMachine::WaitNode, [188](#)
 - HermiteSplineSegment, [461](#)
 - NonUniformHermiteSplineSegment, [653](#)
- BAccelOffset
 - ERS210Info, [42](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [99](#)
- Back
 - SoundTestBehavior, [821](#)
- back
 - AutoGetupBehavior, [174](#)
 - ListMemBuf, [500](#)
- BackButOffset
 - ERS210Info, [39](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [96](#)
- BackButSID
 - ButtonSourceID, [30](#)
- BackLEDMask
 - ERS210Info, [42](#)
 - ERS220Info, [70](#)
 - ERS2xxInfo, [99](#)
- BackLeft1LEDMask
 - ERS220Info, [70](#)
 - ERS2xxInfo, [100](#)
- BackLeft1LEDOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- BackLeft2LEDMask
 - ERS220Info, [71](#)
 - ERS2xxInfo, [100](#)
- BackLeft2LEDOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- BackLeft3LEDMask
 - ERS220Info, [71](#)
 - ERS2xxInfo, [100](#)
- BackLeft3LEDOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- BackRight1LEDMask
 - ERS220Info, [71](#)
 - ERS2xxInfo, [100](#)
- BackRight1LEDOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- BackRight2LEDMask
 - ERS220Info, [71](#)
 - ERS2xxInfo, [100](#)
- BackRight2LEDOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- BackRight3LEDMask
 - ERS220Info, [71](#)
 - ERS2xxInfo, [101](#)
- BackRight3LEDOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- bandit
 - BanditMachine, [178](#)
- BanditMachine, [175](#)
- BanditMachine, [176](#), [177](#)

- BanditMachine
 - ~BanditMachine, 177
 - bandit, 178
 - BanditMachine, 176, 177
 - DoStart, 177
 - DoStop, 177
 - getClassDescription, 177
 - liedown, 178
 - operator=, 178
 - setup, 178
 - stare, 178
 - start, 179
- BanditMachine.h, 1068
- BanditMachine::DecideNode, 180
- BanditMachine::DecideNode
 - b, 182
 - DecideNode, 181
 - DoStart, 181
 - l, 182
 - operator=, 182
 - r, 182
- BanditMachine::PressNode, 183
- BanditMachine::PressNode
 - ~PressNode, 184
 - DoStart, 184
 - DoStop, 184
 - index, 185
 - press_id, 185
 - PressNode, 184
- BanditMachine::WaitNode, 186
- BanditMachine::WaitNode
 - ~WaitNode, 187
 - b, 188
 - DoStart, 187
 - DoStop, 187
 - leds_id, 188
 - processEvent, 188
 - reward, 188
 - WaitNode, 187
- baseaddr
 - MotionManager::Command-Entry, 601
- basic_iNetStream, 190
 - basic_iNetStream, 191, 192
- basic_iNetStream
 - basic_iNetStream, 191, 192
 - char_type, 191
 - close, 192
 - getEcho, 192
 - int_type, 191
 - is_open, 192
 - nb, 194
 - off_type, 191
 - open, 193
 - pos_type, 191
 - rdbuf, 193
 - setEcho, 193
 - traits_type, 191
- basic_ioNetStream, 195
 - basic_ioNetStream, 196, 197
- basic_ioNetStream
 - basic_ioNetStream, 196, 197
 - char_type, 196
 - close, 197
 - getEcho, 197
 - int_type, 196
 - is_open, 198
 - nb, 199
 - off_type, 196
 - open, 198
 - pos_type, 196
 - rdbuf, 198
 - setEcho, 198
 - traits_type, 196
- basic_netbuf, 200
 - ~basic_netbuf, 203
 - basic_netbuf, 202, 203
 - buf_in, 206
 - buf_out, 206
 - char_type, 201
 - close, 203
 - def_buf_in_size, 207
 - def_buf_out_size, 207
 - getEcho, 203
 - in_sync, 203
 - Init, 204
 - int_type, 201
 - is_echoing, 207
 - is_open, 204
 - off_type, 201
 - open, 204, 205
 - out_flush, 205

- overflow, [205](#)
- pos_type, [202](#)
- printBuffer, [205](#)
- setEcho, [205](#)
- showmanyc, [206](#)
- sock, [207](#)
- sync, [206](#)
- traits_type, [202](#)
- uflow, [206](#)
- underflow, [206](#)
- using_buf_in, [207](#)
- using_buf_out, [207](#)
- basic_oNetStream, [208](#)
 - basic_oNetStream, [209](#), [210](#)
- basic_oNetStream
 - basic_oNetStream, [209](#), [210](#)
 - char_type, [209](#)
 - close, [210](#)
 - getEcho, [210](#)
 - int_type, [209](#)
 - is_open, [211](#)
 - nb, [212](#)
 - off_type, [209](#)
 - open, [211](#)
 - pos_type, [209](#)
 - rdbuf, [211](#)
 - setEcho, [211](#)
 - traits_type, [209](#)
- BatteryCheckControl, [213](#)
 - BatteryCheckControl, [214](#)
- BatteryCheckControl
 - ~BatteryCheckControl, [214](#)
 - activate, [214](#)
 - BatteryCheckControl, [214](#)
 - deactivate, [215](#)
 - doSelect, [215](#)
 - pause, [215](#)
 - processEvent, [215](#)
 - refresh, [215](#)
 - report, [216](#)
- BatteryCheckControl.h, [1070](#)
- BatteryConnectSID
 - PowerSourceID, [124](#)
- BatteryEmptySID
 - PowerSourceID, [124](#)
- BatteryFullSID
 - PowerSourceID, [124](#)
- BatteryInitSID
 - PowerSourceID, [124](#)
- BatteryMonitorBehavior, [217](#)
 - BatteryMonitorBehavior, [219](#)
- BatteryMonitorBehavior
 - ~BatteryMonitorBehavior, [219](#)
 - BatteryMonitorBehavior, [219](#)
 - calcFlipDelay, [220](#)
 - DoStart, [220](#)
 - DoStop, [220](#)
 - getClassDescription, [220](#)
 - getName, [220](#)
 - high_power_p, [222](#)
 - led_id, [222](#)
 - max_t, [222](#)
 - no_power_p, [223](#)
 - operator=, [221](#)
 - pose, [223](#)
 - pose_id, [223](#)
 - processEvent, [221](#)
 - setFlipper, [221](#)
 - shouldWarn, [221](#)
 - startWarning, [222](#)
 - stopWarning, [222](#)
- BatteryMonitorBehavior.h, [1072](#)
- BatteryOverCurrentSID
 - PowerSourceID, [124](#)
- batteryStatus
 - WorldState, [1016](#)
- begin
 - ListMemBuf, [500](#)
 - MotionManager, [588](#)
- BehaviorActivatorControl, [224](#)
 - BehaviorActivatorControl, [226](#)
 - start, [225](#)
 - stop, [225](#)
 - toggle, [225](#)
- BehaviorActivatorControl
 - ~BehaviorActivatorControl, [226](#)
 - activate, [227](#)
 - BehaviorActivatorControl, [226](#)
 - init, [227](#)
 - mode, [227](#)
 - Mode_t, [225](#)
 - operator=, [227](#)

- target, 227
- BehaviorActivatorControl.h, 1074
- BehaviorBase, 228
 - BehaviorBase, 230
- BehaviorBase
 - ~BehaviorBase, 230
 - BehaviorBase, 230
 - DoStart, 230
 - DoStop, 230
 - getClassDescription, 231
 - getDescription, 231
 - getName, 231
 - isActive, 232
 - processEvent, 232
 - started, 232
- BehaviorBase.h, 1076
- BehaviorGroup
 - BehaviorSwitchControl-Base::BehaviorGroup, 250
- behaviors
 - Config, 273
- BehaviorSwitchActivatorControl, 234
 - BehaviorSwitchActivatorControl, 235, 236
 - start, 235
 - stop, 235
 - toggle, 235
- BehaviorSwitchActivatorControl
 - ~BehaviorSwitchActivatorControl, 235
 - activate, 236
 - BehaviorSwitchActivatorControl, 235, 236
 - behswitch, 237
 - getDescription, 236
 - getName, 236
 - mode, 237
 - Mode_t, 235
 - operator=, 237
- BehaviorSwitchActivatorControl.h, 1078
- BehaviorSwitchControl, 238
 - BehaviorSwitchControl, 240, 241
- BehaviorSwitchControl
 - ~BehaviorSwitchControl, 240
 - BehaviorSwitchControl, 240, 241
 - getDescription, 241
 - getName, 241
 - isRunning, 241
 - isValid, 241
 - mybeh, 243
 - operator=, 242
 - start, 242
 - startmine, 242
 - stop, 242
 - stopother, 243
 - toggle, 243
- BehaviorSwitchControl.h, 1079
- BehaviorSwitchControlBase, 245
 - BehaviorSwitchControlBase, 246
- BehaviorSwitchControlBase
 - ~BehaviorSwitchControlBase, 246
 - activate, 247
 - BehaviorSwitchControlBase, 246
 - behgrp, 248
 - operator=, 247
 - retained, 248
 - start, 247
 - stop, 247
 - toggle, 247
- BehaviorSwitchControlBase::BehaviorGroup, 249
- BehaviorSwitchControl-Base::BehaviorGroup
 - ~BehaviorGroup, 250
 - BehaviorGroup, 250
 - curBehavior, 251
 - operator=, 250
- behgrp
 - BehaviorSwitchControlBase, 248
- behswitch
 - BehaviorSwitchActivatorControl, 237
- BindCont
 - MMCombo, 561
 - Wireless, 977
- BinJointOffset
 - ERS210Info, 43
 - ERS220Info, 71
 - ERS2xxInfo, 101

- bits_u
 - Vision.h, [1386](#)
- bits_v
 - Vision.h, [1386](#)
- bits_y
 - Vision.h, [1386](#)
- BL_in_use
 - MutexLock::door_t, [650](#)
- BL_ready
 - MutexLock::door_t, [650](#)
- blockingSend
 - Wireless, [977](#)
- BMNDebugModeSID
 - PowerSourceID, [125](#)
- body_angle
 - Vision, [913](#)
 - WalkMC, [954](#)
 - WalkMC::WalkParam, [963](#)
- body_height
 - Vision, [913](#)
 - WalkMC::WalkParam, [963](#)
- body_loc
 - WalkMC, [954](#)
- body_to_neck
 - Kinematics.h, [1209](#)
- BodyPosition, [252](#)
 - BodyPosition, [252](#)
- BodyPosition
 - angle, [253](#)
 - BodyPosition, [252](#)
 - loc, [253](#)
- BodyRelative
 - HeadPointerMC, [450](#)
- BotLLEDMask
 - ERS210Info, [43](#)
 - ERS220Info, [72](#)
 - ERS2xxInfo, [101](#)
- BotLLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- BotRLEDMask
 - ERS210Info, [43](#)
 - ERS220Info, [72](#)
 - ERS2xxInfo, [101](#)
- BotRLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- bound
 - Util.h, [1372](#)
- BOUND_MOTION
 - WalkMC.cc, [1406](#)
- buckets
 - Profiler, [707](#)
- buf
 - SharedQueue, [758](#)
- buf_end
 - SharedQueue, [757](#)
- buf_in
 - basic_netbuf, [206](#)
- buf_out
 - basic_netbuf, [206](#)
- buffertime
 - EventRouter, [392](#)
- button_times
 - WorldState, [1016](#)
- buttonEGID
 - EventBase, [361](#)
- ButtonOffset_t
 - ERS210Info, [39](#)
 - ERS220Info, [66](#)
 - ERS2xxInfo, [96](#)
- buttons
 - WorldState, [1016](#)
- ButtonSourceID, [29](#)
 - BackButSID, [30](#)
 - ChinButSID, [30](#)
 - HeadBkButSID, [30](#)
 - HeadFrButSID, [30](#)
 - LBkPawSID, [30](#)
 - LFrPawSID, [30](#)
 - RBkPawSID, [30](#)
 - RFrPawSID, [30](#)
 - TailCenterButSID, [30](#)
 - TailLeftButSID, [30](#)
 - TailRightButSID, [30](#)
- ButtonSourceID
 - ButtonSourceID_t, [29](#)
- ButtonSourceID_t
 - ButtonSourceID, [29](#)
- byteswap

- LoadSave, 520
- c
 - HermiteSplineSegment, 461
 - NonUniformHermiteSplineSegment, 653
- calcCycle
 - LedEngine, 481
- calcDers
 - WorldState, 1014
- calcEdgeMask
 - Vision, 907
- calcFlash
 - LedEngine, 481
- calcFlipDelay
 - BatteryMonitorBehavior, 220
- calcInvert
 - LedEngine, 482
- calcLEDValue
 - MMCombo, 561
- calcOutput
 - MotionSequence, 621
- calcPulse
 - Controller, 313
- calcsz
 - SharedObject, 749
- calcTotalArea
 - Vision, 913
- calcValue
 - LedEngine, 482
- callback
 - EStopControllerBehavior, 354
- calls
 - Profiler::SectionInfo, 711
- camera_click
 - CameraBehavior, 257
- camera_dir
 - Vision, 913
- camera_loc
 - Vision, 914
- camera_right
 - Vision, 914
- camera_up
 - Vision, 914
- CameraBehavior, 254
 - CameraBehavior, 255
- CameraBehavior
 - ~CameraBehavior, 256
 - camera_click, 257
 - CameraBehavior, 255
 - DoStart, 256
 - DoStop, 256
 - getClassDescription, 256
 - getName, 257
 - headpointer_id, 257
 - led_id, 258
 - processEvent, 257
 - RAD, 257
 - sensor_update, 258
 - tailwag_id, 258
- CameraBehavior.h, 1081
- cancel
 - Controller, 319
- cancel_snd
 - Config::controller_config, 277
- carryOver
 - AGM, 156
- ccycle
 - LedEngine, 482
- cen_x
 - Marker, 546
- cen_y
 - Marker, 546
- cflash
 - LedEngine, 482
- Chain
 - SoundManager, 789
- ChainBuffer
 - SoundManager, 789
- ChainFile
 - SoundManager, 790
- chanlist
 - SoundManager, 797
- chanlist_t
 - SoundManager, 787
- char_traits, 259
 - char_type, 259
 - copy, 260
 - eof, 260
 - int_type, 259
 - move, 260
 - off_type, 259

- pos_type, [259](#)
- to_int_type, [260](#)
- char_type
 - basic_iNetStream, [191](#)
 - basic_ioNetStream, [196](#)
 - basic_netbuf, [201](#)
 - basic_oNetStream, [209](#)
 - char_traits, [259](#)
- ChargerStatusSID
 - PowerSourceID, [124](#)
- ChargingSID
 - PowerSourceID, [124](#)
- charhexout
 - debuget.h, [1112](#)
- ChaseBallBehavior, [261](#)
 - ChaseBallBehavior, [262](#)
- ChaseBallBehavior
 - ~ChaseBallBehavior, [262](#)
 - ChaseBallBehavior, [262](#)
 - DoStart, [262](#)
 - DoStop, [262](#)
 - getClassDescription, [263](#)
 - getName, [263](#)
 - headpointer_id, [264](#)
 - processEvent, [263](#)
 - walker_id, [264](#)
- ChaseBallBehavior.cc, [1083](#)
- ChaseBallBehavior.cc
 - DtoR, [1084](#)
- ChaseBallBehavior.h, [1085](#)
- checkCreator
 - LoadSave, [520](#), [521](#)
- checkin
 - MMAccessor, [552](#), [553](#)
- checkinMotion
 - MotionManager, [588](#)
- checkLogFile
 - EventLogger, [375](#)
- checkout
 - MMAccessor, [553](#)
- checkoutLevel
 - MotionManager, [588](#)
- checkoutMotion
 - MotionManager, [589](#)
- checksum
 - WalkMC.cc, [1406](#)
- childTime
 - Profiler::SectionInfo, [711](#)
- ChinButOffset
 - ERS210Info, [39](#)
 - ERS220Info, [66](#)
 - ERS2xxInfo, [96](#)
- ChinButSID
 - ButtonSourceID, [30](#)
- ChkAdvance
 - MotionSequence, [621](#)
 - PostureEngine, [683](#)
- chkCmdStack
 - Controller, [313](#)
- chkEvent
 - WorldState, [1014](#)
- chkPowerEvent
 - WorldState, [1014](#)
- chkTimers
 - EventRouter, [386](#)
- clean
 - EventRouter::EventManager, [397](#)
- clear
 - DynamicMotionSequence, [340](#)
 - EventRouter::EventManager, [397](#)
 - LedEngine, [482](#)
 - ListMemBuf, [500](#)
 - MotionSequence, [621](#)
 - MotionSequenceMC, [638](#)
 - PostureEngine, [683](#)
 - PostureMC, [693](#)
 - SharedQueue, [757](#)
- clearbits
 - Util.h, [1372](#)
- clearMenu
 - ControlBase, [298](#)
- clearSlots
 - ControlBase, [298](#)
- clock
 - FollowHeadBehavior, [428](#)
- close
 - basic_iNetStream, [192](#)
 - basic_ioNetStream, [197](#)
 - basic_netbuf, [203](#)
 - basic_oNetStream, [210](#)
 - Vision, [907](#)
 - WalkToTargetMachine, [968](#)

- Wireless, 978
- CloseCont
 - MMCombo, 561
 - Wireless, 978
- closeGUI
 - Controller, 313
- cluster
 - _dm_cell, 150
 - _hm_cell, 154
- cmap
 - Vision, 914
- cmap_t
 - Vision.h, 1386
- cmd
 - MotionSequence::Move, 634
- CMD_fwd
 - WalkControllerBehavior, 940
- CMD_opt0
 - WalkControllerBehavior, 940
- CMD_opt1
 - WalkControllerBehavior, 940
- CMD_opt2
 - WalkControllerBehavior, 940
- CMD_opt3
 - WalkControllerBehavior, 941
- CMD_opt4
 - WalkControllerBehavior, 941
- CMD_opt5
 - WalkControllerBehavior, 941
- CMD_opt6
 - WalkControllerBehavior, 941
- CMD_opt7
 - WalkControllerBehavior, 941
- CMD_opt8
 - WalkControllerBehavior, 941
- CMD_opt9
 - WalkControllerBehavior, 942
- CMD_pan
 - HeadPointControllerBehavior, 445
- CMD_roll
 - HeadPointControllerBehavior, 445
- CMD_roto
 - WalkControllerBehavior, 942
- CMD_side
 - WalkControllerBehavior, 942
- CMD_tilt
 - HeadPointControllerBehavior, 445
- cmdlist
 - MotionManager, 596
- cmds
 - LedMC, 494
 - MotionManager, 596
 - PostureEngine, 688
 - RemoteControllerMC, 730
 - WalkMC, 955
- cmdsock
 - Aibo3DControllerBehavior, 162
 - EStopControllerBehavior, 356
 - HeadPointControllerBehavior, 445
 - WalkControllerBehavior, 942
- cmdstack
 - Controller, 319
- cmdstatelist_t
 - MotionManager, 586
- cmdstates
 - MotionManager, 597
- cmdSums
 - MotionManager, 597
- color
 - _dm_cell, 150
 - _hm_cell, 154
 - Vision, 914
- COLOR_BACKGROUND
 - Visiondefines.h, 1388
- COLOR_BGREEN
 - Visiondefines.h, 1388
- COLOR_BLACK
 - Visiondefines.h, 1388
- COLOR_BLUE
 - Visiondefines.h, 1389
- color_class_state
 - Vision.h, 1386
- COLOR_GRAY
 - Visiondefines.h, 1389
- COLOR_GREEN
 - Visiondefines.h, 1389
- COLOR_ORANGE
 - Visiondefines.h, 1389

- COLOR_PINK
 - Visiondefines.h, [1389](#)
- COLOR_PURPLE
 - Visiondefines.h, [1389](#)
- COLOR_RED
 - Visiondefines.h, [1389](#)
- COLOR_SKIN
 - Visiondefines.h, [1389](#)
- COLOR_YELLOW
 - Visiondefines.h, [1389](#)
- colors
 - Config::vision_config, [287](#)
- colortype
 - almStructures.h, [1058](#)
- CommandEntry
 - MotionManager::Command-Entry, [601](#)
- CompareTrans, [265](#)
 - CompareTrans, [267](#)
 - EQ, [267](#)
 - GT, [267](#)
 - GTE, [267](#)
 - LT, [267](#)
 - LTE, [267](#)
 - NE, [267](#)
- CompareTrans
 - CompareTrans, [267](#)
 - disable, [267](#)
 - enable, [268](#)
 - mon, [268](#)
 - operator=, [268](#)
 - poller, [268](#)
 - processEvent, [268](#)
 - Test.t, [267](#)
 - tst, [269](#)
 - val, [269](#)
- CompareTrans.h, [1087](#)
- compress
 - MotionSequence, [622](#)
- confidence
 - _dm_cell, [150](#)
 - _hm_cell, [154](#)
 - VisionEventSpec, [924](#)
 - VisionInterface::VObject, [933](#)
- Config, [270](#)
 - ~Config, [272](#)
 - behaviors, [273](#)
 - Config, [272](#)
 - controller, [273](#)
 - extractBool, [272](#)
 - main, [273](#)
 - motion, [273](#)
 - readConfig, [272](#)
 - sec_behaviors, [271](#)
 - sec_controller, [271](#)
 - sec_invalid, [271](#)
 - sec_main, [271](#)
 - sec_motion, [271](#)
 - sec_sound, [271](#)
 - sec_vision, [271](#)
 - sec_wireless, [271](#)
 - sec_worldmodel2, [271](#)
 - section.t, [271](#)
 - sound, [273](#)
 - vision, [273](#)
 - wireless, [273](#)
 - worldmodel2, [273](#)
- config
 - Config.cc, [1090](#)
 - Config.h, [1093](#)
- Config.cc, [1089](#)
 - config, [1090](#)
- Config.h, [1091](#)
 - config, [1093](#)
- Config::behaviors_config, [275](#)
- Config::controller_config, [276](#)
 - cancel_snd, [277](#)
 - controller_config, [276](#)
 - gui_port, [277](#)
 - next_snd, [277](#)
 - prev_snd, [277](#)
 - read_snd, [277](#)
 - select_snd, [277](#)
- Config::main_config, [278](#)
 - aiibo3d_port, [279](#)
 - console_port, [279](#)
 - debug_level, [279](#)
 - error_level, [279](#)
 - estopControl_port, [280](#)
 - headControl_port, [280](#)
 - main_config, [279](#)
 - stderr_port, [280](#)

- use_VT100, [280](#)
 - verbose_level, [280](#)
 - walkControl_port, [280](#)
 - wsjoints_port, [280](#)
 - wspids_port, [281](#)
- Config::motion_config, [282](#)
 - estop_off_snd, [283](#)
 - estop_on_snd, [283](#)
 - makePath, [283](#)
 - motion_config, [282](#)
 - root, [283](#)
- Config::sound_config, [284](#)
 - makePath, [285](#)
 - preload, [285](#)
 - root, [285](#)
 - sample_bits, [285](#)
 - sample_rate, [285](#)
 - sound_config, [284](#)
- Config::vision_config, [286](#)
 - colors, [287](#)
 - gain, [287](#)
 - obj_port, [287](#)
 - raw_port, [287](#)
 - resolution, [287](#)
 - rle_port, [288](#)
 - shutter_speed, [288](#)
 - thresh, [288](#)
 - vision_config, [287](#)
 - white_balance, [288](#)
- Config::wireless_config, [289](#)
 - id, [289](#)
 - wireless_config, [289](#)
- Config::worldmodel2_config, [290](#)
 - dm_port, [291](#)
 - fs_port, [291](#)
 - gm_port, [291](#)
 - hm_port, [291](#)
 - worldmodel2_config, [290](#)
- Configuration.h, [1094](#), [1097](#)
- connect
 - Wireless, [978](#), [979](#)
- ConnectCont
 - MMCombo, [562](#)
 - Wireless, [979](#)
- CONNECTION_CLOSED
 - SocketNS, [128](#)
- CONNECTION_CLOSING
 - SocketNS, [128](#)
- CONNECTION_CONNECTED
 - SocketNS, [128](#)
- CONNECTION_CONNECTING
 - SocketNS, [128](#)
- CONNECTION_ERROR
 - SocketNS, [128](#)
- CONNECTION_LISTENING
 - SocketNS, [128](#)
- ConnectionState
 - SocketNS, [128](#)
- console_callback
 - Controller, [314](#)
- console_port
 - Config::main_config, [279](#)
- construct
 - Factory, [415](#)
 - Factory1Arg, [417](#)
- ControlBase, [292](#)
 - ControlBase, [297](#)
- ControlBase
 - ~ControlBase, [297](#)
 - activate, [298](#)
 - clearMenu, [298](#)
 - clearSlots, [298](#)
 - ControlBase, [297](#)
 - deactivate, [298](#)
 - description, [304](#)
 - display_id, [304](#)
 - doCancel, [299](#)
 - doNextItem, [299](#)
 - doPrevItem, [299](#)
 - doReadStdIn, [299](#)
 - doRewrite, [304](#)
 - doSelect, [300](#)
 - getDescription, [300](#)
 - getDisplay, [300](#)
 - getHighlights, [301](#)
 - getName, [301](#)
 - getSlotName, [301](#)
 - getSlots, [301](#)
 - gui_comm, [305](#)
 - highlightFirst, [301](#)
 - hilights, [305](#)
 - hilightsAvg, [301](#)

- name, 305
- operator=, 302
- options, 305
- pause, 302
- pushSlot, 302
- refresh, 302
- setDescription, 302
- setDisplay, 303
- setHighlights, 303
- setName, 303
- setSlot, 303
- slotsSize, 303
- takeInput, 304
- validInput, 304
- ControlBase.cc, 1102
- ControlBase.h, 1103
- Controller, 306
 - ~Controller, 313
 - activate, 313
 - alreadyGotBoth, 319
 - calcPulse, 313
 - cancel, 319
 - chkCmdStack, 313
 - closeGUI, 313
 - cmdstack, 319
 - console_callback, 314
 - Controller, 312, 313
 - cur_time, 319
 - deactivate, 314
 - display, 319
 - DoStart, 314
 - DoStop, 314
 - estop_id, 319
 - getClassDescription, 315
 - getName, 315
 - gui_comm, 319
 - gui_comm_callback, 315
 - init, 315
 - last_time, 320
 - loadGUI, 315, 316
 - makeLower, 316
 - nextEv_dur, 320
 - nextEv_val, 320
 - nextItem, 320
 - nextItemFast, 320
 - operator=, 316
 - pop, 316
 - prevEv_dur, 320
 - prevEv_val, 320
 - prevItem, 321
 - prevItemFast, 321
 - processEvent, 316
 - push, 316
 - refresh, 317
 - removePrefix, 317
 - reset, 317
 - root, 321
 - selectItem, 321
 - setEStopID, 317
 - setNext, 317
 - setRoot, 318
 - takeLine, 318
 - theOneController, 321
 - top, 318
 - trapEvent, 318
- controller
 - Config, 273
- Controller.cc, 1105
- Controller.h, 1107
- controller_config
 - Config::controller_config, 276
- convert
 - HeadPointerMC, 451
- convFromBodyRelative
 - HeadPointerMC, 451
- convToBodyRelative
 - HeadPointerMC, 451
- CoordFrame_t
 - HeadPointerMC, 450
- copies
 - ValueEditControl, 884
- copy
 - char_traits, 260
- CopyTo
 - SoundManager, 790
 - WAV, 971
- count
 - VisionEventSpec, 924
- countb
 - ListMemBuf, 500
- countf
 - ListMemBuf, 500

- CPCJointLFElevator
 - ERS210Info, [43](#)
 - ERS220Info, [72](#)
- CPCJointLFKnee
 - ERS210Info, [43](#)
 - ERS220Info, [72](#)
- CPCJointLFRotator
 - ERS210Info, [43](#)
 - ERS220Info, [72](#)
- CPCJointLHElevator
 - ERS210Info, [43](#)
 - ERS220Info, [72](#)
- CPCJointLHKnee
 - ERS210Info, [44](#)
 - ERS220Info, [72](#)
- CPCJointLHRotator
 - ERS210Info, [44](#)
 - ERS220Info, [72](#)
- CPCJointMouth
 - ERS210Info, [44](#)
- CPCJointNeckPan
 - ERS210Info, [44](#)
 - ERS220Info, [72](#)
- CPCJointNeckRoll
 - ERS210Info, [44](#)
 - ERS220Info, [73](#)
- CPCJointNeckTilt
 - ERS210Info, [44](#)
 - ERS220Info, [73](#)
- CPCJointRFElevator
 - ERS210Info, [44](#)
 - ERS220Info, [73](#)
- CPCJointRFKnee
 - ERS210Info, [45](#)
 - ERS220Info, [73](#)
- CPCJointRFRotator
 - ERS210Info, [45](#)
 - ERS220Info, [73](#)
- CPCJointRHElevator
 - ERS210Info, [45](#)
 - ERS220Info, [73](#)
- CPCJointRHKnee
 - ERS210Info, [45](#)
 - ERS220Info, [73](#)
- CPCJointRHRotator
 - ERS210Info, [45](#)
- ERS220Info, [73](#)
- CPCJointTailPan
 - ERS210Info, [45](#)
- CPCJointTailTilt
 - ERS210Info, [45](#)
- CPCSensorAccelFB
 - ERS210Info, [46](#)
 - ERS220Info, [73](#)
- CPCSensorAccelLR
 - ERS210Info, [46](#)
 - ERS220Info, [73](#)
- CPCSensorAccelUD
 - ERS210Info, [46](#)
 - ERS220Info, [74](#)
- CPCSensorBackSwitch
 - ERS210Info, [46](#)
 - ERS220Info, [74](#)
- CPCSensorChinSwitch
 - ERS210Info, [46](#)
 - ERS220Info, [74](#)
- CPCSensorHeadBackPressure
 - ERS210Info, [46](#)
 - ERS220Info, [74](#)
- CPCSensorHeadFrontPressure
 - ERS210Info, [46](#)
 - ERS220Info, [74](#)
- CPCSensorLFPaw
 - ERS210Info, [47](#)
 - ERS220Info, [74](#)
- CPCSensorLHPaw
 - ERS210Info, [47](#)
 - ERS220Info, [74](#)
- CPCSensorPSD
 - ERS210Info, [47](#)
 - ERS220Info, [74](#)
- CPCSensorRFPaw
 - ERS210Info, [47](#)
 - ERS220Info, [74](#)
- CPCSensorRHPaw
 - ERS210Info, [47](#)
 - ERS220Info, [74](#)
- CPCSensorTailCenterSwitch
 - ERS220Info, [75](#)
- CPCSensorTailLeftSwitch
 - ERS220Info, [75](#)
- CPCSensorTailRightSwitch

- ERS220Info, [75](#)
- CPCSensorThermoSensor
 - ERS210Info, [47](#)
 - ERS220Info, [75](#)
- create
 - HermiteSplineSegment, [461](#)
 - NonUniformHermiteSplineSegment, [653](#)
- createAverage
 - PostureEngine, [683](#)
- createCombine
 - PostureEngine, [684](#)
- createEvent
 - Vision, [907](#)
- createOverlay
 - PostureEngine, [684](#)
- createUnderlay
 - PostureEngine, [684](#)
- creatorSize
 - LoadSave, [521](#)
- cross
 - GVector, [117](#)
 - GVector::vector2d, [892](#)
 - GVector::vector3d, [898](#)
- cset
 - LedEngine, [483](#)
- cumulative
 - SoundManager::PlayState, [801](#)
- cur
 - ValueEditControl, [884](#)
- cur_cmd
 - MotionManager, [597](#)
- cur_time
 - Controller, [319](#)
- cur_tmap
 - Vision, [914](#)
- curBehavior
 - BehaviorSwitchControlBase::BehaviorGroup, [251](#)
- curplay
 - SoundTestBehavior, [821](#)
- curs
 - MotionSequence, [629](#)
- curSection
 - Profiler, [707](#)
- cursize
 - ListMemBuf, [507](#)
- curstamps
 - MotionSequence, [629](#)
- curtime
 - WorldState, [1016](#)
- cx
 - VisionEventSpec, [924](#)
- cy
 - VisionEventSpec, [924](#)
- cycle
 - LedEngine, [483](#)
- d
 - HermiteSplineSegment, [461](#)
 - NonUniformHermiteSplineSegment, [653](#)
 - TimeOutTrans, [869](#)
- da
 - _FastSLAM_update, [153](#)
 - motionReshapeKludge.h, [1254](#)
 - WalkControllerBehavior, [942](#)
 - WorldModel2, [993](#)
- DA_LEFT_MOT_THRESH
 - motionReshapeKludge.h, [1252](#)
- DA_PAN
 - Poses.h, [1273](#)
- DA_RIGHT_MOT_THRESH
 - motionReshapeKludge.h, [1252](#)
- DA_ROLL
 - Poses.h, [1273](#)
- DA_TILT
 - Poses.h, [1273](#)
- DA_TOLERANCE
 - Poses.h, [1273](#)
- DAccelOffset
 - ERS210Info, [42](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [99](#)
- data
 - ListMemBuf::entry_t, [510](#)
 - SharedObjectBase, [752](#)
 - SharedQueue, [757](#)
 - SoundManager::SoundData, [803](#)
- data_received
 - RemoteProcess, [734](#)

- dataCasted
 - SharedObject, [750](#)
- dataCurrent
 - WAV, [972](#)
- dataEnd
 - WAV, [972](#)
- DataFromStationSID
 - PowerSourceID, [124](#)
- dataStart
 - WAV, [972](#)
- deactivate
 - BatteryCheckControl, [215](#)
 - ControlBase, [298](#)
 - Controller, [314](#)
 - LoadPostureControl, [512](#)
 - SaveWalkControl, [743](#)
- deactivateETID
 - EventBase, [362](#)
- debug_level
 - Config::main_config, [279](#)
- debuget.h, [1109](#)
 - _extractFilename, [1112](#)
 - ASSERT, [1111](#)
 - ASSERTFATAL, [1111](#)
 - ASSERTRET, [1111](#)
 - ASSERTRETVAL, [1111](#)
 - charhexout, [1112](#)
 - hexdigit, [1112](#)
 - hexout, [1112](#)
- decay
 - AGM, [156](#)
- decide
 - karmedbanditExp3, [471](#)
 - karmedbanditExp3.1, [475](#)
- DecideNode
 - BanditMachine::DecideNode, [181](#)
- decode
 - LoadSave, [522–527](#)
- def
 - ValueSetControl, [890](#)
- def_buf_in_size
 - basic_netbuf, [207](#)
- def_buf_out_size
 - basic_netbuf, [207](#)
- DefaultPIDs
 - ERS210Info, [47](#)
 - ERS220Info, [75](#)
 - ERS2xxInfo, [101](#)
- DEG
 - Kinematics.h, [1209](#)
- del
 - SoundManagerMsg, [807](#)
- delay
 - EventRouter::TimerEntry, [402](#)
- DeleteLarge
 - SystemUtility.h, [1353](#)
- deleteMotion
 - MotionManagerMsg, [610](#)
- depth
 - _dm_cell, [151](#)
- description
 - ControlBase, [304](#)
- diff
 - PostureEngine, [684](#)
- dirty
 - HeadPointerMC, [455](#)
 - LedEngine, [487](#)
 - PIDMC, [676](#)
 - PostureMC, [697](#)
 - RemoteControllerMC, [731](#)
- dirtyTime
 - LedEngine, [487](#)
- disable
 - CompareTrans, [267](#)
 - TimeOutTrans, [868](#)
 - Transition, [877](#)
 - VisualTargetCloseTrans, [931](#)
- disableEvents
 - Vision, [908](#)
- disableGPA
 - WorldModel2, [989](#)
- disableIR
 - WorldModel2, [989](#)
- disableKludge
 - WorldModel2, [989](#)
- DischargingSID
 - PowerSourceID, [124](#)
- display
 - Controller, [319](#)
- display_id
 - ControlBase, [304](#)

- displayNumber
 - LedEngine, [483](#)
- displayPercent
 - LedEngine, [484](#)
- dispTimers
 - EventRouter, [386](#)
- distance
 - _afsRRP, [148](#)
 - GVector, [117](#)
 - mathutils, [121](#)
 - VisionInterface::VObject, [933](#)
- distance_to_line
 - GVector, [118](#)
- DM
 - almMain.cc, [1055](#)
- dm_cell
 - almStructures.h, [1058](#)
- DM_CELL_COUNT
 - Maps/Configuration.h, [1101](#)
- dm_index2angles
 - almUtility.cc, [1060](#)
 - almUtility.h, [1063](#)
- dm_port
 - Config::worldmodel2_config, [291](#)
- dmPickCluster, [322](#)
- dmPickCluster
 - operator(), [322](#)
- dmPickColor, [323](#)
- dmPickColor
 - operator(), [323](#)
- dmPickConfidence, [324](#)
- dmPickConfidence
 - operator(), [324](#)
- dmPickDepth, [325](#)
- dmPickDepth
 - operator(), [325](#)
- dmPicker, [326](#)
- dmPicker
 - operator(), [326](#)
- DMS
 - almMain.cc, [1055](#)
- do_try_lock
 - MutexLock, [644](#)
- doCancel
 - ControlBase, [299](#)
- DoDestroy
 - MMCombo, [562](#)
 - RemoteProcess, [734](#)
 - SoundPlay, [813](#)
- DoInit
 - MMCombo, [562](#)
 - RemoteProcess, [734](#)
 - SoundPlay, [813](#)
- doKalmanUpdate
 - afsMeasurementUpdate.cc, [1035](#)
- done
 - SharedQueue, [757](#)
- doNextItem
 - ControlBase, [299](#)
 - NullControl, [655](#)
 - ValueEditControl, [882](#)
- door_t
 - MutexLock::door_t, [650](#)
- doors
 - MutexLock, [647](#)
- doors_used
 - MutexLock, [647](#)
- doPrevItem
 - ControlBase, [299](#)
 - NullControl, [656](#)
 - ValueEditControl, [882](#)
- doPriming
 - afsMeasurementUpdate.cc, [1036](#)
- doReadStdIn
 - ControlBase, [299](#)
 - NullControl, [656](#)
 - StringInputControl, [846](#)
- doRewrite
 - ControlBase, [304](#)
- doSelect
 - BatteryCheckControl, [215](#)
 - ControlBase, [300](#)
 - EventLogger, [375](#)
 - FileBrowserControl, [421](#)
 - MCValueEditControl, [548](#)
 - NullControl, [656](#)
 - RebootControl, [724](#)
 - ShutdownControl, [762](#)
 - ValueEditControl, [882](#)
- doSendBuffer
 - EventRouter, [386](#)
- doSendBufferLock

- EventRouter, 393
- doSendEvent
 - EventRouter, 387
- doSendSound
 - SoundPlay, 813
- DoStart
 - Aibo3DControllerBehavior, 160
 - AutoGetupBehavior, 173
 - BanditMachine, 177
 - BanditMachine::DecideNode, 181
 - BanditMachine::PressNode, 184
 - BanditMachine::WaitNode, 187
 - BatteryMonitorBehavior, 220
 - BehaviorBase, 230
 - CameraBehavior, 256
 - ChaseBallBehavior, 262
 - Controller, 314
 - DriveMeBehavior, 329
 - DumbWM2Behavior, 333
 - EStopControllerBehavior, 354
 - EvtRptBehavior, 413
 - FollowHeadBehavior, 427
 - FreeMemReportControl, 432
 - HeadLevelBehavior, 438
 - HeadPointControllerBehavior, 443
 - MMCombo, 562
 - MotionCommand, 575
 - OutputNode, 667
 - RemoteProcess, 734
 - SimpleChaseBallBehavior, 764
 - SoundPlay, 814
 - SoundTestBehavior, 820
 - StareAtBallBehavior, 827
 - StartupBehavior, 832
 - StateNode, 839
 - ToggleHeadLightBehavior, 873
 - WalkControllerBehavior, 938
 - WalkMC, 949
 - WalkToTargetMachine, 967
 - WorldModel2Behavior, 998
 - WorldModel2Behavior::Gawk-Node, 1001
 - WorldModel2Behavior::Wait-Node, 1005
 - WorldModel2Behavior::Walk-Node, 1009
- DoStop
 - Aibo3DControllerBehavior, 160
 - AutoGetupBehavior, 173
 - BanditMachine, 177
 - BanditMachine::PressNode, 184
 - BanditMachine::WaitNode, 187
 - BatteryMonitorBehavior, 220
 - BehaviorBase, 230
 - CameraBehavior, 256
 - ChaseBallBehavior, 262
 - Controller, 314
 - DriveMeBehavior, 329
 - DumbWM2Behavior, 333
 - EStopControllerBehavior, 354
 - EvtRptBehavior, 413
 - FollowHeadBehavior, 427
 - FreeMemReportControl, 432
 - HeadLevelBehavior, 438
 - HeadPointControllerBehavior, 443
 - MMCombo, 563
 - MotionCommand, 575
 - RemoteProcess, 734
 - SimpleChaseBallBehavior, 764
 - SoundPlay, 814
 - SoundTestBehavior, 820
 - StareAtBallBehavior, 827
 - StartupBehavior, 832
 - StateNode, 839
 - ToggleHeadLightBehavior, 873
 - WalkControllerBehavior, 938
 - WalkMC, 949
 - WalkToTargetMachine, 967
 - WorldModel2Behavior, 998
 - WorldModel2Behavior::Gawk-Node, 1001
 - WorldModel2Behavior::Wait-Node, 1005
 - WorldModel2Behavior::Walk-Node, 1009
- dot
 - GVector, 118
 - GVector::vector2d, 892
 - GVector::vector3d, 898

- down_time
 - WalkMC::LegParam, [959](#)
- down_vel
 - WalkMC::LegParam, [959](#)
- DriveMeBehavior, [327](#)
 - DriveMeBehavior, [329](#)
- DriveMeBehavior
 - ~DriveMeBehavior, [329](#)
 - DoStart, [329](#)
 - DoStop, [329](#)
 - DriveMeBehavior, [329](#)
 - getClassDescription, [329](#)
 - getName, [330](#)
 - last_da, [330](#)
 - last_dx, [330](#)
 - last_dy, [330](#)
 - last_time, [331](#)
 - processEvent, [330](#)
 - stand, [331](#)
 - stand_id, [331](#)
 - walker_id, [331](#)
- DriveMeBehavior.cc, [1113](#)
- DriveMeBehavior.h, [1115](#)
- dst
 - Transition, [877](#)
- DtoR
 - ChaseBallBehavior.cc, [1084](#)
 - StareAtBallBehavior.cc, [1338](#)
 - WalkToTargetMachine.cc, [1413](#)
- DumbWM2Behavior, [332](#)
 - DumbWM2Behavior, [333](#)
- DumbWM2Behavior
 - ~DumbWM2Behavior, [333](#)
 - DoStart, [333](#)
 - DoStop, [333](#)
 - DumbWM2Behavior, [333](#)
 - getClassDescription, [334](#)
 - getName, [334](#)
 - processEvent, [334](#)
 - WM2, [334](#)
- DumbWM2Behavior.h, [1117](#)
- dump
 - AGM, [156](#)
- dumpDM
 - ALM, [167](#)
- DumpFileControl, [336](#)
 - DumpFileControl, [336](#)
- DumpFileControl
 - ~DumpFileControl, [337](#)
 - DumpFileControl, [336](#)
 - selectedFile, [337](#)
- DumpFileControl.h, [1119](#)
- dumpHM
 - ALM, [167](#)
- duration
 - EmergencyStopMC, [350](#)
 - EventBase, [370](#)
- dx
 - _FastSLAM_update, [153](#)
 - motionReshapeKludge.h, [1254](#)
 - WalkControllerBehavior, [942](#)
 - WorldModel2, [993](#)
- dx0
 - SplinePath, [824](#)
- dx1
 - SplinePath, [824](#)
- DX_MOT_THRESH
 - motionReshapeKludge.h, [1252](#)
- dy
 - _FastSLAM_update, [153](#)
 - WalkControllerBehavior, [942](#)
 - WorldModel2, [993](#)
- DynamicMotionSequence, [338](#)
 - DynamicMotionSequence, [340](#)
- DynamicMotionSequence
 - ~DynamicMotionSequence, [340](#)
 - clear, [340](#)
 - DynamicMotionSequence, [340](#)
 - erased, [343](#)
 - eraseKeyFrame, [341](#)
 - getKeyFrame, [341](#)
 - getMaxFrames, [341](#)
 - getUsedFrames, [341](#)
 - list_t, [340](#)
 - moves, [343](#)
 - newKeyFrame, [342](#)
 - setRange, [342](#)
 - updateOutputs, [342](#)
- DynamicMotionSequence.h, [1121](#)
- EarOffset
 - ERS210Info, [48](#)

- ERS2xxInfo, 102
- edge
 - VisionInterface::VObject, 933
- el
 - EventRouter::TimerEntry, 402
- elapsed
 - Profiler::Timer, 716
- ElevatorOffset
 - ERS210Info, 41
 - ERS220Info, 69
 - ERS2xxInfo, 98
- EmergencyStopMC, 344
 - EmergencyStopMC, 346
- EmergencyStopMC
 - ~EmergencyStopMC, 347
 - active, 350
 - duration, 350
 - EmergencyStopMC, 346
 - freezeJoints, 347
 - getActive, 347
 - getDbtTapDuration, 347
 - getResetSensitivity, 347
 - getStopped, 348
 - ledengine, 350
 - paused, 350
 - period, 350
 - pidcutoff, 350
 - piddutyavgs, 350
 - releaseJoints, 348
 - setActive, 348
 - setDbtTapDuration, 348
 - setResetSensitivity, 348
 - setStopped, 349
 - stilldown, 351
 - takeSnapshot, 349
 - timeoflastbtn, 351
 - timeoflastfreeze, 351
 - timeofthisbtn, 351
 - updateOutputs, 349
- EmergencyStopMC.cc, 1123
- EmergencyStopMC.h, 1125
- empty
 - _afsLastObservation, 142
 - ListMemBuf, 501
- enable
 - CompareTrans, 268
 - TimeOutTrans, 868
 - Transition, 877
 - VisualTargetCloseTrans, 931
- enabledGPA
 - WorldModel2, 993
- enabledIR
 - WorldModel2, 994
- enableEvents
 - Vision, 908
- enableGPA
 - WorldModel2, 989
- enableIR
 - WorldModel2, 990
- enableKludge
 - WorldModel2, 990
- encode
 - LoadSave, 527–532
 - Serializer, 745
- encodeDoublesAsFloats
 - Serializer, 746
- encodeVisionRaw
 - VisionSerializer, 928
- encodeVisionRLE
 - VisionSerializer, 928
- encodeVisionRun
 - VisionSerializer, 929
- end
 - ListMemBuf, 501
 - MotionManager, 589
- endPlay
 - SoundManager, 790
- endpoint
 - Socket, 778
- endtime
 - MotionSequence, 630
 - SoundTestBehavior, 821
- Enqueue
 - SoundManager, 789
- enqueue
 - EventTranslator, 408
- entries
 - ListMemBuf, 507
 - SharedQueue, 758
- entry.h, 1127
- entry_t
 - ListMemBuf::entry_t, 509

- eof
 - char_traits, [260](#)
- EPS
 - Spline.h, [1336](#)
- EQ
 - CompareTrans, [267](#)
- equalOrLongerThan
 - EventBase, [363](#)
- equalOrShorterThan
 - EventBase, [363](#)
- erase
 - ListMemBuf, [501](#)
- erased
 - DynamicMotionSequence, [343](#)
- eraseKeyFrame
 - DynamicMotionSequence, [341](#)
 - MotionSequence, [622](#)
 - MotionSequenceMC, [638](#)
- erouter
 - EventRouter.cc, [1146](#)
 - EventRouter.h, [1150](#)
- error_level
 - Config::main_config, [279](#)
- errors
 - Kinematics.cc, [1203](#)
- ErrorSID
 - PowerSourceID, [124](#)
- ERS210Info, [31](#)
 - AllLEDMask, [42](#)
 - BAccelOffset, [42](#)
 - BackButOffset, [39](#)
 - BackLEDMask, [42](#)
 - BinJointOffset, [43](#)
 - BotLLEDMask, [43](#)
 - BotLLEDOffset, [40](#)
 - BotRLEDMask, [43](#)
 - BotRLEDOffset, [40](#)
 - ButtonOffset_t, [39](#)
 - ChinButOffset, [39](#)
 - CPCJointLFElevator, [43](#)
 - CPCJointLFKnee, [43](#)
 - CPCJointLFRotator, [43](#)
 - CPCJointLHElevator, [43](#)
 - CPCJointLHKnee, [44](#)
 - CPCJointLHRotator, [44](#)
 - CPCJointMouth, [44](#)
 - CPCJointNeckPan, [44](#)
 - CPCJointNeckRoll, [44](#)
 - CPCJointNeckTilt, [44](#)
 - CPCJointRFElevator, [44](#)
 - CPCJointRFKnee, [45](#)
 - CPCJointRFRotator, [45](#)
 - CPCJointRHElevator, [45](#)
 - CPCJointRHKnee, [45](#)
 - CPCJointRHRotator, [45](#)
 - CPCJointTailPan, [45](#)
 - CPCJointTailTilt, [45](#)
 - CPCSensorAccelFB, [46](#)
 - CPCSensorAccelLR, [46](#)
 - CPCSensorAccelUD, [46](#)
 - CPCSensorBackSwitch, [46](#)
 - CPCSensorChinSwitch, [46](#)
 - CPCSensorHeadBackPressure, [46](#)
 - CPCSensorHeadFrontPressure, [46](#)
 - CPCSensorLFPaw, [47](#)
 - CPCSensorLHPaw, [47](#)
 - CPCSensorPSD, [47](#)
 - CPCSensorRFPaw, [47](#)
 - CPCSensorRHPaw, [47](#)
 - CPCSensorThermoSensor, [47](#)
 - DAccelOffset, [42](#)
 - DefaultPIDs, [47](#)
 - EarOffset, [48](#)
 - ElevatorOffset, [41](#)
 - FaceBackLeftLEDOffset, [40](#)
 - FaceBackRightLEDOffset, [40](#)
 - FaceCenterLeftLEDOffset, [40](#)
 - FaceCenterRightLEDOffset, [40](#)
 - FaceFrontLeftLEDOffset, [40](#)
 - FaceFrontRightLEDOffset, [40](#)
 - FaceLEDMask, [48](#)
 - FrameTime, [48](#)
 - HeadBkButOffset, [39](#)
 - HeadFrButOffset, [39](#)
 - HeadLEDMask, [48](#)
 - HeadOffset, [49](#)
 - IRDistOffset, [42](#)
 - IsFastOutput, [49](#)
 - IsRealERS210, [49](#)
 - JointsPerLeg, [49](#)

KneeOffset, [41](#)
LAccelOffset, [42](#)
LBkLegOffset, [40](#)
LBkLegOrder, [41](#)
LBkPawOffset, [39](#)
LEDBitMask_t, [39](#)
LEDOffset, [49](#)
LEDOffset_t, [39](#)
LegOffset, [49](#)
LegOffset_t, [40](#)
LegOrder_t, [40](#)
LFrLegOffset, [40](#)
LFrLegOrder, [41](#)
LFrPawOffset, [39](#)
MaxOutputSpeed, [50](#)
MaxRange, [41](#)
mechanicalLimits, [50](#)
MidLLEDMask, [51](#)
MidLLEDOffset, [40](#)
MidRLEDMask, [51](#)
MidRLEDOffset, [40](#)
MinMaxRange_t, [41](#)
MinRange, [41](#)
ModelLEDOffset, [40](#)
MouthOffset, [51](#)
NumBinJoints, [51](#)
NumButtons, [51](#)
NumEarJoints, [52](#)
NumFrames, [52](#)
NumHeadJoints, [52](#)
NumLEDs, [52](#)
NumLegJoints, [52](#)
NumLegs, [52](#)
NumMouthJoints, [52](#)
NumOutputs, [52](#)
NumPIDJoints, [53](#)
NumSensors, [53](#)
NumSlowFrames, [53](#)
NumTailJoints, [53](#)
outputNameLen, [53](#)
outputNames, [53](#)
outputRanges, [53](#)
PanOffset, [42](#)
PIDJointOffset, [54](#)
PowerCapacityOffset, [42](#)
PowerCurrentOffset, [42](#)
PowerRemainOffset, [42](#)
PowerThermoOffset, [42](#)
PowerVoltageOffset, [42](#)
PrimitiveName, [54](#)
RBkLegOffset, [40](#)
RBkLegOrder, [41](#)
RBkPawOffset, [39](#)
REKOffset_t, [41](#)
RFrLegOffset, [40](#)
RFrLegOrder, [41](#)
RFrPawOffset, [39](#)
RollOffset, [42](#)
RotatorOffset, [41](#)
SensorOffset_t, [41](#)
SlowFrameTime, [55](#)
SoundBufferTime, [55](#)
TailLEDMask, [56](#)
TailLeftLEDOffset, [40](#)
TailOffset, [56](#)
TailRightLEDOffset, [40](#)
ThermoOffset, [42](#)
TiltOffset, [42](#)
TIBluLEDMask, [56](#)
TIBluLEDOffset, [40](#)
TIRedLEDMask, [56](#)
TIRedLEDOffset, [40](#)
TopBrLEDMask, [56](#)
TopBrLEDOffset, [40](#)
TopLLEDMask, [56](#)
TopLLEDOffset, [40](#)
TopRLEDMask, [57](#)
TopRLEDOffset, [40](#)
TPROffset_t, [42](#)
ERS210Info.h, [1128](#)
 _RI_RAD_FLAG, [1129](#)
 RAD, [1129](#)
ERS210Mask
 WorldState, [1016](#)
ERS210numMasks
 LedEngine, [487](#)
ERS220Info, [58](#)
 AllLEDMask, [70](#)
 BAccelOffset, [69](#)
 BackButOffset, [67](#)
 BackLEDMask, [70](#)
 BackLeft1LEDMask, [70](#)

BackLeft1LEDOffset, [67](#)
BackLeft2LEDMask, [71](#)
BackLeft2LEDOffset, [67](#)
BackLeft3LEDMask, [71](#)
BackLeft3LEDOffset, [67](#)
BackRight1LEDMask, [71](#)
BackRight1LEDOffset, [67](#)
BackRight2LEDMask, [71](#)
BackRight2LEDOffset, [67](#)
BackRight3LEDMask, [71](#)
BackRight3LEDOffset, [67](#)
BinJointOffset, [71](#)
BotLLEDMask, [72](#)
BotLLEDOffset, [67](#)
BotRLEDMask, [72](#)
BotRLEDOffset, [68](#)
ButtonOffset.t, [66](#)
ChinButOffset, [66](#)
CPCJointLFElevator, [72](#)
CPCJointLFKnee, [72](#)
CPCJointLFRotator, [72](#)
CPCJointLHElevator, [72](#)
CPCJointLHKnee, [72](#)
CPCJointLHRotator, [72](#)
CPCJointNeckPan, [72](#)
CPCJointNeckRoll, [73](#)
CPCJointNeckTilt, [73](#)
CPCJointRFElevator, [73](#)
CPCJointRFKnee, [73](#)
CPCJointRFRotator, [73](#)
CPCJointRHElevator, [73](#)
CPCJointRHKnee, [73](#)
CPCJointRHRotator, [73](#)
CPCSensorAccelFB, [73](#)
CPCSensorAccelLR, [73](#)
CPCSensorAccelUD, [74](#)
CPCSensorBackSwitch, [74](#)
CPCSensorChinSwitch, [74](#)
CPCSensorHeadBackPressure,
[74](#)
CPCSensorHeadFrontPressure,
[74](#)
CPCSensorLFPaw, [74](#)
CPCSensorLHPaw, [74](#)
CPCSensorPSD, [74](#)
CPCSensorRFPaw, [74](#)
CPCSensorRHPaw, [74](#)
CPCSensorTailCenterSwitch, [75](#)
CPCSensorTailLeftSwitch, [75](#)
CPCSensorTailRightSwitch, [75](#)
CPCSensorThermoSensor, [75](#)
DAccelOffset, [69](#)
DefaultPIDs, [75](#)
ElevatorOffset, [69](#)
FaceBackLeftLEDMask, [76](#)
FaceBackLeftLEDOffset, [67](#)
FaceBackRightLEDMask, [76](#)
FaceBackRightLEDOffset, [67](#)
FaceCenterLeftLEDMask, [76](#)
FaceCenterLeftLEDOffset, [67](#)
FaceCenterRightLEDMask, [76](#)
FaceCenterRightLEDOffset, [67](#)
FaceFrontALEDMask, [76](#)
FaceFrontALEDOffset, [67](#)
FaceFrontBLEDMask, [76](#)
FaceFrontBLEDOffset, [67](#)
FaceFrontCLEDMask, [76](#)
FaceFrontCLEDOffset, [67](#)
FaceFrontLeftLEDMask, [77](#)
FaceFrontLeftLEDOffset, [67](#)
FaceFrontRightLEDMask, [77](#)
FaceFrontRightLEDOffset, [67](#)
FaceLEDMask, [77](#)
FrameTime, [77](#)
HeadBkButOffset, [67](#)
HeadFrButOffset, [67](#)
HeadLEDMask, [77](#)
HeadOffset, [78](#)
IRDistOffset, [69](#)
IsFastOutput, [78](#)
IsRealERS220, [78](#)
JointsPerLeg, [79](#)
KneeOffset, [69](#)
LAccelOffset, [69](#)
LBkLegOffset, [68](#)
LBkLegOrder, [68](#)
LBkPawOffset, [66](#)
LEDBitMask.t, [66](#)
LEDOffset, [79](#)
LEDOffset.t, [67](#)
LegOffset, [79](#)
LegOffset.t, [68](#)

- LegOrder_t, 68
- LFrLegOffset, 68
- LFrLegOrder, 68
- LFrPawOffset, 66
- MaxOutputSpeed, 79
- MaxRange, 69
- mechanicalLimits, 80
- MidLLEDMask, 81
- MidLLEDOffset, 68
- MidRLEDMask, 81
- MidRLEDOffset, 68
- MinMaxRange_t, 68
- MinRange, 69
- ModeLEDMask, 81
- ModeLEDOffset, 67
- NumBinJoints, 81
- NumButtons, 81
- NumEarJoints, 81
- NumFrames, 81
- NumHeadJoints, 82
- NumLEDs, 82
- NumLegJoints, 82
- NumLegs, 82
- NumMouthJoints, 82
- NumOutputs, 82
- NumPIDJoints, 82
- NumSensors, 83
- NumSlowFrames, 83
- NumTailJoints, 83
- outputNameLen, 83
- outputNames, 83
- outputRanges, 83
- PanOffset, 70
- PIDJointOffset, 84
- PowerCapacityOffset, 69
- PowerCurrentOffset, 70
- PowerRemainOffset, 69
- PowerThermoOffset, 69
- PowerVoltageOffset, 70
- PrimitiveName, 84
- RBkLegOffset, 68
- RBkLegOrder, 68
- RBkPawOffset, 66
- REKOffset_t, 69
- RetractableHeadLEDMask, 85
- RetractableHeadLEDOffset, 67
- RFrLegOffset, 68
- RFrLegOrder, 68
- RFrPawOffset, 66
- RollOffset, 70
- RotatorOffset, 69
- SensorOffset_t, 69
- SlowFrameTime, 85
- SoundBufferTime, 86
- TailCenterButOffset, 67
- TailCenterLEDMask, 86
- TailCenterLEDOffset, 67
- TailLEDMask, 86
- TailLeftButOffset, 67
- TailLeftLEDMask, 86
- TailLeftLEDOffset, 67
- TailRightButOffset, 67
- TailRightLEDMask, 86
- TailRightLEDOffset, 67
- ThermoOffset, 69
- TiltOffset, 70
- TIBluLEDMask, 86
- TIBluLEDOffset, 68
- TIRedLEDMask, 87
- TIRedLEDOffset, 68
- TopBrLEDMask, 87
- TopBrLEDOffset, 68
- TopLLEDMask, 87
- TopLLEDOffset, 68
- TopRLEDMask, 87
- TopRLEDOffset, 68
- TPROffset_t, 70
- ERS220Info.h, 1130
- __RI_RAD_FLAG, 1131
- RAD, 1131
- ERS220Mask
 - WorldState, 1017
- ERS220numMasks
 - LedEngine, 488
- ERS2xxInfo, 88
 - BAccelOffset, 99
 - BackButOffset, 96
 - BackLeft1LEDOffset, 97
 - BackLeft2LEDOffset, 97
 - BackLeft3LEDOffset, 97
 - BackRight1LEDOffset, 97
 - BackRight2LEDOffset, 97

- BackRight3LEDOffset, [97](#)
- BotLLEDOffset, [97](#)
- BotRLEDOffset, [97](#)
- ChinButOffset, [96](#)
- DAccelOffset, [99](#)
- ElevatorOffset, [98](#)
- FaceBackLeftLEDOffset, [97](#)
- FaceBackRightLEDOffset, [97](#)
- FaceCenterLeftLEDOffset, [96](#)
- FaceCenterRightLEDOffset, [96](#)
- FaceFrontALEDOffset, [97](#)
- FaceFrontBLEDOffset, [97](#)
- FaceFrontCLEDOffset, [97](#)
- FaceFrontLeftLEDOffset, [96](#)
- FaceFrontRightLEDOffset, [96](#)
- HeadBkButOffset, [96](#)
- HeadFrButOffset, [96](#)
- IRDistOffset, [99](#)
- KneeOffset, [98](#)
- LAccelOffset, [99](#)
- LBkLegOffset, [97](#)
- LBkLegOrder, [98](#)
- LBkPawOffset, [96](#)
- LFrLegOffset, [97](#)
- LFrLegOrder, [98](#)
- LFrPawOffset, [96](#)
- MaxRange, [98](#)
- MidLLEDOffset, [97](#)
- MidRLEDOffset, [97](#)
- MinRange, [98](#)
- ModeLEDOffset, [97](#)
- PanOffset, [99](#)
- PowerCapacityOffset, [99](#)
- PowerCurrentOffset, [99](#)
- PowerRemainOffset, [99](#)
- PowerThermoOffset, [99](#)
- PowerVoltageOffset, [99](#)
- RBkLegOffset, [97](#)
- RBkLegOrder, [98](#)
- RBkPawOffset, [96](#)
- RetractableHeadLEDOffset, [97](#)
- RFrLegOffset, [97](#)
- RFrLegOrder, [98](#)
- RFrPawOffset, [96](#)
- RollOffset, [99](#)
- RotatorOffset, [98](#)
- TailCenterButOffset, [96](#)
- TailCenterLEDOffset, [97](#)
- TailLeftButOffset, [96](#)
- TailLeftLEDOffset, [97](#)
- TailRightButOffset, [96](#)
- TailRightLEDOffset, [97](#)
- ThermoOffset, [99](#)
- TiltOffset, [99](#)
- TIBluLEDOffset, [97](#)
- TIRedLEDOffset, [97](#)
- TopBrLEDOffset, [97](#)
- TopLLEDOffset, [97](#)
- TopRLEDOffset, [97](#)
- ERS2xxInfo
 - AllLEDMask, [99](#)
 - BackLEDMask, [99](#)
 - BackLeft1LEDMask, [100](#)
 - BackLeft2LEDMask, [100](#)
 - BackLeft3LEDMask, [100](#)
 - BackRight1LEDMask, [100](#)
 - BackRight2LEDMask, [100](#)
 - BackRight3LEDMask, [101](#)
 - BinJointOffset, [101](#)
 - BotLLEDMask, [101](#)
 - BotRLEDMask, [101](#)
 - ButtonOffset.t, [96](#)
 - DefaultPIDs, [101](#)
 - EarOffset, [102](#)
 - FaceBackLeftLEDMask, [102](#)
 - FaceBackRightLEDMask, [102](#)
 - FaceCenterLeftLEDMask, [102](#)
 - FaceCenterRightLEDMask, [102](#)
 - FaceFrontALEDMask, [103](#)
 - FaceFrontBLEDMask, [103](#)
 - FaceFrontCLEDMask, [103](#)
 - FaceFrontLeftLEDMask, [103](#)
 - FaceFrontRightLEDMask, [103](#)
 - FaceLEDMask, [103](#)
 - FrameTime, [104](#)
 - HeadLEDMask, [104](#)
 - HeadOffset, [104](#)
 - IsFastOutput, [104](#)
 - IsRealERS210, [105](#)
 - IsRealERS220, [105](#)
 - JointsPerLeg, [106](#)
 - LEDBitMask.t, [95](#)

- LEDOffset, [106](#)
- LEDOffset_t, [96](#)
- LegOffset, [106](#)
- LegOffset_t, [97](#)
- LegOrder_t, [97](#)
- MaxOutputSpeed, [106](#)
- mechanicalLimits, [107](#)
- MidLEDMask, [107](#)
- MidRLEDMask, [107](#)
- MinMaxRange_t, [98](#)
- ModeLEDMask, [108](#)
- MouthOffset, [108](#)
- NumBinJoints, [108](#)
- NumButtons, [108](#)
- NumEarJoints, [108](#)
- NumFrames, [108](#)
- NumHeadJoints, [108](#)
- NumLEDs, [109](#)
- NumLegJoints, [109](#)
- NumLegs, [109](#)
- NumMouthJoints, [109](#)
- NumOutputs, [109](#)
- NumPIDJoints, [109](#)
- NumSensors, [109](#)
- NumSlowFrames, [110](#)
- NumTailJoints, [110](#)
- outputNameLen, [110](#)
- outputNames, [110](#)
- outputRanges, [110](#)
- PIDJointOffset, [111](#)
- PrimitiveName, [111](#)
- REKOffset_t, [98](#)
- RetractableHeadLEDMask, [112](#)
- SensorOffset_t, [98](#)
- SlowFrameTime, [113](#)
- SoundBufferTime, [113](#)
- TailCenterLEDMask, [113](#)
- TailLEDMask, [113](#)
- TailLeftLEDMask, [113](#)
- TailOffset, [113](#)
- TailRightLEDMask, [114](#)
- TIBluLEDMask, [114](#)
- TIRedLEDMask, [114](#)
- TopBrLEDMask, [114](#)
- TopLLEDMask, [114](#)
- TopRLEDMask, [114](#)
- TPROffset_t, [99](#)
- ERS2xxInfo.h, [1132](#)
- ERS2xxInfo.h
 - __RI_RAD_FLAG, [1133](#)
 - RAD, [1133](#)
- estop_id
 - Controller, [319](#)
 - EStopControllerBehavior, [356](#)
- estop_off_snd
 - Config::motion_config, [283](#)
- estop_on_snd
 - Config::motion_config, [283](#)
- estopControl_port
 - Config::main_config, [280](#)
- EStopControllerBehavior, [352](#)
 - EStopControllerBehavior, [353](#), [354](#)
- EStopControllerBehavior
 - ~EStopControllerBehavior, [354](#)
 - callback, [354](#)
 - cmdsock, [356](#)
 - DoStart, [354](#)
 - DoStop, [354](#)
 - estop_id, [356](#)
 - EStopControllerBehavior, [353](#), [354](#)
 - getClassDescription, [355](#)
 - getName, [355](#)
 - operator=, [355](#)
 - processEvent, [355](#)
 - runCommand, [355](#)
 - theOne, [356](#)
- EStopControllerBehavior.cc, [1134](#)
- EStopControllerBehavior.h, [1135](#)
- estopEGID
 - EventBase, [362](#)
- estopid
 - LoadPostureControl, [513](#)
 - RunSequenceControl, [739](#)
- etrans
 - MMCombo, [568](#)
 - SoundPlay, [816](#)
- eval
 - HermiteSplineSegment, [461](#)
 - NonUniformHermiteSplineSegment, [653](#)

- SplinePath, 824
- eval_deriv
 - HermiteSplineSegment, 461
 - NonUniformHermiteSplineSegment, 653
 - SplinePath, 824
- EventBase, 357
 - activateETID, 362
 - aiEGID, 361
 - audioEGID, 361
 - buttonEGID, 361
 - deactivateETID, 362
 - estopEGID, 362
 - EventBase, 362, 363
 - locomotionEGID, 362
 - motmanEGID, 362
 - numEGIDs, 362
 - numETIDs, 362
 - powerEGID, 362
 - sensorEGID, 361
 - stateMachineEGID, 362
 - statusETID, 362
 - textmsgEGID, 362
 - timerEGID, 362
 - unknownEGID, 361
 - visionEGID, 361
 - worldModelEGID, 361
- EventBase
 - ~EventBase, 363
 - duration, 370
 - equalOrLongerThan, 363
 - equalOrShorterThan, 363
 - EventBase, 362, 363
 - EventGeneratorID.t, 361
 - EventGeneratorNames, 370
 - EventTypeID.t, 362
 - genID, 370
 - genName, 364
 - getBinSize, 364
 - getDuration, 364
 - getGeneratorID, 364
 - getMagnitude, 365
 - getName, 365
 - getSourceID, 365
 - getTimeStamp, 365
 - getTypeID, 366
 - isCustomName, 366
 - LoadBuffer, 366
 - longerThan, 366
 - magnitude, 370
 - nameisgen, 371
 - operator<, 367
 - operator==, 367
 - resetName, 367
 - sameGenSource, 367
 - SaveBuffer, 367
 - setDuration, 368
 - setGeneratorID, 368
 - setMagnitude, 368
 - setName, 369
 - setSourceID, 369
 - setTypeID, 369
 - shorterThan, 369
 - sourceID, 371
 - stim_id, 371
 - timestamp, 371
 - typeID, 371
- EventBase.cc, 1137
- EventBase.h, 1138
- EventBase_ID
 - EventTranslator, 407
- EventGeneratorID.t
 - EventBase, 361
- EventGeneratorNames
 - EventBase, 370
- EventListener, 372
- EventListener
 - ~EventListener, 372
 - processEvent, 373
- EventListener.h, 1140
- EventLogger, 374
 - EventLogger, 375
- EventLogger
 - checkLogFile, 375
 - doSelect, 375
 - EventLogger, 375
 - logfile, 376
 - logfilePath, 376
 - processEvent, 376
 - refresh, 376
 - setStatus, 376
 - verbosity, 377

- EventLogger.cc, [1142](#)
- EventLogger.h, [1143](#)
- EventManager
 - EventRouter::EventManager, [396](#)
- EventRouter, [378](#)
 - EventRouter, [383](#)
- EventRouter
 - ~EventRouter, [383](#)
 - addListener, [384](#)
 - addTimer, [384](#), [385](#)
 - addTrapper, [385](#), [386](#)
 - buffertime, [392](#)
 - chkTimers, [386](#)
 - dispTimers, [386](#)
 - doSendBuffer, [386](#)
 - doSendBufferLock, [393](#)
 - doSendEvent, [387](#)
 - EventRouter, [383](#)
 - events, [393](#)
 - forgetListener, [387](#)
 - getBufferTime, [387](#)
 - hasListeners, [387](#), [388](#)
 - lastBufClear, [393](#)
 - listeners, [393](#)
 - postEvent, [388](#)
 - processEvent, [388](#)
 - processEventBuffer, [389](#)
 - processTimers, [389](#)
 - removeAllTimers, [389](#)
 - removeListener, [389](#), [390](#)
 - removeTimer, [390](#), [391](#)
 - removeTrapper, [391](#), [392](#)
 - reset, [392](#)
 - setBufferTime, [392](#)
 - timer_it_t, [383](#)
 - timers, [393](#)
 - trappers, [393](#)
- EventRouter.cc, [1145](#)
- EventRouter.cc
 - erouter, [1146](#)
- EventRouter.h, [1147](#)
- EventRouter.h
 - erouter, [1150](#)
- EventRouter::EventManager, [394](#)
- EventRouter::EventManager
 - addMapping, [396](#)
 - allevents, [399](#)
 - clean, [397](#)
 - clear, [397](#)
 - EventManager, [396](#)
 - filteredevents, [399](#)
 - getMapping, [397](#)
 - hasMapping, [397](#), [398](#)
 - operator=, [398](#)
 - removeMapping, [398](#), [399](#)
 - SIDtoListenerVectorMap_t, [396](#)
- EventRouter::TimerEntry, [400](#)
- EventRouter::TimerEntry
 - delay, [402](#)
 - el, [402](#)
 - next, [402](#)
 - operator=, [401](#)
 - repeat, [402](#)
 - Set, [401](#)
 - sid, [402](#)
 - TimerEntry, [401](#)
- EventRouter::TimerEntryPtrCmp, [404](#)
- EventRouter::TimerEntryPtrCmp
 - operator(), [404](#)
- events
 - EventRouter, [393](#)
- EventTranslator, [405](#)
 - EventBase_ID, [407](#)
 - EventTranslator, [407](#)
 - LocomotionEvent_ID, [407](#)
 - TextMsgEvent_ID, [407](#)
 - VisionEvent_ID, [407](#)
- EventTranslator
 - enqueue, [408](#)
 - EventTranslator, [407](#)
 - operator=, [408](#)
 - queue, [409](#)
 - Queue_t, [407](#)
 - sendEvent, [408](#)
 - setQueue, [408](#)
 - translateEvents, [408](#)
 - trapEvent, [408](#)
 - TypeID_t, [407](#)
- EventTranslator.cc, [1151](#)
- EventTranslator.h, [1153](#)
- eventTranslatorQueueMemRgn
 - MMCombo, [568](#)

- SoundPlay, [816](#)
- EventTrapper, [410](#)
- EventTrapper
 - ~EventTrapper, [410](#)
 - trapEvent, [411](#)
- EventTrapper.h, [1155](#)
- EventTypeID.t
 - EventBase, [362](#)
- EvtRptBehavior, [412](#)
- EvtRptBehavior, [413](#)
- EvtRptBehavior
 - ~EvtRptBehavior, [413](#)
 - DoStart, [413](#)
 - DoStop, [413](#)
 - EvtRptBehavior, [413](#)
 - getClassDescription, [413](#)
 - getName, [414](#)
 - processEvent, [414](#)
- EvtRptBehavior.cc, [1157](#)
- EvtRptBehavior.h, [1158](#)
- execExpAvg
 - Profiler::SectionInfo, [712](#)
- execHist
 - Profiler::SectionInfo, [712](#)
- exp3
 - karmedbanditExp3_1, [476](#)
- ExternalPortSID
 - PowerSourceID, [124](#)
- ExternalPowerSID
 - PowerSourceID, [124](#)
- extractBool
 - Config, [272](#)
- f_body_to_shoulder
 - Kinematics.cc, [1201](#)
- f_knee_kmin
 - Kinematics.cc, [1203](#)
 - Kinematics.h, [1211](#)
- f_leg_knee_to_ball
 - Kinematics.cc, [1201](#)
- f_leg_shoulder_to_knee
 - Kinematics.cc, [1201](#)
- f_lower
 - Kinematics.cc, [1203](#)
- f_upper
 - Kinematics.cc, [1203](#)
- FaceBackLeftLEDMask
 - ERS220Info, [76](#)
 - ERS2xxInfo, [102](#)
- FaceBackLeftLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- FaceBackRightLEDMask
 - ERS220Info, [76](#)
 - ERS2xxInfo, [102](#)
- FaceBackRightLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- FaceCenterLeftLEDMask
 - ERS220Info, [76](#)
 - ERS2xxInfo, [102](#)
- FaceCenterLeftLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [96](#)
- FaceCenterRightLEDMask
 - ERS220Info, [76](#)
 - ERS2xxInfo, [102](#)
- FaceCenterRightLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [96](#)
- FaceFrontALEDMask
 - ERS220Info, [76](#)
 - ERS2xxInfo, [103](#)
- FaceFrontALEDOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- FaceFrontBLEDMask
 - ERS220Info, [76](#)
 - ERS2xxInfo, [103](#)
- FaceFrontBLEDOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- FaceFrontCLEDMask
 - ERS220Info, [76](#)
 - ERS2xxInfo, [103](#)
- FaceFrontCLEDOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)

- FaceFrontLeftLEDMask
 - ERS220Info, [77](#)
 - ERS2xxInfo, [103](#)
- FaceFrontLeftLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [96](#)
- FaceFrontRightLEDMask
 - ERS220Info, [77](#)
 - ERS2xxInfo, [103](#)
- FaceFrontRightLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [96](#)
- FaceLEDMask
 - ERS210Info, [48](#)
 - ERS220Info, [77](#)
 - ERS2xxInfo, [103](#)
- Factory, [415](#)
 - construct, [415](#)
- Factory.h, [1159](#)
- Factory1Arg, [417](#)
 - construct, [417](#)
- Fast
 - SoundManager, [788](#)
- FastSLAM/Configuration.h
 - AFS_COVARIANCE_FUDGE, [1094](#)
 - AFS_MAX_TRIANG_ANGLE, [1094](#)
 - AFS_MEASURE_VARIANCE, [1094](#)
 - AFS_MIN_TRIANG_ANGLE, [1095](#)
 - AFS_NUM_LANDMARKS, [1095](#)
 - AFS_NUM_PARTICLES, [1095](#)
 - AFS_PERTURB_DA_MEAN, [1095](#)
 - AFS_PERTURB_DA_-
VARIANCE, [1095](#)
 - AFS_PERTURB_DX_MEAN, [1095](#)
 - AFS_PERTURB_DX_-
VARIANCE, [1095](#)
 - AFS_PERTURB_DY_MEAN, [1095](#)
 - AFS_PERTURB_DY_-
VARIANCE, [1095](#)
 - AFS_VARIANCE_-
MULTIPLIER, [1095](#)
- FastSLAM_update
 - WorldModel2.h, [1430](#)
- fbuf
 - Aibo3DControllerBehavior, [162](#)
- FCFS_in_use
 - MutexLock::door_t, [650](#)
- file
 - LoadPostureControl, [513](#)
- FileBrowserControl, [418](#)
 - FileBrowserControl, [420](#)
- FileBrowserControl
 - activate, [420](#)
 - doSelect, [421](#)
 - FileBrowserControl, [420](#)
 - filter, [423](#)
 - getRecurse, [421](#)
 - getRoot, [421](#)
 - init, [421](#)
 - makePath, [421](#), [422](#)
 - match, [422](#)
 - paths, [423](#)
 - rebuildmenu, [422](#)
 - recurse, [423](#)
 - root, [424](#)
 - selectedFile, [422](#)
 - setFilter, [422](#)
 - setPath, [423](#)
 - setRecurse, [423](#)
 - setRoot, [423](#)
- FileBrowserControl.cc, [1160](#)
- FileBrowserControl.h, [1161](#)
- filter
 - FileBrowserControl, [423](#)
 - VisionEventSpec, [924](#)
- filteredevents
 - EventRouter::EventManager, [399](#)
- find_dtheta
 - afsUtility.cc, [1046](#)
 - afsUtility.h, [1047](#)
- findBall

- Vision, [908](#)
- FindFreeRegion
 - SoundPlay, [814](#)
- findGesture
 - Vision, [908](#)
- findHand
 - Vision, [908](#)
- findMarkers
 - Vision, [909](#)
- findSpan
 - Vision, [909](#)
- findThing
 - Vision, [909](#)
- finished
 - Profiler, [705](#)
- flash
 - LedEngine, [484](#)
- flashtime
 - LedEngine::LEDInfo, [490](#)
- flashvalue
 - LedEngine::LEDInfo, [490](#)
- flType
 - Socket, [778](#)
- flush
 - Socket, [774](#)
- FLUSH_BLOCKING
 - SocketNS, [129](#)
- FLUSH_NONBLOCKING
 - SocketNS, [129](#)
- FlushType_t
 - SocketNS, [128](#)
- fmodt
 - Util.h, [1372](#)
- FMTSIZE_WITHOUT_EXTINFO
 - WAV, [973](#)
- FocalDist
 - Vision.h, [1387](#)
- FocalDistV
 - Vision.h, [1387](#)
- FollowHeadBehavior, [425](#)
 - FollowHeadBehavior, [426](#)
- FollowHeadBehavior
 - ~FollowHeadBehavior, [426](#)
 - clock, [428](#)
 - DoStart, [427](#)
 - DoStop, [427](#)
 - FollowHeadBehavior, [426](#)
 - getClassDescription, [427](#)
 - getName, [427](#)
 - head_lock, [428](#)
 - head_release, [428](#)
 - processEvent, [428](#)
 - walker_id, [428](#)
- FollowHeadBehavior.cc, [1163](#)
- FollowHeadBehavior.h, [1165](#)
- forget
 - MutexLock, [645](#)
- forgetListener
 - EventRouter, [387](#)
- forwardSock
 - Socket, [779](#)
- frame_count
 - Vision, [914](#)
- frames
 - MotionManager::OutputState, [605](#)
- FrameTime
 - ERS210Info, [48](#)
 - ERS220Info, [77](#)
 - ERS2xxInfo, [104](#)
- frameTimestamp
 - Vision, [914](#)
- freeBack
 - ListMemBuf, [507](#)
- freeBegin
 - ListMemBuf, [507](#)
- freeMem
 - FreeMemReportControl, [433](#)
- FreeMemReportControl, [430](#)
 - FreeMemReportControl, [432](#)
- FreeMemReportControl
 - ~FreeMemReportControl, [432](#)
 - DoStart, [432](#)
 - DoStop, [432](#)
 - freeMem, [433](#)
 - FreeMemReportControl, [432](#)
 - getName, [433](#)
 - init, [433](#)
 - isWarning, [434](#)
 - low_mem, [434](#)
 - monitor_freq, [435](#)
 - processEvent, [433](#)

- refresh, [434](#)
- report, [434](#)
- report_freq, [435](#)
- resetTimerFreq, [434](#)
- FreeMemReportControl.cc, [1167](#)
- FreeMemReportControl.h, [1168](#)
- freezeJoints
 - EmergencyStopMC, [347](#)
- front
 - ListMemBuf, [501](#)
- fs_port
 - Config::worldmodel2_config, [291](#)
- FSUdeque
 - WorldModel2.h, [1430](#)
- func_begin
 - MotionManager, [589](#)
- func_end
 - MotionManager, [589](#)
- G
 - karmedbanditExp3_1, [476](#)
- g
 - karmedbanditExp3, [472](#)
 - SmoothCompareTrans, [768](#)
 - WorldState, [1017](#)
- gain
 - Config::vision_config, [287](#)
- gamma
 - AutoGetupBehavior, [174](#)
 - Profiler, [707](#)
- gaussian_constant
 - Util.h, [1374](#)
- gaussian_prob
 - Util.h, [1373](#)
- gaussian_with_min
 - Util.h, [1373](#)
- GawkNode
 - WorldModel2Behavior::GawkNode, [1001](#)
- generateEvent
 - Vision, [909](#)
- genID
 - EventBase, [370](#)
- genName
 - EventBase, [364](#)
- genRequests
 - AGM, [156](#)
 - ALM, [167](#)
- Geometry.h, [1170](#)
 - point2d, [1172](#)
 - point2f, [1172](#)
 - point3d, [1172](#)
 - point3f, [1172](#)
 - vector2d, [1172](#)
 - vector2f, [1172](#)
 - vector3d, [1172](#)
 - vector3f, [1172](#)
- get_camera_dir
 - Vision, [909](#)
- get_camera_loc
 - Vision, [909](#)
- get_lock_level
 - MutexLock, [645](#)
- get_longword
 - WAV, [971](#)
- get_time
 - get_time.cc, [1174](#)
 - get_time.h, [1176](#)
- get_time.cc, [1173](#)
- get_time, [1174](#)
- get_time.h, [1175](#)
- get_time, [1176](#)
- get_word
 - WAV, [971](#)
- getActive
 - EmergencyStopMC, [347](#)
 - HeadPointerMC, [451](#)
 - TailWagMC, [851](#)
- getAngle
 - WalkMC, [949](#)
- GetAutoDelete
 - ReferenceCounter, [726](#)
- getAutoPrune
 - MotionCommand, [575](#)
- GETB
 - WorldState.cc, [1438](#)
- getBinSize
 - EventBase, [364](#)
 - LoadSave, [532](#)
 - LocomotionEvent, [543](#)
 - MotionSequence, [622](#)
 - PostureEngine, [685](#)

- TextMsgEvent, 856
- VisionEvent, 920
- GetBitsPerSample
 - WAV, 971
- GetBodyLocation
 - Kinematics.cc, 1201
 - Kinematics.h, 1209
- getBucket
 - Profiler, 705
- getBuckets
 - Profiler, 705
- getBufferTime
 - EventRouter, 387
- getCenterX
 - VisionEvent, 920
- getCenterY
 - VisionEvent, 921
- getClassDescription
 - Aibo3DControllerBehavior, 161
 - Aibo3DMonitorBehavior, 165
 - AutoGetupBehavior, 173
 - BanditMachine, 177
 - BatteryMonitorBehavior, 220
 - BehaviorBase, 231
 - CameraBehavior, 256
 - ChaseBallBehavior, 263
 - Controller, 315
 - DriveMeBehavior, 329
 - DumbWM2Behavior, 334
 - EStopControllerBehavior, 355
 - EvtRptBehavior, 413
 - FollowHeadBehavior, 427
 - HeadLevelBehavior, 438
 - HeadPointControllerBehavior, 443
 - SoundTestBehavior, 820
 - StareAtBallBehavior, 828
 - StartupBehavior, 832
 - ToggleHeadLightBehavior, 873
 - WalkControllerBehavior, 939
 - WalkToTargetMachine, 967
 - WorldModel2Behavior, 998
- getColor
 - Vision, 910
- getColorUnsafe
 - Vision, 910
- getCopies
 - ValueEditControl, 882
- getCurVelocity
 - WalkMC, 950
- GETD
 - WorldState.cc, 1438
- GetDataEnd
 - WAV, 971
- GetDataStart
 - WAV, 971
- getDbtTapDuration
 - EmergencyStopMC, 347
- getDefault
 - ValueSetControl, 889
- getDescription
 - BehaviorBase, 231
 - BehaviorSwitchActivatorControl, 236
 - BehaviorSwitchControl, 241
 - ControlBase, 300
 - StateNode, 839
- getDisplay
 - ControlBase, 300
- getDistance
 - VisionEvent, 921
- getDM
 - ALM, 168
- getDuration
 - EventBase, 364
- GETDUTY
 - WorldState.cc, 1438
- getEcho
 - basic_iNetStream, 192
 - basic_ioNetStream, 197
 - basic_netbuf, 203
 - basic_oNetStream, 210
- getEndTime
 - MotionSequence, 622
- getGamma
 - karmedbanditExp3, 471
- getGeneratorID
 - EventBase, 364
- getGM
 - AGM, 157
- getGUIType
 - Aibo3DControllerBehavior, 161

- Aibo3DMonitorBehavior, 165
- GetHeadAngles
 - Kinematics.cc, 1201
 - Kinematics.h, 1209
- GetHeadPosition
 - Kinematics.cc, 1201
 - Kinematics.h, 1209
- getHeight
 - Vision, 910
 - WalkMC, 950
- getHighlights
 - ControlBase, 301
- getHM
 - ALM, 168
- getHop
 - WalkMC, 950
- getID
 - MotionManagerMsg, 610
 - ProcessID, 699
 - SoundManagerMsg, 807
- getInfos
 - Profiler, 705
- getJointMode
 - HeadPointerMC, 452
- getJointValue
 - HeadPointerMC, 452
- getK
 - karmedbanditExp3, 471
- getKeyFrame
 - DynamicMotionSequence, 341
 - MotionSequence, 622, 623
 - MotionSequenceMC, 638
- getLastInput
 - StringInputControl, 846
- GetLegAngles
 - Kinematics.cc, 1201, 1202
 - Kinematics.h, 1209, 1210
- GetLegPosition
 - Kinematics.cc, 1202
 - Kinematics.h, 1210
- getMagnitude
 - EventBase, 365
 - TailWagMC, 851
- getMapping
 - EventRouter::EventManager, 397
- getMaxCapacity
 - ListMemBuf, 502
- getMaxFrames
 - DynamicMotionSequence, 341
 - MotionSequence, 623
 - MotionSequenceMC, 639
- getName
 - Aibo3DControllerBehavior, 161
 - Aibo3DMonitorBehavior, 165
 - AutoGetupBehavior, 173
 - BatteryMonitorBehavior, 220
 - BehaviorBase, 231
 - BehaviorSwitchActivatorControl, 236
 - BehaviorSwitchControl, 241
 - CameraBehavior, 257
 - ChaseBallBehavior, 263
 - ControlBase, 301
 - Controller, 315
 - DriveMeBehavior, 330
 - DumbWM2Behavior, 334
 - EStopControllerBehavior, 355
 - EventBase, 365
 - EvtRptBehavior, 414
 - FollowHeadBehavior, 427
 - FreeMemReportControl, 433
 - HeadLevelBehavior, 438
 - HeadPointControllerBehavior, 443
 - SimpleChaseBallBehavior, 765
 - SoundTestBehavior, 820
 - StareAtBallBehavior, 828
 - StartupBehavior, 832
 - StateNode, 840
 - ToggleHeadLightBehavior, 874
 - ValueEditControl, 882
 - WalkControllerBehavior, 939
- getNearColor
 - Vision, 910
- getNewID
 - Profiler, 706
- getNodes
 - StateNode, 840
- GetNumPlaying
 - SoundManager, 791
- getOutputCmd
 - HeadPointerMC, 452

- MotionManager, [589](#)
- MotionSequence, [623](#)
- PostureEngine, [685](#)
- getOutputIndex
 - MotionSequence, [623](#)
- getOutputs
 - MotionManager, [590](#)
- getPaused
 - WalkMC, [950](#)
- getPeriod
 - TailWagMC, [851](#)
 - WalkMC, [950](#)
- getPID
 - PIDMC, [673](#), [674](#)
- getPixelDirection
 - Vision, [910](#)
- getPlaySpeed
 - MotionSequence, [623](#)
- getPlayTime
 - MotionSequence, [624](#)
- getPort
 - Aibo3DControllerBehavior, [161](#)
 - Aibo3DMonitorBehavior, [165](#)
- getPriority
 - MotionManager, [590](#)
- getProperty
 - VisionEvent, [921](#)
- getReadBuffer
 - Socket, [774](#)
- getRecurse
 - FileBrowserControl, [421](#)
- GetReferences
 - ReferenceCounter, [726](#)
- getRegion
 - SharedObjectBase, [752](#)
- GetRemainTime
 - SoundManager, [791](#)
- getRequests
 - WorldModel2, [990](#)
- getResetSensitivity
 - EmergencyStopMC, [347](#)
- getRoot
 - FileBrowserControl, [421](#)
- GetSamplingRate
 - WAV, [972](#)
- getSetting
 - LedEngine, [484](#)
- GETSIG
 - WorldState.cc, [1438](#)
- getSlotName
 - ControlBase, [301](#)
- getSlots
 - ControlBase, [301](#)
- GetSoundUnitSize
 - WAV, [972](#)
- getSourceID
 - EventBase, [365](#)
- getStopped
 - EmergencyStopMC, [348](#)
- getSway
 - WalkMC, [950](#)
- getTarget
 - ValueEditControl, [883](#)
 - ValueSetControl, [889](#)
- getTargetVelocity
 - WalkMC, [951](#)
- getText
 - TextMsgEvent, [857](#)
- getTilt
 - TailWagMC, [851](#)
- getTimeStamp
 - EventBase, [365](#)
- getToken
 - TextMsgEvent, [857](#)
- getTolerance
 - PostureMC, [693](#)
- getTransitions
 - StateNode, [840](#)
- getTravelTime
 - WalkMC, [951](#)
- GetTrigAngle
 - Kinematics.cc, [1202](#)
- getTypeID
 - EventBase, [366](#)
- getUsedFrames
 - DynamicMotionSequence, [341](#)
 - MotionSequence, [624](#)
 - MotionSequenceMC, [639](#)
- getValue
 - LedEngine, [485](#)
- getWidth
 - Vision, [910](#)

- getWriteBuffer
 - Socket, [774](#)
- GM
 - agmMain.cc, [1048](#)
- GM_CELL_COUNT
 - Maps/Configuration.h, [1101](#)
- gm_index2xy
 - almUtility.cc, [1060](#)
 - almUtility.h, [1063](#)
- gm_port
 - Config::worldmodel2_config, [291](#)
- GO_TO
 - MotionRequest, [612](#)
- GotEventTranslatorQueue
 - MMCombo, [563](#)
 - SoundPlay, [814](#)
- GotImage
 - MMCombo, [563](#)
- GotMotionManager
 - MMCombo, [563](#)
- GotMotionMsg
 - MMCombo, [563](#)
- GotPowerEvent
 - MMCombo, [564](#)
- GotSensorFrame
 - MMCombo, [564](#)
- GotSoundManager
 - MMCombo, [564](#)
- GotSoundMsg
 - SoundPlay, [814](#)
- gotweight
 - _afsParticle, [144](#)
- GotWorldState
 - MMCombo, [564](#)
- GPAdelay
 - WorldModel2, [994](#)
- gr
 - karmedbanditExp3_1, [476](#)
- GravityRelative
 - HeadPointerMC, [450](#)
- gStartup
 - StartupBehavior.cc, [1344](#)
- GT
 - CompareTrans, [267](#)
- GTE
 - CompareTrans, [267](#)
- gui_comm
 - ControlBase, [305](#)
 - Controller, [319](#)
- gui_comm_callback
 - Controller, [315](#)
- gui_port
 - Config::controller_config, [277](#)
- GVector, [116](#)
 - cross, [117](#)
 - distance, [117](#)
 - distance_to_line, [118](#)
 - dot, [118](#)
 - intersect_ray_plane, [118](#)
 - offset_to_line, [118](#)
 - point_on_segment, [119](#)
 - sdistance, [119](#)
 - VECTOR2D_EQUAL_-
BINARY_OPERATOR,
[119](#)
 - VECTOR2D_LOGIC_-
OPERATOR, [120](#)
 - VECTOR3D_EQUAL_-
BINARY_OPERATOR,
[120](#)
 - VECTOR3D_LOGIC_-
OPERATOR, [120](#)
- gvector.h, [1177](#)
 - V2COMP, [1179](#)
 - V3COMP, [1179](#)
 - VECTOR2D_BINARY_-
OPERATOR, [1179](#)
 - VECTOR2D_EQUAL_-
BINARY_OPERATOR,
[1179](#)
 - VECTOR2D_EQUAL_-
SCALAR_OPERATOR,
[1179](#)
 - VECTOR2D_LOGIC_-
OPERATOR, [1180](#)
 - VECTOR2D_SCALAR_-
OPERATOR, [1180](#)
 - VECTOR3D_BINARY_-
OPERATOR, [1180](#)
 - VECTOR3D_EQUAL_-
BINARY_OPERATOR,
[1180](#)

- VECTOR3D.EQUAL_-
 - SCALAR_OPERATOR, 1181
- VECTOR3D.LOGIC_-
 - OPERATOR, 1181
- VECTOR3D.SCALAR_-
 - OPERATOR, 1181
- GVector::vector2d, 891
 - angle, 892
 - cross, 892
 - dot, 892
 - length, 892
 - norm, 893
 - normalize, 893
 - operator *, 893
 - operator *=, 893
 - operator !=, 893
 - operator +, 893
 - operator +=, 893
 - operator -, 893, 894
 - operator =, 894
 - operator /, 894
 - operator /=, 894
 - operator <, 894
 - operator <=, 894
 - operator =, 894
 - operator ==, 894
 - operator >, 895
 - operator >=, 895
 - rotate, 895
 - set, 895
 - sqlength, 895
 - vector2d, 892
 - x, 895
 - y, 895
- GVector::vector3d, 897
 - cross, 898
 - dot, 898
 - length, 899
 - norm, 899
 - normalize, 899
 - operator *, 899, 900
 - operator *=, 900
 - operator !=, 900
 - operator +, 900
 - operator +=, 900
 - operator -, 900
 - operator =, 901
 - operator /=, 901
 - operator /, 901
 - operator /=, 901
 - operator <, 901
 - operator <=, 901
 - operator =, 901
 - operator ==, 901
 - operator >, 901
 - operator >=, 901
 - rotate_x, 901
 - rotate_y, 902
 - rotate_z, 902
 - set, 902
 - sqlength, 902
 - vector3d, 898
 - x, 902
 - y, 902
 - z, 903
- h_body_to_shoulder
 - Kinematics.cc, 1202
- h_knee_kmin
 - Kinematics.cc, 1203
 - Kinematics.h, 1211
- h_leg_knee_to_ball
 - Kinematics.cc, 1202
- h_leg_shoulder_to_knee
 - Kinematics.cc, 1202
- h_lower
 - Kinematics.cc, 1203
- h_upper
 - Kinematics.cc, 1203
- HAND
 - VisionInterface, 133
- hand
 - VisionInterface::ObjectInfo, 658
- HandSID
 - VisionEventNS, 131
- hasData
 - Wireless, 979
- hasListeners
 - EventRouter, 387, 388
- hasMapping
 - EventRouter::EventMapper, 397, 398

- head
 - HeadLevelBehavior, [439](#)
- head_angles
 - Vision, [914](#)
- head_id
 - HeadLevelBehavior, [439](#)
 - HeadPointControllerBehavior, [445](#)
 - WorldModel2Behavior::Gawk-Node, [1002](#)
 - WorldModel2Behavior::Wait-Node, [1005](#)
 - WorldModel2Behavior::Walk-Node, [1010](#)
- head_lock
 - FollowHeadBehavior, [428](#)
 - HeadLevelBehavior, [439](#)
- head_pan_max
 - Kinematics.cc, [1204](#)
 - Kinematics.h, [1211](#)
- head_pan_min
 - Kinematics.cc, [1204](#)
 - Kinematics.h, [1211](#)
- head_range2xyz
 - almUtility.cc, [1061](#)
 - almUtility.h, [1064](#)
- head_release
 - FollowHeadBehavior, [428](#)
 - HeadLevelBehavior, [439](#)
- head_roll_max
 - Kinematics.cc, [1204](#)
 - Kinematics.h, [1211](#)
- head_roll_min
 - Kinematics.cc, [1204](#)
 - Kinematics.h, [1212](#)
- head_tilt_max
 - Kinematics.cc, [1204](#)
 - Kinematics.h, [1212](#)
- head_tilt_min
 - Kinematics.cc, [1204](#)
 - Kinematics.h, [1212](#)
- HeadBkButOffset
 - ERS210Info, [39](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [96](#)
- HeadBkButSID
 - ButtonSourceID, [30](#)
- headControl_port
 - Config::main_config, [280](#)
- HeadFrButOffset
 - ERS210Info, [39](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [96](#)
- HeadFrButSID
 - ButtonSourceID, [30](#)
- headJoints
 - HeadPointerMC, [455](#)
- HeadLEDMask
 - ERS210Info, [48](#)
 - ERS220Info, [77](#)
 - ERS2xxInfo, [104](#)
- HeadLevelBehavior, [436](#)
 - HeadLevelBehavior, [437](#)
- HeadLevelBehavior
 - ~HeadLevelBehavior, [437](#)
 - DoStart, [438](#)
 - DoStop, [438](#)
 - getClassDescription, [438](#)
 - getName, [438](#)
 - head, [439](#)
 - head_id, [439](#)
 - head_lock, [439](#)
 - head_release, [439](#)
 - HeadLevelBehavior, [437](#)
 - processEvent, [438](#)
- HeadLevelBehavior.h, [1183](#)
- headModes
 - HeadPointerMC, [455](#)
- HeadOffset
 - ERS210Info, [49](#)
 - ERS220Info, [78](#)
 - ERS2xxInfo, [104](#)
- HeadPointControllerBehavior, [440](#)
 - HeadPointControllerBehavior, [442](#)
- HeadPointControllerBehavior
 - ~HeadPointControllerBehavior, [442](#)
 - CMD_pan, [445](#)
 - CMD_roll, [445](#)
 - CMD_tilt, [445](#)
 - cmdsock, [445](#)

- DoStart, 443
- DoStop, 443
- getClassDescription, 443
- getName, 443
- head_id, 445
- HeadPointControllerBehavior, 442
- mechacmd_callback, 444
- operator=, 444
- p, 445
- processEvent, 444
- r, 445
- runCommand, 444
- t, 446
- theLastOne, 446
- theOne, 446
- HeadPointControllerBehavior.cc, 1185
- HeadPointControllerBehavior.h, 1186
- headpointer_id
 - CameraBehavior, 257
 - ChaseBallBehavior, 264
 - StareAtBallBehavior, 829
 - WalkToTargetMachine, 969
- HeadPointerMC, 447
 - BodyRelative, 450
 - GravityRelative, 450
 - HeadPointerMC, 450
- HeadPointerMC
 - ~HeadPointerMC, 450
 - active, 455
 - convert, 451
 - convFromBodyRelative, 451
 - convToBodyRelative, 451
 - CoordFrame.t, 450
 - dirty, 455
 - getActive, 451
 - getJointMode, 452
 - getJointValue, 452
 - getOutputCmd, 452
 - headJoints, 455
 - headModes, 455
 - HeadPointerMC, 450
 - headValues, 455
 - isAlive, 452
 - isDirty, 452
 - setActive, 453
 - setJointMode, 453
 - setJoints, 453
 - setJointValue, 453
 - setJointValueAndMode, 453
 - setJointValueFromMode, 454
 - setMode, 454
 - setWeight, 454
 - unused, 456
 - updateOutputs, 454
- HeadPointerMC.cc, 1188
- HeadPointerMC.h, 1189
- headValues
 - HeadPointerMC, 455
- height
 - _hm_cell, 155
 - Vision, 914
- HelpControl, 457
 - HelpControl, 458
- HelpControl
 - activate, 458
 - HelpControl, 458
 - operator=, 458
 - report, 459
 - root, 459
 - term.width, 459
- HelpControl.cc, 1191
- HelpControl.h, 1192
- HermiteSplineSegment, 460
 - HermiteSplineSegment, 460
- HermiteSplineSegment
 - a, 461
 - b, 461
 - c, 461
 - create, 461
 - d, 461
 - eval, 461
 - eval_deriv, 461
 - HermiteSplineSegment, 460
- hexdigit
 - debuget.h, 1112
- hexout
 - debuget.h, 1112
- high_power_p
 - BatteryMonitorBehavior, 222
- hilightFirst

- ControlBase, 301
- hilights
 - ControlBase, 305
- hilightsAvg
 - ControlBase, 301
- HistCurve
 - Profiler, 707
- HistSize
 - Profiler, 707
- HistTime
 - Profiler, 708
- HM
 - almMain.cc, 1055
- hm_cell
 - almStructures.h, 1059
- HM_CELL_COUNT
 - Maps/Configuration.h, 1101
- hm_index2xy
 - almUtility.cc, 1061
 - almUtility.h, 1064
- hm_port
 - Config::worldmodel2_config, 291
- hmPickCluster, 462
- hmPickCluster
 - operator(), 462
- hmPickColor, 463
- hmPickColor
 - operator(), 463
- hmPickConfidence, 464
- hmPickConfidence
 - operator(), 464
- hmPicker, 465
- hmPicker
 - operator(), 465
- hmPickHeight, 466
- hmPickHeight
 - operator(), 466
- hmPickTrav, 467
- hmPickTrav
 - operator(), 467
- HMs
 - almMain.cc, 1055
- hop
 - WalkMC::WalkParam, 963
- HorzFOV
 - VisionInterface, 133
- hostToNetwork
 - Serializer, 746
- HSPLINE
 - Spline.h, 1336
- HSPLINE_TEM
 - Spline.h, 1336
- ID
 - ProcessID, 700
- id
 - Config::wireless_config, 289
 - MutexLock::door_t, 650
 - SoundManagerMsg, 809
- identifyMarker
 - Vision, 910
- if
 - motionReshapeKludge.h, 1254
- IgnoreGreenItems
 - WM2Kludge, 136
- IgnoreZLessThanZero
 - WM2Kludge, 136
- image
 - Vision.h, 1386
- img
 - Vision, 915
- in_sync
 - basic_netbuf, 203
- index
 - BanditMachine::PressNode, 185
- index_t
 - ListMemBuf, 499
- iNetStream
 - ionetstream.h, 1195
- infos
 - LedEngine, 488
 - ProfilerOfSize, 722
- infosOffset
 - Profiler, 708
- Init
 - basic_netbuf, 204
- init
 - AGM, 157
 - ALM, 168
 - BehaviorActivatorControl, 227
 - Controller, 315
 - FileBrowserControl, 421

- FreeMemReportControl, [433](#)
- MutexLock, [645](#)
- Socket, [775](#)
- SplinePath, [824](#)
- WalkMC, [951](#)
- InitAccess
 - MotionManager, [590](#)
 - SoundManager, [791](#)
- initBuckets
 - Profiler, [706](#)
- initialize
 - Vision, [911](#)
- initializeEventSpecs
 - Vision, [911](#)
- InitRegion
 - MMCombo, [564](#)
 - SoundPlay, [815](#)
- initRegion
 - SoundManager, [791](#)
- int_type
 - basic_iNetStream, [191](#)
 - basic_ioNetStream, [196](#)
 - basic_netbuf, [201](#)
 - basic_oNetStream, [209](#)
 - char_traits, [259](#)
- interExpAvg
 - Profiler::SectionInfo, [712](#)
- interHist
 - Profiler::SectionInfo, [712](#)
- interpolate
 - MotionCommand, [575](#), [576](#)
- intersect_ray_plane
 - GVector, [118](#)
- invadeDMData
 - WorldModel2, [991](#)
- invadeHMData
 - WorldModel2, [991](#)
- invalid_MC_ID
 - MotionManager, [597](#)
- invalid_move
 - MotionSequence, [630](#)
- invalid_Play_ID
 - SoundManager, [797](#)
- invalid_Snd_ID
 - SoundManager, [797](#)
- INVALID_SOCKET
 - ionetstream.h, [1195](#)
- invert
 - LedEngine, [485](#)
- ioNetStream
 - ionetstream.h, [1195](#)
- ionetstream.h, [1194](#)
 - iNetStream, [1195](#)
 - INVALID_SOCKET, [1195](#)
 - ioNetStream, [1195](#)
 - netbuf, [1195](#)
 - oNetStream, [1195](#)
- iostream, [468](#)
- ipstackRef
 - Wireless, [983](#)
- IRDistOffset
 - ERS210Info, [42](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [99](#)
- IROORDIST
 - almMain.cc, [1055](#)
- IROORDist
 - WorldState, [1017](#)
- is_echoing
 - basic_netbuf, [207](#)
- is_open
 - basic_iNetStream, [192](#)
 - basic_ioNetStream, [198](#)
 - basic_netbuf, [204](#)
 - basic_oNetStream, [211](#)
- isActive
 - BehaviorBase, [232](#)
 - MotionCommand, [577](#)
- isAdjacent
 - Vision, [911](#)
- isAlive
 - HeadPointerMC, [452](#)
 - LedMC, [493](#)
 - MotionCommand, [577](#)
 - MotionSequence, [624](#)
 - PIDMC, [674](#)
 - PostureMC, [693](#)
 - RemoteControllerMC, [729](#)
 - TailWagMC, [852](#)
 - WalkMC, [951](#)
- isConnected
 - Wireless, [979](#)

- isCustomName
 - EventBase, [366](#)
- isCycling
 - LedEngine::LEDInfo, [490](#)
- isDirty
 - HeadPointerMC, [452](#)
 - LedEngine, [485](#)
 - LedMC, [493](#)
 - MotionCommand, [577](#)
 - MotionSequence, [624](#)
 - PIDMC, [674](#)
 - PostureMC, [693](#), [694](#)
 - RemoteControllerMC, [730](#)
 - TailWagMC, [852](#)
 - WalkMC, [951](#)
- IsFastOutput
 - ERS210Info, [49](#)
 - ERS220Info, [78](#)
 - ERS2xxInfo, [104](#)
- isIn
 - Vision, [911](#)
- isPaused
 - WalkMC, [955](#)
- isPID
 - PIDMC, [674](#)
- isPlaying
 - MotionSequence, [625](#)
- isReady
 - Wireless, [980](#)
- IsRealERS210
 - ERS210Info, [49](#)
 - ERS2xxInfo, [105](#)
- IsRealERS220
 - ERS220Info, [78](#)
 - ERS2xxInfo, [105](#)
- isRunning
 - BehaviorSwitchControl, [241](#)
- isSaveDegrees
 - MotionSequence, [625](#)
- isSaveRadians
 - MotionSequence, [625](#)
- issetup
 - StateNode, [842](#)
- isStopped
 - MMCombo, [568](#)
- istream, [469](#)
- isValid
 - BehaviorSwitchControl, [241](#)
- isWarning
 - FreeMemReportControl, [434](#)
- joint
 - MotionManager::PIDUpdate, [607](#)
- JointsPerLeg
 - ERS210Info, [49](#)
 - ERS220Info, [79](#)
 - ERS2xxInfo, [106](#)
- karmedbandit.h, [1196](#)
- karmedbanditExp3, [470](#)
 - karmedbanditExp3, [471](#)
- karmedbanditExp3
 - decide, [471](#)
 - g, [472](#)
 - getGamma, [471](#)
 - getK, [471](#)
 - karmedbanditExp3, [471](#)
 - last, [472](#)
 - lastp, [472](#)
 - reset, [471](#)
 - reward, [472](#)
 - setGamma, [472](#)
 - w, [472](#)
- karmedbanditExp3_1, [474](#)
 - karmedbanditExp3_1, [475](#)
- karmedbanditExp3_1
 - decide, [475](#)
 - exp3, [476](#)
 - G, [476](#)
 - gr, [476](#)
 - karmedbanditExp3_1, [475](#)
 - last, [476](#)
 - r, [476](#)
 - restart, [475](#)
 - reward, [475](#)
- kBackgroundPriority
 - MotionManager, [597](#)
- kEmergencyPriority
 - MotionManager, [597](#)
- kHighPriority
 - MotionManager, [597](#)

- kIgnoredPriority
 - MotionManager, 598
- KinClearErrors
 - Kinematics.cc, 1202
 - Kinematics.h, 1210
- Kinematics.cc, 1198
 - errors, 1203
 - f_body_to_shoulder, 1201
 - f_knee_kmin, 1203
 - f_leg_knee_to_ball, 1201
 - f_leg_shoulder_to_knee, 1201
 - f_lower, 1203
 - f_upper, 1203
 - GetBodyLocation, 1201
 - GetHeadAngles, 1201
 - GetHeadPosition, 1201
 - GetLegAngles, 1201, 1202
 - GetLegPosition, 1202
 - GetTrigAngle, 1202
 - h_body_to_shoulder, 1202
 - h_knee_kmin, 1203
 - h_leg_knee_to_ball, 1202
 - h_leg_shoulder_to_knee, 1202
 - h_lower, 1203
 - h_upper, 1203
 - head_pan_max, 1204
 - head_pan_min, 1204
 - head_roll_max, 1204
 - head_roll_min, 1204
 - head_tilt_max, 1204
 - head_tilt_min, 1204
 - KinClearErrors, 1202
 - KinGetErrors, 1202
 - knee_kmax, 1204
 - knee_max, 1204
 - knee_min, 1204
 - rotator_kmax, 1204
 - rotator_kmin, 1205
 - rotator_max, 1205
 - rotator_min, 1205
 - RunForwardModel, 1203
 - shoulder_kmax, 1205
 - shoulder_kmin, 1205
 - shoulder_max, 1205
 - shoulder_min, 1205
 - tail_max, 1205
 - tail_min, 1205
- Kinematics.h, 1206
 - body_to_neck, 1209
 - DEG, 1209
 - f_knee_kmin, 1211
 - GetBodyLocation, 1209
 - GetHeadAngles, 1209
 - GetHeadPosition, 1209
 - GetLegAngles, 1209, 1210
 - GetLegPosition, 1210
 - h_knee_kmin, 1211
 - head_pan_max, 1211
 - head_pan_min, 1211
 - head_roll_max, 1211
 - head_roll_min, 1212
 - head_tilt_max, 1212
 - head_tilt_min, 1212
 - KinClearErrors, 1210
 - KinGetErrors, 1211
 - knee_kmax, 1212
 - knee_max, 1212
 - knee_min, 1212
 - neck_to_camera, 1211
 - RAD, 1209
 - rotator_kmax, 1212
 - rotator_kmin, 1212
 - rotator_max, 1212
 - rotator_min, 1212
 - RunForwardModel, 1211
 - shoulder_kmax, 1213
 - shoulder_kmin, 1213
 - shoulder_max, 1213
 - shoulder_min, 1213
 - tail_max, 1213
 - tail_min, 1213
- KinGetErrors
 - Kinematics.cc, 1202
 - Kinematics.h, 1211
- kLowPriority
 - MotionManager, 598
- kludges
 - WorldModel2, 994
- knee_kmax
 - Kinematics.cc, 1204
 - Kinematics.h, 1212
- knee_max

- Kinematics.cc, [1204](#)
 - Kinematics.h, [1212](#)
- knee_min
 - Kinematics.cc, [1204](#)
 - Kinematics.h, [1212](#)
- KneeOffset
 - ERS210Info, [41](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [98](#)
- kStdPriority
 - MotionManager, [598](#)
- l
 - BanditMachine::DecideNode, [182](#)
 - LockScope, [540](#)
- LAccelOffset
 - ERS210Info, [42](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [99](#)
- LANDMARK
 - _FastSLAM_update, [152](#)
- landmarks
 - _afsParticle, [144](#)
 - Test.c, [1357](#)
- last
 - karmedbanditExp3, [472](#)
 - karmedbanditExp3_1, [476](#)
- last_da
 - DriveMeBehavior, [330](#)
- last_dx
 - DriveMeBehavior, [330](#)
- last_dy
 - DriveMeBehavior, [330](#)
- last_time
 - Controller, [320](#)
 - DriveMeBehavior, [331](#)
- lastAccessor
 - MotionManager::Command-Entry, [601](#)
- lastBufClear
 - EventRouter, [393](#)
- lastInput
 - StringInputControl, [847](#)
- lastp
 - karmedbanditExp3, [472](#)
- lastSensorUpdateTime
 - WorldState, [1017](#)
- lastTime
 - Profiler::SectionInfo, [712](#)
- lasttime
 - MotionSequence, [630](#)
- LazyFastSLAM
 - WM2Kludge, [136](#)
- LBk
 - SoundTestBehavior, [822](#)
- LBkLegOffset
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- LBkLegOrder
 - ERS210Info, [41](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [98](#)
- LBkPawOffset
 - ERS210Info, [39](#)
 - ERS220Info, [66](#)
 - ERS2xxInfo, [96](#)
- LBkPawSID
 - ButtonSourceID, [30](#)
- led_id
 - BatteryMonitorBehavior, [222](#)
 - CameraBehavior, [258](#)
- ledActivation
 - MMCombo, [568](#)
- LEDBitMask_t
 - ERS210Info, [39](#)
 - ERS220Info, [66](#)
 - ERS2xxInfo, [95](#)
- LedEngine, [477](#)
 - LedEngine, [481](#)
 - major, [481](#)
 - minor, [481](#)
 - none, [481](#)
 - onedigit, [480](#)
 - twodigit, [480](#)
- LedEngine
 - ~LedEngine, [481](#)
 - calcCycle, [481](#)
 - calcFlash, [481](#)
 - calcInvert, [482](#)
 - calcValue, [482](#)

- ccycle, [482](#)
- cflash, [482](#)
- clear, [482](#)
- cset, [483](#)
- cycle, [483](#)
- dirty, [487](#)
- dirtyTime, [487](#)
- displayNumber, [483](#)
- displayPercent, [484](#)
- ERS210numMasks, [487](#)
- ERS220numMasks, [488](#)
- flash, [484](#)
- getSetting, [484](#)
- getValue, [485](#)
- infos, [488](#)
- invert, [485](#)
- isDirty, [485](#)
- LedEngine, [481](#)
- numCycling, [488](#)
- numStyle.t, [480](#)
- percentStyle.t, [481](#)
- set, [485](#)
- setColumn, [485](#)
- setOneOfTwo, [486](#)
- updateLEDFrames, [486](#)
- updateLEDs, [486](#)
- ledengine
 - EmergencyStopMC, [350](#)
- LedEngine.cc, [1214](#)
- LedEngine.h, [1215](#)
- LedEngine::LEDInfo, [489](#)
- LedEngine::LEDInfo
 - amp, [489](#)
 - flashtime, [490](#)
 - flashvalue, [490](#)
 - isCycling, [490](#)
 - offset, [490](#)
 - period, [490](#)
 - starttime, [490](#)
 - value, [490](#)
- LedMC, [492](#)
 - LedMC, [493](#)
- LedMC
 - ~LedMC, [493](#)
 - cmds, [494](#)
 - isAlive, [493](#)
 - isDirty, [493](#)
 - LedMC, [493](#)
 - setWeights, [493](#)
 - updateOutputs, [494](#)
- LedMC.h, [1217](#)
- LEDOffset
 - ERS210Info, [49](#)
 - ERS220Info, [79](#)
 - ERS2xxInfo, [106](#)
- LEDOffset.t
 - ERS210Info, [39](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [96](#)
- leds_id
 - BanditMachine::WaitNode, [188](#)
- left
 - VisionInterface::VObject, [933](#)
- leg
 - WalkMC::WalkParam, [963](#)
- LegOffset
 - ERS210Info, [49](#)
 - ERS220Info, [79](#)
 - ERS2xxInfo, [106](#)
- LegOffset.t
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- LegOrder.t
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- LegParam
 - WalkMC::LegParam, [958](#)
- legpos
 - WalkMC, [955](#)
- legw
 - WalkMC, [955](#)
- LegWalkState
 - WalkMC::LegWalkState, [960](#)
- len
 - SharedQueue, [759](#)
 - SoundManager::SoundData, [803](#)
- length
 - GVector::vector2d, [892](#)
 - GVector::vector3d, [899](#)
- LFr

- SoundTestBehavior, 822
- LFrLegOffset
 - ERS210Info, 40
 - ERS220Info, 68
 - ERS2xxInfo, 97
- LFrLegOrder
 - ERS210Info, 41
 - ERS220Info, 68
 - ERS2xxInfo, 98
- LFrPawOffset
 - ERS210Info, 39
 - ERS220Info, 66
 - ERS2xxInfo, 96
- LFrPawSID
 - ButtonSourceID, 30
- liedown
 - BanditMachine, 178
- lift.time
 - WalkMC::LegParam, 959
- lift.vel
 - WalkMC::LegParam, 959
- light_id
 - ToggleHeadLightBehavior, 874
- limitRange
 - mathutils, 121
- lin_distort_coeff
 - Vision, 915
- list.length
 - Util.h, 1373
- list_t
 - DynamicMotionSequence, 340
 - MotionSequenceMC, 637
- listen
 - Wireless, 980
- ListenCont
 - MMCombo, 565
 - Wireless, 981
- listeners
 - EventRouter, 393
 - VisionEventSpec, 924
- ListMemBuf, 495
 - ListMemBuf, 499
- ListMemBuf
 - activeBack, 507
 - activeBegin, 507
 - back, 500
 - begin, 500
 - clear, 500
 - countb, 500
 - countf, 500
 - cursize, 507
 - empty, 501
 - end, 501
 - entries, 507
 - erase, 501
 - freeBack, 507
 - freeBegin, 507
 - front, 501
 - getMaxCapacity, 502
 - index_t, 499
 - ListMemBuf, 499
 - MAX_ENTRIES, 507
 - new_after, 502
 - new_back, 502
 - new_before, 502
 - new_front, 503
 - next, 503
 - operator[], 503
 - pop_back, 503, 504
 - pop_free, 504
 - pop_front, 504
 - prev, 505
 - push_after, 505
 - push_back, 505
 - push_before, 505
 - push_free, 505
 - push_front, 506
 - size, 506
 - swap, 506
 - T, 499
- ListMemBuf.h, 1219
- ListMemBuf::entry_t, 509
- ListMemBuf::entry_t
 - data, 510
 - entry_t, 509
 - next, 510
 - prev, 510
- load
 - WalkMC, 952
- LoadBuffer
 - EventBase, 366
 - LoadSave, 532

- LocomotionEvent, [543](#)
- MotionSequence, [625](#)
- PostureEngine, [685](#)
- PostureMC, [694](#)
- SoundManager, [792](#)
- TextMsgEvent, [857](#)
- VisionEvent, [921](#)
- LoadFile
 - LoadSave, [533](#)
 - SoundManager, [792](#)
- loadGUI
 - Controller, [315](#), [316](#)
- LoadPostureControl, [511](#)
 - LoadPostureControl, [512](#)
- LoadPostureControl
 - ~LoadPostureControl, [512](#)
 - deactivate, [512](#)
 - estopid, [513](#)
 - file, [513](#)
 - LoadPostureControl, [512](#)
 - processEvent, [512](#)
 - selectedFile, [513](#)
- LoadPostureControl.h, [1220](#)
- LoadSave, [514](#)
 - LoadSave, [520](#)
- LoadSave
 - ~LoadSave, [520](#)
 - byteswap, [520](#)
 - checkCreator, [520](#), [521](#)
 - creatorSize, [521](#)
 - decode, [522](#)–[527](#)
 - encode, [527](#)–[532](#)
 - getBinSize, [532](#)
 - LoadBuffer, [532](#)
 - LoadFile, [533](#)
 - LoadSave, [520](#)
 - SaveBuffer, [533](#)
 - saveCreator, [534](#)
 - SaveFile, [535](#)
 - stringpad, [535](#)
- LoadSave.cc, [1222](#)
- LoadSave.h, [1223](#)
- loadSaveMode
 - MotionSequence, [630](#)
- LoadWalkControl, [537](#)
 - LoadWalkControl, [538](#)
- LoadWalkControl
 - ~LoadWalkControl, [538](#)
 - LoadWalkControl, [538](#)
 - selectedFile, [538](#)
 - walk_id, [538](#)
- LoadWalkControl.h, [1225](#)
- loc
 - BodyPosition, [253](#)
 - VisionInterface::VObject, [933](#)
- lock
 - MotionManager, [591](#)
 - MotionManager::Command-Entry, [601](#)
 - MutexLock, [645](#)
 - SharedQueue, [759](#)
 - SoundManager, [797](#)
- lockcount
 - MutexLock, [647](#)
- locked
 - WorldModel2, [994](#)
- LockScope, [539](#)
 - LockScope, [539](#)
- LockScope
 - ~LockScope, [539](#)
 - l, [540](#)
 - LockScope, [539](#)
- LockScope.h, [1227](#)
- locomotionEGID
 - EventBase, [362](#)
- LocomotionEvent, [541](#)
 - LocomotionEvent, [542](#)
- LocomotionEvent
 - a, [544](#)
 - getBinSize, [543](#)
 - LoadBuffer, [543](#)
 - LocomotionEvent, [542](#)
 - SaveBuffer, [543](#)
 - setXYA, [544](#)
 - x, [544](#)
 - y, [544](#)
- LocomotionEvent.h, [1229](#)
- LocomotionEvent_ID
 - EventTranslator, [407](#)
- log2t
 - mathutils, [121](#)
- logfile

- EventLogger, [376](#)
- logfilePath
 - EventLogger, [376](#)
- longerThan
 - EventBase, [366](#)
- LOOK_AT
 - MotionRequest, [612](#)
- LOOK_DOWN_AT
 - MotionRequest, [612](#)
- lookup
 - MutexLock, [646](#)
 - SoundManager, [792](#)
- lookupPath
 - SoundManager, [793](#)
- lost
 - WalkToTargetMachine, [969](#)
- low_mem
 - FreeMemReportControl, [434](#)
- LowPowerWarnSID
 - PowerSourceID, [124](#)
- LT
 - CompareTrans, [267](#)
- LTE
 - CompareTrans, [267](#)
- magnitude
 - EventBase, [370](#)
 - TailWagMC, [854](#)
- main
 - Config, [273](#)
 - Test.c, [1357](#)
- main_config
 - Config::main_config, [279](#)
- MainProcess
 - ProcessID, [699](#)
- mainProfile
 - WorldState, [1017](#)
- major
 - LedEngine, [481](#)
- makeLower
 - Controller, [316](#)
- makePath
 - Config::motion_config, [283](#)
 - Config::sound_config, [285](#)
 - FileBrowserControl, [421](#), [422](#)
 - SoundManager, [793](#)
- makeSafe
 - MotionSequence, [625](#)
- Maps/Configuration.h
 - AGM_BOTTOM, [1098](#)
 - AGM_H_SIZE, [1098](#)
 - AGM_LEFT, [1098](#)
 - AGM_NUMCLUSTERS, [1098](#)
 - AGM_RIGHT, [1098](#)
 - AGM_TOP, [1098](#)
 - AGM_V_SIZE, [1098](#)
 - AIBO_CAM_H_FOV, [1098](#)
 - AIBO_CAM_V_FOV, [1099](#)
 - AIBO_HEAD_LENGTH, [1099](#)
 - AIBO_IR_CAL_MULTIMPLIER, [1099](#)
 - AIBO_IR_CAL_OFFSET, [1099](#)
 - AIBO_MAX_BUMP, [1099](#)
 - AIBO_MIN_CLEARANCE, [1099](#)
 - AIBO_NECK_HEIGHT, [1099](#)
 - AIBO_TILT_PIVOT_HEIGHT, [1099](#)
 - AIBO_TURN_PIVOT_DIST, [1099](#)
 - ALM_DM_BOTTOM, [1099](#)
 - ALM_DM_H_SIZE, [1100](#)
 - ALM_DM_LEFT, [1100](#)
 - ALM_DM_NUMCLUSTERS, [1100](#)
 - ALM_DM_RIGHT, [1100](#)
 - ALM_DM_TAX, [1100](#)
 - ALM_DM_TOP, [1100](#)
 - ALM_DM_V_SIZE, [1100](#)
 - ALM_GM_TAX, [1100](#)
 - ALM_GPA_CONFIDENCE, [1100](#)
 - ALM_HM_NUMCLUSTERS, [1100](#)
 - ALM_HM_RADIUS, [1101](#)
 - ALM_HM_SIZE, [1101](#)
 - ALM_HM_TAX, [1101](#)
 - ALM_IR_SPLAT_STDDEV, [1101](#)
 - AM_KMEANS_ITERATIONS, [1101](#)
 - DM_CELL_COUNT, [1101](#)

- GM.CELL_COUNT, [1101](#)
- HM.CELL_COUNT, [1101](#)
- R, [1101](#)
- Marker, [546](#)
 - cen_x, [546](#)
 - cen_y, [546](#)
 - marker, [546](#)
 - regs, [546](#)
- marker
 - Marker, [546](#)
 - VisionInterface::ObjectInfo, [658](#)
- MARKER_GOG
 - VisionInterface, [133](#)
- MARKER_GOP
 - VisionInterface, [133](#)
- MARKER_GPG
 - VisionInterface, [133](#)
- MARKER_GPO
 - VisionInterface, [133](#)
- MARKER_OGO
 - VisionInterface, [134](#)
- MARKER_OGP
 - VisionInterface, [134](#)
- MARKER_OPG
 - VisionInterface, [134](#)
- MARKER_OPO
 - VisionInterface, [134](#)
- MARKER_PGO
 - VisionInterface, [134](#)
- MARKER_PGP
 - VisionInterface, [134](#)
- MARKER_POG
 - VisionInterface, [134](#)
- MARKER_POP
 - VisionInterface, [134](#)
- markers
 - Vision, [915](#)
- MarkersSID
 - VisionEventNS, [131](#)
- match
 - FileBrowserControl, [422](#)
- mathutils, [121](#)
 - abs_t, [121](#)
 - distance, [121](#)
 - limitRange, [121](#)
 - log2t, [121](#)
 - squared, [121](#)
 - squareDistance, [122](#)
- mathutils.h, [1231](#)
- max3
 - Util.h, [1373](#)
- max_accel_xya
 - WalkMC, [955](#)
- MAX_ACCESS
 - MotionManager, [598](#)
- max_chan
 - SoundManager, [797](#)
- MAX_COLORS
 - Visiondefines.h, [1389](#)
- MAX_DA
 - WalkMC, [955](#)
- MAX_DX
 - WalkMC, [955](#)
- MAX_DY
 - WalkMC, [956](#)
- MAX_ENTRIES
 - ListMemBuf, [507](#)
 - SharedQueue, [759](#)
- max_height
 - Vision, [915](#)
- MAX_MOTIONS
 - MotionManager, [598](#)
- MAX_NAMELEN
 - SoundManager, [798](#)
- MAX_PLAY
 - SoundManager, [798](#)
- max_regions
 - Vision, [915](#)
- max_runs
 - Vision, [915](#)
- MAX_SIZE
 - SharedQueue, [759](#)
- MAX_SND
 - SoundManager, [798](#)
- max_t
 - BatteryMonitorBehavior, [222](#)
- MAX_TMAPS
 - Vision.h, [1385](#)
- max_width
 - Vision, [915](#)
- maxdiff
 - PostureEngine, [686](#)

- MaxOutputSpeed
 - ERS210Info, [50](#)
 - ERS220Info, [79](#)
 - ERS2xxInfo, [106](#)
- MaxRange
 - ERS210Info, [41](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [98](#)
- MaxSectionNameLen
 - Profiler, [708](#)
- maxSections
 - Profiler, [708](#)
- mc
 - MMAccessor, [553](#)
- MC_ID
 - MotionManager, [586](#)
 - MotionManagerMsg, [609](#)
- mc_id
 - MMAccessor, [555](#)
 - MotionManagerMsg, [611](#)
- mcid
 - MCValueEditControl, [548](#)
 - MotionManager::OutputState, [605](#)
- mcopy
 - Util.h, [1373](#)
- mcptr
 - MMAccessor, [555](#)
- MCValueEditControl, [547](#)
 - MCValueEditControl, [548](#)
- MCValueEditControl
 - ~MCValueEditControl, [548](#)
 - doSelect, [548](#)
 - mcid, [548](#)
 - MCValueEditControl, [548](#)
- MCValueEditControl.h, [1232](#)
- mean
 - _afsLandmarkLoc, [140](#)
- mechacmd_callback
 - HeadPointControllerBehavior, [444](#)
 - WalkControllerBehavior, [939](#)
- mechanicalLimits
 - ERS210Info, [50](#)
 - ERS220Info, [80](#)
 - ERS2xxInfo, [107](#)
- MidLEDMask
 - ERS210Info, [51](#)
 - ERS220Info, [81](#)
 - ERS2xxInfo, [107](#)
- MidLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- MidRLEDMask
 - ERS210Info, [51](#)
 - ERS220Info, [81](#)
 - ERS2xxInfo, [107](#)
- MidRLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- min3
 - Util.h, [1373](#)
- MIN_EXP_REGION_SIZE
 - Vision.h, [1385](#)
- MIN_EXP_RUN_LENGTH
 - Vision.h, [1385](#)
- MinMaxRange_t
 - ERS210Info, [41](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [98](#)
- minor
 - LedEngine, [481](#)
- MinRange
 - ERS210Info, [41](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [98](#)
- mix_mode
 - SoundManager, [798](#)
- MixMode_t
 - SoundManager, [788](#)
- MMAccessor, [549](#)
 - ~MMAccessor, [552](#)
 - checkin, [552](#), [553](#)
 - checkout, [553](#)
 - mc, [553](#)
 - mc_id, [555](#)
 - mcptr, [555](#)
 - MMAccessor, [551](#)
 - operator *, [553](#), [554](#)
 - operator->, [554](#)

- operator=, [554](#)
- operator[], [554](#), [555](#)
- MMAccessor.h, [1234](#)
- MMCombo, [556](#)
 - ~MMCombo, [561](#)
 - addRunLevel, [561](#)
 - BindCont, [561](#)
 - calcLEDValue, [561](#)
 - CloseCont, [561](#)
 - ConnectCont, [562](#)
 - DoDestroy, [562](#)
 - DoInit, [562](#)
 - DoStart, [562](#)
 - DoStop, [563](#)
 - etrans, [568](#)
 - eventTranslatorQueueMemRgn, [568](#)
 - GotEventTranslatorQueue, [563](#)
 - GotImage, [563](#)
 - GotMotionManager, [563](#)
 - GotMotionMsg, [563](#)
 - GotPowerEvent, [564](#)
 - GotSensorFrame, [564](#)
 - GotSoundManager, [564](#)
 - GotWorldState, [564](#)
 - InitRegion, [564](#)
 - isStopped, [568](#)
 - ledActivation, [568](#)
 - ListenCont, [565](#)
 - MMCombo, [561](#)
 - motmanMemRgn, [568](#)
 - NUM_COMMAND_VECTOR, [568](#)
 - num_open, [569](#)
 - observer, [569](#)
 - open, [569](#)
 - OpenPrimitives, [565](#)
 - operator=, [565](#)
 - primIDs, [569](#)
 - ProcessID, [699](#)
 - readyLevel, [569](#)
 - ReadyRegisterEventTranslatorQueue, [565](#)
 - ReadyRegisterMotionManager, [565](#)
 - ReadyRegisterWorldState, [566](#)
 - ReadySendJoins, [566](#)
 - ReceiveCont, [566](#)
 - region, [569](#)
 - RPOPENR_isReady, [566](#)
 - RPOPENR_isready, [570](#)
 - RPOPENR_notify, [567](#)
 - RPOPENR_ready, [567](#)
 - RPOPENR_send, [567](#)
 - runLevel, [570](#)
 - SendCont, [567](#)
 - SetupOutputs, [567](#)
 - soundManagerMemRgn, [570](#)
 - subject, [570](#)
 - worldStateMemRgn, [570](#)
- MMCombo.cc, [1236](#)
 - startupBehavior, [1238](#)
- MMCombo.h, [1239](#)
- MMlock
 - MotionManager, [598](#)
- mode
 - BehaviorActivatorControl, [227](#)
 - BehaviorSwitchActivatorControl, [237](#)
- Mode_t
 - BehaviorActivatorControl, [225](#)
 - BehaviorSwitchActivatorControl, [235](#)
- ModeLEDMask
 - ERS220Info, [81](#)
 - ERS2xxInfo, [108](#)
- ModeLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- mon
 - CompareTrans, [268](#)
- monitor_freq
 - FreeMemReportControl, [435](#)
- MONO16K16B_UNIT_SIZE
 - WAV, [973](#)
- MONO8K8B_UNIT_SIZE
 - WAV, [973](#)
- MOTION
 - _FastSLAM_update, [152](#)
- motion
 - Config, [273](#)

- motion_config
 - Config::motion_config, 282
- MotionCommand, 571
 - MotionCommand, 574
- MotionCommand
 - ~MotionCommand, 574
 - autoprune, 579
 - DoStart, 575
 - DoStop, 575
 - getAutoPrune, 575
 - interpolate, 575, 576
 - isActive, 577
 - isAlive, 577
 - isDirty, 577
 - MotionCommand, 574
 - postEvent, 577
 - queue, 579
 - setAutoPrune, 578
 - setQueue, 578
 - shouldPrune, 578
 - started, 579
 - updateOutputs, 578
- MotionCommand.cc, 1241
- MotionCommand.h, 1242
- MotionManager, 580
 - MotionManager, 587
 - MotionManagerMsg, 611
- MotionManager
 - _MMaccID, 596
 - accID_t, 586
 - addMotion, 587, 588
 - begin, 588
 - checkinMotion, 588
 - checkoutLevel, 588
 - checkoutMotion, 589
 - cmdlist, 596
 - cmds, 596
 - cmdstatelist_t, 586
 - cmdstates, 597
 - cmdSums, 597
 - cur_cmd, 597
 - end, 589
 - func_begin, 589
 - func_end, 589
 - getOutputCmd, 589
 - getOutputs, 590
 - getPriority, 590
 - InitAccess, 590
 - invalid_MC_ID, 597
 - kBackgroundPriority, 597
 - kEmergencyPriority, 597
 - kHighPriority, 597
 - kIgnoredPriority, 598
 - kLowPriority, 598
 - kStdPriority, 598
 - lock, 591
 - MAX_ACCESS, 598
 - MAX_MOTIONS, 598
 - MC_ID, 586
 - MMlock, 598
 - MotionManager, 587
 - next, 591
 - numAcc, 599
 - operator=, 591
 - peekMotion, 591
 - pidchanges, 599
 - pop_free, 591
 - push_free, 591
 - receivedMsg, 592
 - release, 592
 - removeMotion, 592
 - setOutput, 592–594
 - setPID, 594
 - setPriority, 595
 - size, 595
 - skip_ahead, 595
 - subjs, 599
 - trylock, 595
 - updatePIDs, 596
 - updateWorldState, 596
- MotionManager.cc, 1244
- MotionManager.cc
 - motman, 1245
 - uint, 1245
- MotionManager.h, 1246
- MotionManager.h
 - motman, 1249
- MotionManager::CommandEntry, 600
- MotionManager::CommandEntry
 - baseaddrs, 601
 - CommandEntry, 601
 - lastAccessor, 601

- lock, [601](#)
- operator=, [601](#)
- priority, [602](#)
- rcr, [602](#)
- MotionManager::OutputState, [603](#)
- MotionManager::OutputState
 - frames, [605](#)
 - mcid, [605](#)
 - OutputState, [604](#)
 - pid, [605](#)
 - priority, [605](#)
- MotionManager::PIDUpdate, [606](#)
- MotionManager::PIDUpdate
 - joint, [607](#)
 - pids, [607](#)
 - PIDUpdate, [606](#)
- MotionManagerMsg, [608](#)
 - addMotion, [610](#)
 - deleteMotion, [610](#)
 - MotionManagerMsg, [610](#)
 - unknown, [610](#)
- MotionManagerMsg
 - ~MotionManagerMsg, [610](#)
 - getID, [610](#)
 - MC_ID, [609](#)
 - mc_id, [611](#)
 - MotionManager, [611](#)
 - MotionManagerMsg, [610](#)
 - MsgType, [609](#)
 - setAdd, [610](#)
 - setDelete, [610](#)
 - type, [611](#)
- MotionManagerMsg.h, [1250](#)
- MotionProcess
 - ProcessID, [699](#)
- motionProfile
 - WorldState, [1017](#)
- MotionRequest, [612](#)
 - GO_TO, [612](#)
 - LOOK_AT, [612](#)
 - LOOK_DOWN_AT, [612](#)
- MotionRequest
 - altitude, [612](#)
 - azalt, [612](#)
 - azimuth, [613](#)
 - type, [613](#)
 - x, [613](#)
 - xy, [613](#)
 - y, [613](#)
- motionReshapeKludge.h, [1251](#)
- motionReshapeKludge.h
 - da, [1254](#)
 - DA_LEFT_MOT_THRESH, [1252](#)
 - DA_RIGHT_MOT_THRESH, [1252](#)
 - dx, [1254](#)
 - DX_MOT_THRESH, [1252](#)
 - if, [1254](#)
 - RESCALE_DA_LEFT_A, [1252](#)
 - RESCALE_DA_LEFT_B, [1253](#)
 - RESCALE_DA_RIGHT_A, [1253](#)
 - RESCALE_DA_RIGHT_B, [1253](#)
 - RESCALE_DX_A, [1253](#)
 - RESCALE_DX_B, [1253](#)
 - time, [1254](#)
- MotionSequence, [614](#)
 - MotionSequence, [620](#)
- MotionSequence
 - ~MotionSequence, [621](#)
 - calcOutput, [621](#)
 - ChkAdvance, [621](#)
 - clear, [621](#)
 - compress, [622](#)
 - curs, [629](#)
 - curstamps, [629](#)
 - endtime, [630](#)
 - eraseKeyFrame, [622](#)
 - getBinSize, [622](#)
 - getEndTime, [622](#)
 - getKeyFrame, [622](#), [623](#)
 - getMaxFrames, [623](#)
 - getOutputCmd, [623](#)
 - getOutputIndex, [623](#)
 - getPlaySpeed, [623](#)
 - getPlayTime, [624](#)
 - getUsedFrames, [624](#)
 - invalid_move, [630](#)
 - isAlive, [624](#)
 - isDirty, [624](#)
 - isPlaying, [625](#)
 - isSaveDegrees, [625](#)
 - isSaveRadians, [625](#)

- lasttime, [630](#)
- LoadBuffer, [625](#)
- loadSaveMode, [630](#)
- makeSafe, [625](#)
- MotionSequence, [620](#)
- Move_idx_t, [620](#)
- newKeyFrame, [626](#)
- nexts, [630](#)
- overlayPose, [626](#)
- pause, [626](#)
- play, [626](#)
- playing, [630](#)
- playspeed, [630](#)
- playtime, [631](#)
- prevs, [631](#)
- readWord, [626](#)
- resume, [627](#)
- SaveBuffer, [627](#)
- setNextFrameTime, [627](#)
- setOutputCmd, [627](#)
- setPlaySpeed, [628](#)
- setPlayTime, [628](#)
- setPose, [628](#)
- setRange, [628](#)
- setSaveDegrees, [628](#)
- setSaveRadians, [629](#)
- SizeLarge, [631](#)
- SizeMedium, [631](#)
- SizeSmall, [631](#)
- SizeTiny, [631](#)
- SizeXLarge, [632](#)
- starts, [632](#)
- updateOutputs, [629](#)
- MotionSequence::Move, [633](#)
- MotionSequence::Move
 - cmd, [634](#)
 - Move, [633](#)
 - next, [634](#)
 - prev, [634](#)
 - starttime, [634](#)
- MotionSequenceMC, [635](#)
 - MotionSequenceMC, [637](#)
- MotionSequenceMC
 - ~MotionSequenceMC, [637](#)
 - clear, [638](#)
 - eraseKeyFrame, [638](#)
 - getKeyFrame, [638](#)
 - getMaxFrames, [639](#)
 - getUsedFrames, [639](#)
 - list_t, [637](#)
 - MotionSequenceMC, [637](#)
 - moves, [640](#)
 - newKeyFrame, [639](#)
 - setRange, [639](#)
 - updateOutputs, [640](#)
- MotionSequenceMC.cc, [1255](#)
- MotionSequenceMC.h, [1256](#)
- motman
 - MotionManager.cc, [1245](#)
 - MotionManager.h, [1249](#)
- motmanEGID
 - EventBase, [362](#)
- motmanMemRgn
 - MMCombo, [568](#)
- MotorPowerSID
 - PowerSourceID, [124](#)
- MouthOffset
 - ERS210Info, [51](#)
 - ERS2xxInfo, [108](#)
- Move
 - MotionSequence::Move, [633](#)
- move
 - ALM, [168](#)
 - char_traits, [260](#)
- Move_idx_t
 - MotionSequence, [620](#)
- moves
 - DynamicMotionSequence, [343](#)
 - MotionSequenceMC, [640](#)
- moving
 - WorldModel2, [994](#)
- movingSince
 - WorldModel2, [994](#)
- MRvector
 - WorldModel2.h, [1430](#)
- ms_per_sec
 - TimeET, [865](#)
- mset
 - Util.h, [1373](#)
- MSG_SIZE
 - SoundManagerMsg, [809](#)
- MsgType

- MotionManagerMsg, 609
- SoundManagerMsg, 807
- mutexdebugout
 - MutexLock.h, 1259
- MutexLock, 642
 - MutexLock, 644
- MutexLock
 - do_try_lock, 644
 - doors, 647
 - doors_used, 647
 - forget, 645
 - get_lock_level, 645
 - init, 645
 - lock, 645
 - lockcount, 647
 - lookup, 646
 - MutexLock, 644
 - NO_OWNER, 647
 - owner, 646
 - owner_index, 648
 - release, 646
 - spin, 646
 - try_lock, 646
 - unlock, 647
- MutexLock.h, 1258
- MutexLock.h
 - mutexdebugout, 1259
- MutexLock::door_t, 649
- MutexLock::door_t
 - BL_in_use, 650
 - BL_ready, 650
 - door_t, 650
 - FCFS_in_use, 650
 - id, 650
 - next_turn_bit, 650
 - turn, 651
- mybeh
 - BehaviorSwitchControl, 243
- myOID
 - Wireless, 983
- mzero
 - Util.h, 1373
- name
 - ControlBase, 305
 - Profiler::SectionInfo, 712
- SoundManager::SoundData, 803
- StateNode, 842
- nameisgen
 - EventBase, 371
- nb
 - basic_iNetStream, 194
 - basic_ioNetStream, 199
 - basic_oNetStream, 212
- NE
 - CompareTrans, 267
- neck_range2xyz
 - almUtility.cc, 1061
 - almUtility.h, 1064
- neck_to_camera
 - Kinematics.h, 1211
- netbuf
 - ionetstream.h, 1195
- neutral
 - WalkMC::LegParam, 959
- new_after
 - ListMemBuf, 502
- new_back
 - ListMemBuf, 502
- new_before
 - ListMemBuf, 502
- new_front
 - ListMemBuf, 503
- newKeyFrame
 - DynamicMotionSequence, 342
 - MotionSequence, 626
 - MotionSequenceMC, 639
- NewLarge
 - SystemUtility.h, 1353
- newParticles
 - afsMain.cc, 1031
- NewSoundVectorData
 - SoundPlay, 815
- next
 - EventRouter::TimerEntry, 402
 - ListMemBuf, 503
 - ListMemBuf::entry_t, 510
 - MotionManager, 591
 - MotionSequence::Move, 634
 - OutputNode, 667
- next_id
 - SoundManager::PlayState, 801

- next_snd
 - Config::controller_config, 277
- next_turn_bit
 - MutexLock::door_t, 650
- nextEv_dur
 - Controller, 320
- nextEv_val
 - Controller, 320
- nextItem
 - Controller, 320
- nextItemFast
 - Controller, 320
- nexts
 - MotionSequence, 630
- NO_OWNER
 - MutexLock, 647
- no_power_p
 - BatteryMonitorBehavior, 223
- nodes
 - StateNode, 842
- none
 - LedEngine, 481
- NonUniformHermiteSplineSegment, 652
 - NonUniformHermiteSplineSegment, 652
- NonUniformHermiteSplineSegment
 - a, 653
 - b, 653
 - c, 653
 - create, 653
 - d, 653
 - eval, 653
 - eval_deriv, 653
 - NonUniformHermiteSplineSegment, 652
 - t, 653
- norm
 - GVector::vector2d, 893
 - GVector::vector3d, 899
- norm_angle
 - Util.h, 1374
- normalize
 - GVector::vector2d, 893
 - GVector::vector3d, 899
- normRand
 - afsUtility.cc, 1046
 - afsUtility.h, 1047
- NUHSPLINE
 - Spline.h, 1336
- NUHSPLINE_TEM
 - Spline.h, 1336
- nukeAndPaveCurrentMap
 - ALM, 168
- NullControl, 654
 - NullControl, 655
- NullControl
 - activate, 655
 - doNextItem, 655
 - doPrevItem, 656
 - doReadStdIn, 656
 - doSelect, 656
 - NullControl, 655
 - takeInput, 656
- NullControl.h, 1260
- num_colors
 - Vision, 915
- NUM_COMMAND_VECTOR
 - MMCombo, 568
- NUM_MARKERS
 - VisionInterface, 134
- num_open
 - MMCombo, 569
- num_regions
 - Vision, 915
- num_runs
 - Vision, 915
- num_tmaps
 - Vision, 916
- NUM_VEVENTS
 - Visiondefines.h, 1390
- NUM_VISION_OBJECTS
 - VisionInterface, 134
- numAcc
 - MotionManager, 599
- NumBinJoints
 - ERS210Info, 51
 - ERS220Info, 81
 - ERS2xxInfo, 108
- NumButtons
 - ERS210Info, 51
 - ERS220Info, 81

- ERS2xxInfo, [108](#)
- numCycling
 - LedEngine, [488](#)
- NumEarJoints
 - ERS210Info, [52](#)
 - ERS220Info, [81](#)
 - ERS2xxInfo, [108](#)
- numEGIDs
 - EventBase, [362](#)
- numETIDs
 - EventBase, [362](#)
- NumFrames
 - ERS210Info, [52](#)
 - ERS220Info, [81](#)
 - ERS2xxInfo, [108](#)
- NumHeadJoints
 - ERS210Info, [52](#)
 - ERS220Info, [82](#)
 - ERS2xxInfo, [108](#)
- NumLEDs
 - ERS210Info, [52](#)
 - ERS220Info, [82](#)
 - ERS2xxInfo, [109](#)
- NumLegJoints
 - ERS210Info, [52](#)
 - ERS220Info, [82](#)
 - ERS2xxInfo, [109](#)
- NumLegs
 - ERS210Info, [52](#)
 - ERS220Info, [82](#)
 - ERS2xxInfo, [109](#)
- NumMouthJoints
 - ERS210Info, [52](#)
 - ERS220Info, [82](#)
 - ERS2xxInfo, [109](#)
- NumOutputs
 - ERS210Info, [52](#)
 - ERS220Info, [82](#)
 - ERS2xxInfo, [109](#)
- NumPIDJoints
 - ERS210Info, [53](#)
 - ERS220Info, [82](#)
 - ERS2xxInfo, [109](#)
- NumPowerSIDs
 - PowerSourceID, [125](#)
- NumProcesses
 - ProcessID, [699](#)
- NumSensors
 - ERS210Info, [53](#)
 - ERS220Info, [83](#)
 - ERS2xxInfo, [109](#)
- NumSlowFrames
 - ERS210Info, [53](#)
 - ERS220Info, [83](#)
 - ERS2xxInfo, [110](#)
- numStyle_t
 - LedEngine, [480](#)
- NumTailJoints
 - ERS210Info, [53](#)
 - ERS220Info, [83](#)
 - ERS2xxInfo, [110](#)
- obj_info
 - Vision, [916](#)
- obj_port
 - Config::vision_config, [287](#)
- observer
 - MMCombo, [569](#)
 - RemoteProcess, [735](#)
 - SoundPlay, [816](#)
- OFF_EDGE_BOTTOM
 - VisionInterface, [135](#)
- OFF_EDGE_LEFT
 - VisionInterface, [135](#)
- OFF_EDGE_RIGHT
 - VisionInterface, [135](#)
- OFF_EDGE_TOP
 - VisionInterface, [135](#)
- off_type
 - basic_iNetStream, [191](#)
 - basic_ioNetStream, [196](#)
 - basic_netbuf, [201](#)
 - basic_oNetStream, [209](#)
 - char_traits, [259](#)
- offset
 - LedEngine::LEDInfo, [490](#)
 - SharedQueue::entry_t, [760](#)
 - SoundManager::PlayState, [801](#)
- offset_to_line
 - GVector, [118](#)
- onedigit
 - LedEngine, [480](#)

- oNetStream
 - ionetstream.h, [1195](#)
- OObject, [659](#)
- open
 - basic_iNetStream, [193](#)
 - basic_ioNetStream, [198](#)
 - basic_netbuf, [204](#), [205](#)
 - basic_oNetStream, [211](#)
 - MMCombo, [569](#)
- OpenPrimitives
 - MMCombo, [565](#)
- OpenSpeaker
 - SoundPlay, [815](#)
- operator *
 - GVector::vector2d, [893](#)
 - GVector::vector3d, [899](#), [900](#)
 - MMAccessor, [553](#), [554](#)
 - SharedObject, [750](#)
- operator *=
 - GVector::vector2d, [893](#)
 - GVector::vector3d, [900](#)
- operator !=
 - GVector::vector2d, [893](#)
 - GVector::vector3d, [900](#)
 - OutputCmd, [663](#)
- operator()
 - dmPickCluster, [322](#)
 - dmPickColor, [323](#)
 - dmPickConfidence, [324](#)
 - dmPickDepth, [325](#)
 - dmPicker, [326](#)
 - EventRouter::TimerEntryPtr-
 - Cmp, [404](#)
 - hmPickCluster, [462](#)
 - hmPickColor, [463](#)
 - hmPickConfidence, [464](#)
 - hmPicker, [465](#)
 - hmPickHeight, [466](#)
 - hmPickTrav, [467](#)
- operator+
 - GVector::vector2d, [893](#)
 - GVector::vector3d, [900](#)
 - TimeET, [863](#)
 - TimeET.h, [1362](#)
- operator +=
 - GVector::vector2d, [893](#)
 - GVector::vector3d, [900](#)
- TimeET, [863](#)
- operator-
 - GVector::vector2d, [893](#), [894](#)
 - GVector::vector3d, [900](#)
 - TimeET, [863](#)
 - TimeET.h, [1362](#)
- operator-=
 - GVector::vector2d, [894](#)
 - GVector::vector3d, [901](#)
 - TimeET, [863](#)
- operator->
 - MMAccessor, [554](#)
 - SharedObject, [750](#)
- operator/
 - GVector::vector2d, [894](#)
 - GVector::vector3d, [901](#)
- operator/=
 - GVector::vector2d, [894](#)
 - GVector::vector3d, [901](#)
- operator<
 - EventBase, [367](#)
 - GVector::vector2d, [894](#)
 - GVector::vector3d, [901](#)
 - TimeET, [863](#), [864](#)
- operator<<
 - TimeET, [865](#)
 - TimeET.h, [1362](#)
- operator<=
 - GVector::vector2d, [894](#)
 - GVector::vector3d, [901](#)
- operator=
 - Aibo3DControllerBehavior, [162](#)
 - BanditMachine, [178](#)
 - BanditMachine::DecideNode, [182](#)
 - BatteryMonitorBehavior, [221](#)
 - BehaviorActivatorControl, [227](#)
 - BehaviorSwitchActivatorControl, [237](#)
 - BehaviorSwitchControl, [242](#)
 - BehaviorSwitchControlBase, [247](#)
 - BehaviorSwitchControl-
 - Base::BehaviorGroup, [250](#)
 - CompareTrans, [268](#)

- ControlBase, 302
- Controller, 316
- EStopControllerBehavior, 355
- EventRouter::EventManager, 398
- EventRouter::TimerEntry, 401
- EventTranslator, 408
- GVector::vector2d, 894
- GVector::vector3d, 901
- HeadPointControllerBehavior, 444
- HelpControl, 458
- MMAccessor, 554
- MMCombo, 565
- MotionManager, 591
- MotionManager::Command-Entry, 601
- OutputNode, 667
- Profiler::Timer, 716
- SharedObjectBase, 752
- SmoothCompareTrans, 768
- Socket, 775
- SoundManager, 793
- SoundManager::SoundData, 803
- SoundManagerMsg, 808
- StateNode, 840
- Transition, 877
- ValueEditControl, 883
- ValueSetControl, 889
- Vision, 911
- VisionSerializer, 929
- WalkControllerBehavior, 939
- WalkToTargetMachine, 968
- Wireless, 981
- WorldModel2, 991
- WorldModel2Behavior, 998
- WorldModel2Behavior::Gawk-Node, 1002
- WorldState, 1015
- WorldStateSerializer, 1021
- operator==
 - EventBase, 367
 - GVector::vector2d, 894
 - GVector::vector3d, 901
 - OutputCmd, 663
- operator>
 - GVector::vector2d, 895
 - GVector::vector3d, 901
- operator>=
 - GVector::vector2d, 895
 - GVector::vector3d, 901
- operator[]
 - ListMemBuf, 503
 - MMAccessor, 554, 555
 - SharedObject, 750
- options
 - ControlBase, 305
- ostream, 660
- out
 - OutputNode, 667
- out_flush
 - basic_netbuf, 205
- outCountAvgColor
 - Vision, 916
- outCountColorArea
 - Vision, 916
- outCountRaw
 - Vision, 916
- outCountRLE
 - Vision, 916
- OutputCmd, 661
 - OutputCmd, 662
- OutputCmd
 - operator!=, 663
 - operator==, 663
 - OutputCmd, 662
 - set, 663
 - unset, 663
 - value, 664
 - weight, 664
- OutputCmd.h, 1262
- outputNameLen
 - ERS210Info, 53
 - ERS220Info, 83
 - ERS2xxInfo, 110
- outputNames
 - ERS210Info, 53
 - ERS220Info, 83
 - ERS2xxInfo, 110
- OutputNode, 665
 - OutputNode, 666
- OutputNode
 - DoStart, 667

- next, [667](#)
 - operator=, [667](#)
 - out, [667](#)
 - OutputNode, [666](#)
- OutputNode.h, [1263](#)
- OutputPID, [668](#)
 - OutputPID, [669](#)
- OutputPID
 - OutputPID, [669](#)
 - pid, [670](#)
 - set, [669](#), [670](#)
 - set_pid, [670](#)
 - unset, [670](#)
 - weight, [670](#)
- OutputPID.h, [1265](#)
- outputRanges
 - ERS210Info, [53](#)
 - ERS220Info, [83](#)
 - ERS2xxInfo, [110](#)
- outputs
 - WorldState, [1017](#)
- OutputState
 - MotionManager::OutputState, [604](#)
- OverChargedSID
 - PowerSourceID, [124](#)
- overflow
 - basic_netbuf, [205](#)
- OverheatingSID
 - PowerSourceID, [124](#)
- overlayPose
 - MotionSequence, [626](#)
- Override
 - SoundManager, [789](#)
- owner
 - MutexLock, [646](#)
- owner_index
 - MutexLock, [648](#)
- P
 - HeadPointControllerBehavior, [445](#)
- PanOffset
 - ERS210Info, [42](#)
 - ERS220Info, [70](#)
 - ERS2xxInfo, [99](#)
- pans
 - TailWagMC, [854](#)
- parent
 - StateNode, [842](#)
- Particles
 - afsMain.cc, [1031](#)
 - Test.c, [1357](#)
- ParticleSets
 - afsMain.cc, [1031](#)
- Path.h, [1266](#)
 - SPATH, [1267](#)
 - SPATH_TEM, [1267](#)
- paths
 - FileBrowserControl, [423](#)
- Pause
 - SoundManager, [789](#)
- pause
 - BatteryCheckControl, [215](#)
 - ControlBase, [302](#)
 - MotionSequence, [626](#)
 - ValueEditControl, [883](#)
- paused
 - EmergencyStopMC, [350](#)
- PausePlay
 - SoundManager, [793](#)
- PauseSID
 - PowerSourceID, [124](#)
- pauseWhileChin
 - SoundTestBehavior, [822](#)
- PBALL
 - VisionInterface, [135](#)
- pball
 - VisionInterface::ObjectInfo, [658](#)
- pct_from_mean
 - Vision.cc, [1381](#)
- peekMotion
 - MotionManager, [591](#)
- percentStyle_t
 - LedEngine, [481](#)
- period
 - EmergencyStopMC, [350](#)
 - LedEngine::LEDInfo, [490](#)
 - TailWagMC, [854](#)
 - WalkMC::WalkParam, [963](#)
- pickDMData
 - WorldModel2, [991](#)

- pickHMData
 - WorldModel2, [991](#)
- pid
 - MotionManager::OutputState, [605](#)
 - OutputPID, [670](#)
- pid_id
 - StartupBehavior, [833](#)
- pidchanges
 - MotionManager, [599](#)
- pidcutoff
 - EmergencyStopMC, [350](#)
- pidduties
 - WorldState, [1018](#)
- piddutyavgs
 - EmergencyStopMC, [350](#)
- PIDJointOffset
 - ERS210Info, [54](#)
 - ERS220Info, [84](#)
 - ERS2xxInfo, [111](#)
- PIDMC, [671](#)
 - ~PIDMC, [673](#)
 - dirty, [676](#)
 - getPID, [673](#), [674](#)
 - isAlive, [674](#)
 - isDirty, [674](#)
 - isPID, [674](#)
 - PIDMC, [673](#)
 - PIDs, [676](#)
 - setAllPowerLevel, [675](#)
 - setDefaults, [675](#)
 - setJointPowerLevel, [675](#)
 - setPID, [675](#)
 - setRangePowerLevel, [675](#)
 - updateOutputs, [676](#)
- PIDMC.h, [1268](#)
- PIDs
 - PIDMC, [676](#)
- pids
 - MotionManager::PIDUpdate, [607](#)
 - WorldState, [1018](#)
- PIDUpdate
 - MotionManager::PIDUpdate, [606](#)
- PinkBallSID
 - VisionEventNS, [131](#)
- pixel
 - Vision.h, [1386](#)
- Play
 - SoundManager, [793](#)
- play
 - MotionSequence, [626](#)
 - SoundTestBehavior, [821](#)
- Play_ID
 - SoundManager, [787](#)
- PlayBuffer
 - SoundManager, [794](#)
- PlayFile
 - SoundManager, [794](#)
- playing
 - MotionSequence, [630](#)
- playlist
 - SoundManager, [798](#)
- playlist.t
 - SoundManager, [788](#)
- PlaySoundControl, [677](#)
 - PlaySoundControl, [677](#)
- PlaySoundControl
 - ~PlaySoundControl, [678](#)
 - PlaySoundControl, [677](#)
 - selectedFile, [678](#)
- PlaySoundControl.h, [1270](#)
- playspeed
 - MotionSequence, [630](#)
- PlayState
 - SoundManager::PlayState, [800](#)
- playtime
 - MotionSequence, [631](#)
- PlungerSID
 - PowerSourceID, [125](#)
- point2d
 - Geometry.h, [1172](#)
- point2f
 - Geometry.h, [1172](#)
- point3d
 - Geometry.h, [1172](#)
- point3f
 - Geometry.h, [1172](#)
- point_on_segment
 - GVector, [119](#)
- poller

- CompareTrans, 268
- pop
 - Controller, 316
- pop_back
 - ListMemBuf, 503, 504
- pop_free
 - ListMemBuf, 504
 - MotionManager, 591
- pop_front
 - ListMemBuf, 504
- pos
 - Aibo3DControllerBehavior, 162
- pos_delta
 - WalkMC, 956
- pos_type
 - basic_iNetStream, 191
 - basic_ioNetStream, 196
 - basic_netbuf, 202
 - basic_oNetStream, 209
 - char_traits, 259
- pose
 - _afsParticle, 144
 - BatteryMonitorBehavior, 223
- pose_id
 - BatteryMonitorBehavior, 223
- Poses.h, 1272
 - DA_PAN, 1273
 - DA_ROLL, 1273
 - DA_TILT, 1273
 - DA_TOLERANCE, 1273
 - SP_LBK_JOINT, 1273
 - SP_LBK_KNEE, 1273
 - SP_LBK_SHLDR, 1273
 - SP_LFR_JOINT, 1273
 - SP_LFR_KNEE, 1273
 - SP_LFR_SHLDR, 1273
 - SP_RBK_JOINT, 1274
 - SP_RBK_KNEE, 1274
 - SP_RBK_SHLDR, 1274
 - SP_RFR_JOINT, 1274
 - SP_RFR_KNEE, 1274
 - SP_RFR_SHLDR, 1274
 - SP_TOLERANCE, 1274
- postEvent
 - EventRouter, 388
 - MotionCommand, 577
- PostureEngine, 679
 - PostureEngine, 682
- PostureEngine
 - ~PostureEngine, 682
 - avgdiff, 682
 - ChkAdvance, 683
 - clear, 683
 - cmds, 688
 - createAverage, 683
 - createCombine, 684
 - createOverlay, 684
 - createUnderlay, 684
 - diff, 684
 - getBinSize, 685
 - getOutputCmd, 685
 - LoadBuffer, 685
 - maxdiff, 686
 - PostureEngine, 682
 - SaveBuffer, 686
 - setAverage, 686
 - setCombine, 687
 - setOutputCmd, 687
 - setOverlay, 688
 - setUnderlay, 688
 - takeSnapshot, 688
- PostureEngine.cc, 1275
- PostureEngine.h, 1276
- PostureMC, 690
 - PostureMC, 692
- PostureMC
 - ~PostureMC, 692
 - clear, 693
 - dirty, 697
 - getTolerance, 693
 - isAlive, 693
 - isDirty, 693, 694
 - LoadBuffer, 694
 - PostureMC, 692
 - setAverage, 694
 - setCombine, 695
 - setDirty, 695
 - setOutputCmd, 695
 - setOverlay, 695
 - setTolerance, 696
 - setUnderlay, 696
 - takeSnapshot, 696

- tolerance, [697](#)
- updateOutputs, [696](#)
- PostureMC.h, [1278](#)
- PowerCapacityOffset
 - ERS210Info, [42](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [99](#)
- PowerCurrentOffset
 - ERS210Info, [42](#)
 - ERS220Info, [70](#)
 - ERS2xxInfo, [99](#)
- powerEGID
 - EventBase, [362](#)
- powerFlags
 - WorldState, [1018](#)
- PowerGoodSID
 - PowerSourceID, [124](#)
- PowerRemainOffset
 - ERS210Info, [42](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [99](#)
- PowerSourceID, [123](#)
 - BatteryConnectSID, [124](#)
 - BatteryEmptySID, [124](#)
 - BatteryFullSID, [124](#)
 - BatteryInitSID, [124](#)
 - BatteryOverCurrentSID, [124](#)
 - BMNDebugModeSID, [125](#)
 - ChargerStatusSID, [124](#)
 - ChargingSID, [124](#)
 - DataFromStationSID, [124](#)
 - DischargingSID, [124](#)
 - ErrorSID, [124](#)
 - ExternalPortSID, [124](#)
 - ExternalPowerSID, [124](#)
 - LowPowerWarnSID, [124](#)
 - MotorPowerSID, [124](#)
 - NumPowerSIDs, [125](#)
 - OverChargedSID, [124](#)
 - OverheatingSID, [124](#)
 - PauseSID, [124](#)
 - PlungerSID, [125](#)
 - PowerGoodSID, [124](#)
 - RegisterUpdateSID, [124](#)
 - RTCSID, [125](#)
 - SpecialModeSID, [125](#)
 - StationConnectSID, [124](#)
 - SuspendedSID, [124](#)
 - TermChargeSID, [124](#)
 - TermDischargeSID, [124](#)
 - UpdatedSID, [125](#)
 - VibrationSID, [124](#)
- PowerSourceID
 - PowerSourceID_t, [123](#)
- PowerSourceID_t
 - PowerSourceID, [123](#)
- PowerThermoOffset
 - ERS210Info, [42](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [99](#)
- PowerVoltageOffset
 - ERS210Info, [42](#)
 - ERS220Info, [70](#)
 - ERS2xxInfo, [99](#)
- pprintf
 - Socket, [775](#)
- preload
 - Config::sound_config, [285](#)
- present
 - VisionEventSpec, [924](#)
- press_id
 - BanditMachine::PressNode, [185](#)
- PressNode
 - BanditMachine::PressNode, [184](#)
- prev
 - ListMemBuf, [505](#)
 - ListMemBuf::entry_t, [510](#)
 - MotionSequence::Move, [634](#)
- prev_snd
 - Config::controller_config, [277](#)
- prevEv_dur
 - Controller, [320](#)
- prevEv_val
 - Controller, [320](#)
- prevItem
 - Controller, [321](#)
- prevItemFast
 - Controller, [321](#)
- prevs
 - MotionSequence, [631](#)
- PRIMED
 - _afsLandmarkLoc, [140](#)

- primIDs
 - MMCombo, 569
- PRIMING
 - _afsLandmarkLoc, 140
- priming
 - _afsLandmarkLoc, 140
- PrimitiveName
 - ERS210Info, 54
 - ERS220Info, 84
 - ERS2xxInfo, 111
- printBuffer
 - basic_netbuf, 205
- printf
 - Socket, 775
- priority
 - MotionManager::Command-Entry, 602
 - MotionManager::OutputState, 605
- processEvent
 - AutoGetupBehavior, 173
 - BanditMachine::WaitNode, 188
 - BatteryCheckControl, 215
 - BatteryMonitorBehavior, 221
 - BehaviorBase, 232
 - CameraBehavior, 257
 - ChaseBallBehavior, 263
 - CompareTrans, 268
 - Controller, 316
 - DriveMeBehavior, 330
 - DumbWM2Behavior, 334
 - EStopControllerBehavior, 355
 - EventListener, 373
 - EventLogger, 376
 - EventRouter, 388
 - EvtRptBehavior, 414
 - FollowHeadBehavior, 428
 - FreeMemReportControl, 433
 - HeadLevelBehavior, 438
 - HeadPointControllerBehavior, 444
 - LoadPostureControl, 512
 - RunSequenceControl, 738
 - SimpleChaseBallBehavior, 765
 - SmoothCompareTrans, 768
 - SoundTestBehavior, 821
 - StareAtBallBehavior, 828
 - StartupBehavior, 833
 - StateNode, 840
 - TimeOutTrans, 868
 - ValueEditControl, 883
 - VisualTargetCloseTrans, 931
 - WalkControllerBehavior, 939
 - WalkToTargetMachine, 968
 - WorldModel2, 991
 - WorldModel2Behavior::Gawk-Node, 1002
 - WorldModel2Behavior::Walk-Node, 1009
- processEventBuffer
 - EventRouter, 389
- processFrame
 - Vision, 911
- processGround
 - WorldModel2, 992
- ProcessID, 698
 - MainProcess, 699
 - MotionProcess, 699
 - NumProcesses, 699
 - SoundProcess, 699
- ProcessID
 - getID, 699
 - ID, 700
 - MMCombo, 699
 - ProcessID.t, 699
 - setID, 699
 - SoundPlay, 700
- ProcessID.cc, 1280
- ProcessID.h, 1281
- ProcessID.t
 - ProcessID, 699
- processLocomotion
 - WorldModel2, 992
- processSensors
 - WorldModel2, 992
- processTimers
 - EventRouter, 389
- Profiler, 701
 - buckets, 707
 - curSection, 707
 - finished, 705
 - gamma, 707

- getBucket, 705
- getBuckets, 705
- getInfos, 705
- getNewID, 706
- HistCurve, 707
- HistSize, 707
- HistTime, 708
- infosOffset, 708
- initBuckets, 706
- MaxSectionNameLen, 708
- maxSections, 708
- Profiler, 705
- Profiler::Timer, 717
- report, 706
- reset, 706
- sectionsUsed, 708
- setCurrent, 706
- startTime, 708
- Timer, 707
- Profiler.cc, 1282
- Profiler.h, 1283
 - PROFSECTION, 1284
- Profiler::SectionInfo, 710
- Profiler::SectionInfo
 - calls, 711
 - childTime, 711
 - execExpAvg, 712
 - execHist, 712
 - interExpAvg, 712
 - interHist, 712
 - lastTime, 712
 - name, 712
 - reset, 711
 - SectionInfo, 711
 - totalInterval, 712
 - totalTime, 713
- Profiler::Timer, 714
 - ~Timer, 716
 - _id, 717
 - _parent, 717
 - _prof, 717
 - _t, 717
 - elapsed, 716
 - operator=, 716
 - Profiler, 717
 - setID, 716
 - start, 716
 - startTime, 717
 - Timer, 715, 716
- ProfilerCheckControl, 719
 - ProfilerCheckControl, 719
- ProfilerCheckControl
 - ~ProfilerCheckControl, 719
 - activate, 720
 - ProfilerCheckControl, 719
- ProfilerCheckControl.h, 1286
- ProfilerOfSize, 721
 - ProfilerOfSize, 721
- ProfilerOfSize
 - infos, 722
 - ProfilerOfSize, 721
- PROFSECTION
 - Profiler.h, 1284
- push
 - Controller, 316
- push_after
 - ListMemBuf, 505
- push_back
 - ListMemBuf, 505
- push_before
 - ListMemBuf, 505
- push_free
 - ListMemBuf, 505
 - MotionManager, 591
- push_front
 - ListMemBuf, 506
- pushSlot
 - ControlBase, 302
- Quality
 - SoundManager, 788
- queue
 - EventTranslator, 409
 - MotionCommand, 579
- queue_mode
 - SoundManager, 798
- Queue_t
 - EventTranslator, 407
- QueueMode_t
 - SoundManager, 788
- R

- Maps/Configuration.h, [1101](#)
- r
 - BanditMachine::DecideNode, [182](#)
 - HeadPointControllerBehavior, [445](#)
 - karmedbanditExp3_1, [476](#)
- RAD
 - CameraBehavior, [257](#)
 - ERS210Info.h, [1129](#)
 - ERS220Info.h, [1131](#)
 - ERS2xxInfo.h, [1133](#)
 - Kinematics.h, [1209](#)
- raw_port
 - Config::vision_config, [287](#)
- RBALL
 - VisionInterface, [135](#)
- rball
 - VisionInterface::ObjectInfo, [658](#)
- RBk
 - SoundTestBehavior, [822](#)
- RBkLegOffset
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- RBkLegOrder
 - ERS210Info, [41](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [98](#)
- RBkPawOffset
 - ERS210Info, [39](#)
 - ERS220Info, [66](#)
 - ERS2xxInfo, [96](#)
- RBkPawSID
 - ButtonSourceID, [30](#)
- RC_autodelete
 - ReferenceCounter, [727](#)
- rcontrol_id
 - Aibo3DControllerBehavior, [162](#)
- rcr
 - MotionManager::Command-Entry, [602](#)
 - SharedObjectBase, [753](#)
 - SoundManager::SoundData, [803](#)
- rcvcbckfn
 - Socket, [779](#)
- rdbuf
 - basic_iNetStream, [193](#)
 - basic_ioNetStream, [198](#)
 - basic_oNetStream, [211](#)
- read
 - Socket, [776](#)
 - WorldState, [1015](#)
- read_file
 - WalkMC.cc, [1406](#)
- read_snd
 - Config::controller_config, [277](#)
- readConfig
 - Config, [272](#)
- readData
 - Socket, [779](#)
- readSize
 - Socket, [779](#)
- readWord
 - MotionSequence, [626](#)
- readyLevel
 - MMCombo, [569](#)
- ReadyRegisterEventTranslatorQueue
 - MMCombo, [565](#)
- ReadyRegisterMotionManager
 - MMCombo, [565](#)
- ReadyRegisterSoundManager
 - SoundPlay, [815](#)
- ReadyRegisterWorldState
 - MMCombo, [566](#)
- ReadySendJoints
 - MMCombo, [566](#)
- ReadySendSound
 - SoundPlay, [815](#)
- realmon
 - SmoothCompareTrans, [768](#)
- RebootControl, [723](#)
 - RebootControl, [724](#)
- RebootControl
 - activate, [724](#)
 - doSelect, [724](#)
 - RebootControl, [724](#)
- RebootControl.cc, [1288](#)
- RebootControl.h, [1289](#)
- rebuildmenu
 - FileBrowserControl, [422](#)
- receive

- Wireless, [981](#)
- ReceiveCont
 - MMCombo, [566](#)
 - Wireless, [981](#)
- ReceivedMsg
 - SoundManager, [794](#)
- receivedMsg
 - MotionManager, [592](#)
- recurse
 - FileBrowserControl, [423](#)
- recvBuffer
 - Socket, [779](#)
- recvBufSize
 - Socket, [779](#)
- recvData
 - Socket, [779](#)
- recvSize
 - Socket, [779](#)
- RedBallSID
 - VisionEventNS, [131](#)
- ref
 - SoundManager::SoundData, [804](#)
- ReferenceCounter, [725](#)
 - ReferenceCounter, [726](#)
- ReferenceCounter
 - ~ReferenceCounter, [726](#)
 - AddReference, [726](#)
 - GetAutoDelete, [726](#)
 - GetReferences, [726](#)
 - RC_autodelete, [727](#)
 - ReferenceCounter, [726](#)
 - references, [727](#)
 - RemoveReference, [727](#)
 - SetAutoDelete, [727](#)
- ReferenceCounter.h, [1291](#)
- references
 - ReferenceCounter, [727](#)
- refresh
 - BatteryCheckControl, [215](#)
 - ControlBase, [302](#)
 - Controller, [317](#)
 - EventLogger, [376](#)
 - FreeMemReportControl, [434](#)
 - StringInputControl, [846](#)
- reg
 - Vision, [916](#)
 - VisionObjectInfo, [926](#)
- reg2
 - VisionObjectInfo, [926](#)
- region
 - MMCombo, [569](#)
 - SoundManagerMsg, [809](#)
 - SoundPlay, [816](#)
 - Vision.h, [1386](#)
- registerData
 - Aibo3DControllerBehavior, [162](#)
- registerDepth
 - ALM, [169](#)
- registerGround
 - ALM, [169](#)
- RegisterUpdateSID
 - PowerSourceID, [124](#)
- regs
 - Marker, [546](#)
- REKOffset.t
 - ERS210Info, [41](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [98](#)
- Release
 - SoundManager, [795](#)
- release
 - MotionManager, [592](#)
 - MutexLock, [646](#)
- ReleaseFile
 - SoundManager, [795](#)
- releaseJoints
 - EmergencyStopMC, [348](#)
- RemoteControllerMC, [728](#)
 - RemoteControllerMC, [729](#)
- RemoteControllerMC
 - ~RemoteControllerMC, [729](#)
 - active, [730](#)
 - cmds, [730](#)
 - dirty, [731](#)
 - isAlive, [729](#)
 - isDirty, [730](#)
 - RemoteControllerMC, [729](#)
 - setDirty, [730](#)
 - updateOutputs, [730](#)
- RemoteControllerMC.h, [1293](#)
- RemoteProcess, [732](#)
 - RemoteProcess, [733](#)

- RemoteProcess
 - ~RemoteProcess, [733](#)
 - data_received, [734](#)
 - DoDestroy, [734](#)
 - DoInit, [734](#)
 - DoStart, [734](#)
 - DoStop, [734](#)
 - observer, [735](#)
 - RemoteProcess, [733](#)
 - RPOPENR_isReady, [734](#)
 - RPOPENR_isready, [735](#)
 - RPOPENR_notify, [735](#)
 - RPOPENR_ready, [735](#)
 - RPOPENR_send, [735](#)
 - start, [735](#)
 - subject, [736](#)
- RemoteProcess.cc, [1295](#)
- RemoteProcess.h, [1296](#)
- removeAllTimers
 - EventRouter, [389](#)
- removeListener
 - EventRouter, [389](#), [390](#)
- removeMapping
 - EventRouter::EventMapper, [398](#), [399](#)
- removeMotion
 - MotionManager, [592](#)
- removePrefix
 - Controller, [317](#)
- RemoveReference
 - ReferenceCounter, [727](#)
- removeTimer
 - EventRouter, [390](#), [391](#)
- removeTrapper
 - EventRouter, [391](#), [392](#)
- repeat
 - EventRouter::TimerEntry, [402](#)
- report
 - BatteryCheckControl, [216](#)
 - FreeMemReportControl, [434](#)
 - HelpControl, [459](#)
 - Profiler, [706](#)
- report_freq
 - FreeMemReportControl, [435](#)
- RESCALE_DA_LEFT_A
 - motionReshapeKludge.h, [1252](#)
- RESCALE_DA_LEFT_B
 - motionReshapeKludge.h, [1253](#)
- RESCALE_DA_RIGHT_A
 - motionReshapeKludge.h, [1253](#)
- RESCALE_DA_RIGHT_B
 - motionReshapeKludge.h, [1253](#)
- RESCALE_DX_A
 - motionReshapeKludge.h, [1253](#)
- RESCALE_DX_B
 - motionReshapeKludge.h, [1253](#)
- reserve
 - SharedQueue, [757](#)
- reserved
 - WalkMC::WalkParam, [963](#)
- reset
 - Controller, [317](#)
 - EventRouter, [392](#)
 - karmedbanditExp3, [471](#)
 - Profiler, [706](#)
 - Profiler::SectionInfo, [711](#)
- resetLegPos
 - WalkMC, [952](#)
- resetName
 - EventBase, [367](#)
- resetTimer
 - TimeoutTrans, [868](#)
- resetTimerFreq
 - FreeMemReportControl, [434](#)
- resetWorldModel
 - WorldModel2, [992](#)
- resolution
 - Config::vision_config, [287](#)
- restart
 - karmedbanditExp3_1, [475](#)
- resume
 - MotionSequence, [627](#)
- ResumePlay
 - SoundManager, [795](#)
- retain
 - StateNode, [842](#)
- retained
 - BehaviorSwitchControlBase, [248](#)
- RetractableHeadLEDMask
 - ERS220Info, [85](#)
 - ERS2xxInfo, [112](#)
- RetractableHeadLEDOffset

- ERS220Info, [67](#)
- ERS2xxInfo, [97](#)
- reward
 - BanditMachine::WaitNode, [188](#)
 - karmedbanditExp3, [472](#)
 - karmedbanditExp3_1, [475](#)
- Rewind
 - WAV, [972](#)
- RFr
 - SoundTestBehavior, [822](#)
- RFrLegOffset
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- RFrLegOrder
 - ERS210Info, [41](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [98](#)
- RFrPawOffset
 - ERS210Info, [39](#)
 - ERS220Info, [66](#)
 - ERS2xxInfo, [96](#)
- RFrPawSID
 - ButtonSourceID, [30](#)
- right
 - VisionInterface::VObject, [933](#)
- rle_port
 - Config::vision_config, [288](#)
- rmap
 - Vision, [916](#)
- rmap2
 - Vision, [916](#)
- RobotArea
 - Vision.h, [1387](#)
- robotDesign
 - WorldState, [1018](#)
- RobotInfo, [126](#)
- RobotInfo.h, [1298](#)
- robotStatus
 - WorldState, [1018](#)
- RollOffset
 - ERS210Info, [42](#)
 - ERS220Info, [70](#)
 - ERS2xxInfo, [99](#)
- root
 - Config::motion_config, [283](#)
 - Config::sound_config, [285](#)
 - Controller, [321](#)
 - FileBrowserControl, [424](#)
 - HelpControl, [459](#)
- rotate
 - GVector::vector2d, [895](#)
- rotate_x
 - GVector::vector3d, [901](#)
- rotate_y
 - GVector::vector3d, [902](#)
- rotate_z
 - GVector::vector3d, [902](#)
- rotator_kmax
 - Kinematics.cc, [1204](#)
 - Kinematics.h, [1212](#)
- rotator_kmin
 - Kinematics.cc, [1205](#)
 - Kinematics.h, [1212](#)
- rotator_max
 - Kinematics.cc, [1205](#)
 - Kinematics.h, [1212](#)
- rotator_min
 - Kinematics.cc, [1205](#)
 - Kinematics.h, [1212](#)
- RotatorOffset
 - ERS210Info, [41](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [98](#)
- round_up
 - SharedQueue, [758](#)
- RPOPENR_isReady
 - MMCombo, [566](#)
 - RemoteProcess, [734](#)
- RPOPENR_isready
 - MMCombo, [570](#)
 - RemoteProcess, [735](#)
- RPOPENR_notify
 - MMCombo, [567](#)
 - RemoteProcess, [735](#)
- RPOPENR_ready
 - MMCombo, [567](#)
 - RemoteProcess, [735](#)
- RPOPENR_send
 - MMCombo, [567](#)
 - RemoteProcess, [735](#)
- RTCSID

- PowerSourceID, 125
- run
 - Vision.h, 1386
- runCommand
 - EStopControllerBehavior, 355
 - HeadPointControllerBehavior, 444
 - WalkControllerBehavior, 940
- RunForwardModel
 - Kinematics.cc, 1203
 - Kinematics.h, 1211
- runHighLevelVision
 - Vision, 912
- runLevel
 - MMCombo, 570
- runLowLevelVision
 - Vision, 912
- RunSequenceControl, 737
 - RunSequenceControl, 738
- RunSequenceControl
 - estopid, 739
 - processEvent, 738
 - RunSequenceControl, 738
 - selectedFile, 738
- RunSequenceControl.h, 1300
- rx
 - Socket, 780
- s
 - SplinePath, 824
- sameGenSource
 - EventBase, 367
- sample_bits
 - Config::sound_config, 285
- sample_rate
 - Config::sound_config, 285
- save
 - WalkMC, 952
- save_file
 - WalkMC.cc, 1406
- SaveBuffer
 - EventBase, 367
 - LoadSave, 533
 - LocomotionEvent, 543
 - MotionSequence, 627
 - PostureEngine, 686
 - TextMsgEvent, 857
 - VisionEvent, 922
- saveCreator
 - LoadSave, 534
- SaveFile
 - LoadSave, 535
- SavePostureControl, 740
 - SavePostureControl, 740
- SavePostureControl
 - SavePostureControl, 740
 - takeInput, 741
- SavePostureControl.h, 1302
- saveThresholdImage
 - Vision, 912
- SaveWalkControl, 742
 - SaveWalkControl, 743
- SaveWalkControl
 - ~SaveWalkControl, 743
 - activate, 743
 - deactivate, 743
 - SaveWalkControl, 743
 - walk_id, 743
- SaveWalkControl.h, 1304
- saw
 - Util.h, 1374
- sdistance
 - GVector, 119
- sec_behaviors
 - Config, 271
- sec_controller
 - Config, 271
- sec_invalid
 - Config, 271
- sec_main
 - Config, 271
- sec_motion
 - Config, 271
- sec_sound
 - Config, 271
- sec_vision
 - Config, 271
- sec_wireless
 - Config, 271
- sec_worldmodel2
 - Config, 271
- section_t

- Config, [271](#)
- SectionInfo
 - Profiler::SectionInfo, [711](#)
- sectionsUsed
 - Profiler, [708](#)
- select_snd
 - Config::controller_config, [277](#)
- selectChannels
 - SoundManager, [795](#)
- selectedFile
 - DumpFileControl, [337](#)
 - FileBrowserControl, [422](#)
 - LoadPostureControl, [513](#)
 - LoadWalkControl, [538](#)
 - PlaySoundControl, [678](#)
 - RunSequenceControl, [738](#)
- selectItem
 - Controller, [321](#)
- send
 - Wireless, [981](#)
- sendBuffer
 - Socket, [780](#)
- sendBufSize
 - Socket, [780](#)
- sendColorArea
 - Vision, [912](#)
- SendCont
 - MMCombo, [567](#)
 - Wireless, [982](#)
- sendData
 - Socket, [780](#)
- sendEvent
 - EventTranslator, [408](#)
- SendRawImage
 - VisionInterface, [133](#)
- sendRawImage
 - Vision, [912](#)
- SendRLEImage
 - VisionInterface, [133](#)
- sendRLEImage
 - Vision, [912](#)
- sendSize
 - Socket, [780](#)
- sensitivity
 - AutoGetupBehavior, [174](#)
- sensor_update
 - CameraBehavior, [258](#)
- sensorEGID
 - EventBase, [361](#)
- SensorOffset_t
 - ERS210Info, [41](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [98](#)
- sensors
 - WorldState, [1018](#)
- SensorSourceID, [127](#)
 - UpdatedSID, [127](#)
- SensorSourceID
 - SensorSourceID_t, [127](#)
- SensorSourceID_t
 - SensorSourceID, [127](#)
- serialize
 - VisionSerializer, [929](#)
 - WorldModel2, [993](#)
 - WorldStateSerializer, [1021](#)
- Serializer, [745](#)
 - encode, [745](#)
 - encodeDoublesAsFloats, [746](#)
 - hostToNetwork, [746](#)
- Serializer.h, [1306](#)
- serr
 - Socket.cc, [1319](#)
 - Socket.h, [1321](#)
- server_port
 - Socket, [780](#)
- Set
 - EventRouter::TimerEntry, [401](#)
 - TimeET, [864](#)
 - WAV, [972](#)
- set
 - GVector::vector2d, [895](#)
 - GVector::vector3d, [902](#)
 - LedEngine, [485](#)
 - OutputCmd, [663](#)
 - OutputPID, [669](#), [670](#)
- set_pid
 - OutputPID, [670](#)
- setActive
 - EmergencyStopMC, [348](#)
 - HeadPointerMC, [453](#)
 - TailWagMC, [852](#)
- setAdd

- MotionManagerMsg, [610](#)
 - SoundManagerMsg, [808](#)
- setAllPowerLevel
 - PIDMC, [675](#)
- setAngle
 - WalkMC, [952](#)
- SetAutoDelete
 - ReferenceCounter, [727](#)
- setAutoPrune
 - MotionCommand, [578](#)
- setAverage
 - PostureEngine, [686](#)
 - PostureMC, [694](#)
- setbits
 - Util.h, [1374](#)
- setBufferTime
 - EventRouter, [392](#)
- setCameraParam
 - Vision, [912](#)
- setCenterX
 - VisionEvent, [922](#)
- setCenterY
 - VisionEvent, [922](#)
- setColumn
 - LedEngine, [485](#)
- setCombine
 - PostureEngine, [687](#)
 - PostureMC, [695](#)
- setCurrent
 - Profiler, [706](#)
- setDbtTapDuration
 - EmergencyStopMC, [348](#)
- setDefault
 - ValueSetControl, [889](#)
- setDefaults
 - PIDMC, [675](#)
- setDelete
 - MotionManagerMsg, [610](#)
 - SoundManagerMsg, [808](#)
- setDescription
 - ControlBase, [302](#)
- setDirty
 - PostureMC, [695](#)
 - RemoteControllerMC, [730](#)
- setDisplay
 - ControlBase, [303](#)
- setDistance
 - VisionEvent, [922](#)
- setDuration
 - EventBase, [368](#)
- setEcho
 - basic_iNetStream, [193](#)
 - basic_ioNetStream, [198](#)
 - basic_netbuf, [205](#)
 - basic_oNetStream, [211](#)
- setEStopID
 - Controller, [317](#)
- setFilter
 - FileBrowserControl, [422](#)
- setFlipper
 - BatteryMonitorBehavior, [221](#)
- setFlushType
 - Socket, [776](#)
- setForward
 - Socket, [777](#)
- setGamma
 - karmedbanditExp3, [472](#)
- setGeneratorID
 - EventBase, [368](#)
- setHeight
 - WalkMC, [952](#)
- setHighlights
 - ControlBase, [303](#)
- setHop
 - WalkMC, [953](#)
- setID
 - ProcessID, [699](#)
 - Profiler::Timer, [716](#)
- setJointMode
 - HeadPointerMC, [453](#)
- setJointPowerLevel
 - PIDMC, [675](#)
- setJoints
 - HeadPointerMC, [453](#)
- setJointValue
 - HeadPointerMC, [453](#)
- setJointValueAndMode
 - HeadPointerMC, [453](#)
- setJointValueFromMode
 - HeadPointerMC, [454](#)
- setMagnitude
 - EventBase, [368](#)

- TailWagMC, 852
- SetMode
 - SoundManager, 796
- setMode
 - HeadPointerMC, 454
- setName
 - ControlBase, 303
 - EventBase, 369
 - StateNode, 841
- setNext
 - Controller, 317
- setNextFrameTime
 - MotionSequence, 627
- setNoiseThreshold
 - Vision, 912
- setOneOfTwo
 - LedEngine, 486
- setOutput
 - MotionManager, 592–594
- setOutputCmd
 - MotionSequence, 627
 - PostureEngine, 687
 - PostureMC, 695
- setOverlay
 - PostureEngine, 688
 - PostureMC, 695
- setPath
 - FileBrowserControl, 423
- setPaused
 - WalkMC, 953
- setPeriod
 - TailWagMC, 852
 - WalkMC, 953
- setPID
 - MotionManager, 594
 - PIDMC, 675
- setPlaySpeed
 - MotionSequence, 628
- setPlayTime
 - MotionSequence, 628
- setPose
 - MotionSequence, 628
- SetPowerAndVolume
 - SoundPlay, 815
- setPriority
 - MotionManager, 595
- setPrompt
 - StringInputControl, 847
- setProperty
 - VisionEvent, 922
- setQueue
 - EventTranslator, 408
 - MotionCommand, 578
- setRange
 - DynamicMotionSequence, 342
 - MotionSequence, 628
 - MotionSequenceMC, 639
- setRangePowerLevel
 - PIDMC, 675
- setReceiver
 - Wireless, 982
- setRecurse
 - FileBrowserControl, 423
- setResetSensitivity
 - EmergencyStopMC, 348
- setRetain
 - StateNode, 841
- setRoot
 - Controller, 318
 - FileBrowserControl, 423
- setSaveDegrees
 - MotionSequence, 628
- setSaveRadians
 - MotionSequence, 629
- setSlot
 - ControlBase, 303
- setSourceID
 - EventBase, 369
- setStatus
 - EventLogger, 376
- setStopped
 - EmergencyStopMC, 349
- setSway
 - WalkMC, 953
- setTarget
 - ValueEditControl, 884
 - ValueSetControl, 890
- setTargetVelocity
 - WalkMC, 953
- setText
 - TextMsgEvent, 858
- setTextForward

- Socket, [777](#)
- SetThreshold
 - VisionInterface, [133](#)
- setThreshold
 - Vision, [912](#)
- setTilt
 - TailWagMC, [853](#)
- setToken
 - TextMsgEvent, [858](#)
- setTolerance
 - PostureMC, [696](#)
- setTypeID
 - EventBase, [369](#)
- setUnderlay
 - PostureEngine, [688](#)
 - PostureMC, [696](#)
- setup
 - BanditMachine, [178](#)
 - StateNode, [841](#)
 - WalkToTargetMachine, [968](#)
 - WorldModel2Behavior, [998](#)
- SetupMenus
 - StartupBehavior, [833](#)
- SetupOutputs
 - MMCombo, [567](#)
- setVerbosity
 - Socket, [777](#)
- setWakeup
 - SoundManagerMsg, [808](#)
- setWeight
 - HeadPointerMC, [454](#)
- setWeights
 - LedMC, [493](#)
- setXYA
 - LocomotionEvent, [544](#)
- SharedObject, [747](#)
 - SharedObject, [748](#), [749](#)
- SharedObject
 - calcsize, [749](#)
 - dataCasted, [750](#)
 - operator *, [750](#)
 - operator->, [750](#)
 - operator[], [750](#)
 - SharedObject, [748](#), [749](#)
- SharedObject.h, [1307](#)
- SharedObjectBase, [751](#)
 - SharedObjectBase, [752](#)
- SharedObjectBase
 - data, [752](#)
 - getRegion, [752](#)
 - operator=, [752](#)
 - rcr, [753](#)
 - SharedObjectBase, [752](#)
- SharedQueue, [754](#)
 - SharedQueue, [756](#)
- SharedQueue
 - AutoLock, [756](#)
 - buf, [758](#)
 - buf_end, [757](#)
 - clear, [757](#)
 - data, [757](#)
 - done, [757](#)
 - entries, [758](#)
 - len, [759](#)
 - lock, [759](#)
 - MAX_ENTRIES, [759](#)
 - MAX_SIZE, [759](#)
 - reserve, [757](#)
 - round_up, [758](#)
 - SharedQueue, [756](#)
 - size, [758](#)
- SharedQueue.h, [1309](#)
- SharedQueue::entry_t, [760](#)
- SharedQueue::entry_t
 - offset, [760](#)
 - size, [760](#)
- shorterThan
 - EventBase, [369](#)
- shoulder_kmax
 - Kinematics.cc, [1205](#)
 - Kinematics.h, [1213](#)
- shoulder_kmin
 - Kinematics.cc, [1205](#)
 - Kinematics.h, [1213](#)
- shoulder_max
 - Kinematics.cc, [1205](#)
 - Kinematics.h, [1213](#)
- shoulder_min
 - Kinematics.cc, [1205](#)
 - Kinematics.h, [1213](#)
- shouldPrune
 - MotionCommand, [578](#)

- shouldWarn
 - BatteryMonitorBehavior, [221](#)
- showmanyc
 - basic_netbuf, [206](#)
- ShutdownControl, [761](#)
 - ShutdownControl, [762](#)
- ShutdownControl
 - activate, [762](#)
 - doSelect, [762](#)
 - ShutdownControl, [762](#)
- ShutdownControl.cc, [1311](#)
- ShutdownControl.h, [1312](#)
- shutter_speed
 - Config::vision_config, [288](#)
- sid
 - EventRouter::TimerEntry, [402](#)
 - VisualTargetCloseTrans, [932](#)
- side
 - AutoGetupBehavior, [174](#)
- SIDtoListenerVectorMap_t
 - EventRouter::EventManager, [396](#)
- sign
 - Util.h, [1374](#)
- SimpleChaseBallBehavior, [763](#)
 - SimpleChaseBallBehavior, [764](#)
- SimpleChaseBallBehavior
 - ~SimpleChaseBallBehavior, [764](#)
 - DoStart, [764](#)
 - DoStop, [764](#)
 - getName, [765](#)
 - processEvent, [765](#)
 - SimpleChaseBallBehavior, [764](#)
 - walker_id, [765](#)
- SimpleChaseBallBehavior.h, [1314](#)
- size
 - ListMemBuf, [506](#)
 - MotionManager, [595](#)
 - SharedQueue, [758](#)
 - SharedQueue::entry_t, [760](#)
- SizeLarge
 - MotionSequence, [631](#)
- SizeMedium
 - MotionSequence, [631](#)
- SizeSmall
 - MotionSequence, [631](#)
- SizeTiny
 - MotionSequence, [631](#)
- SizeXLarge
 - MotionSequence, [632](#)
- skip_ahead
 - MotionManager, [595](#)
- slotsSize
 - ControlBase, [303](#)
- SlowFrameTime
 - ERS210Info, [55](#)
 - ERS220Info, [85](#)
 - ERS2xxInfo, [113](#)
- SmoothCompareTrans, [766](#)
 - SmoothCompareTrans, [767](#)
- SmoothCompareTrans
 - avg, [768](#)
 - g, [768](#)
 - operator=, [768](#)
 - processEvent, [768](#)
 - realmon, [768](#)
 - SmoothCompareTrans, [767](#)
- SmoothCompareTrans.h, [1316](#)
- Snd_ID
 - SoundManager, [788](#)
 - SoundManagerMsg, [806](#)
- snd_id
 - SoundManager::PlayState, [801](#)
- sndlist
 - SoundManager, [798](#)
- sndlist_t
 - SoundManager, [788](#)
- sndman
 - SoundManager.cc, [1323](#)
 - SoundManager.h, [1327](#)
- sock
 - basic_netbuf, [207](#)
 - Socket, [780](#)
- SOCK_DGRAM
 - SocketNS, [129](#)
- sock_num
 - Wireless, [983](#)
- SOCK_STREAM
 - SocketNS, [129](#)
- sockDM
 - WorldModel2, [995](#)
- Socket, [770](#)
 - ~Socket, [774](#)

- endpoint, [778](#)
- flType, [778](#)
- flush, [774](#)
- forwardSock, [779](#)
- getReadBuffer, [774](#)
- getWriteBuffer, [774](#)
- init, [775](#)
- operator=, [775](#)
- pprintf, [775](#)
- printf, [775](#)
- rcvcbckfn, [779](#)
- read, [776](#)
- readData, [779](#)
- readSize, [779](#)
- recvBuffer, [779](#)
- recvBufSize, [779](#)
- recvData, [779](#)
- recvSize, [779](#)
- rx, [780](#)
- sendBuffer, [780](#)
- sendBufSize, [780](#)
- sendData, [780](#)
- sendSize, [780](#)
- server_port, [780](#)
- setFlushType, [776](#)
- setForward, [777](#)
- setTextForward, [777](#)
- setVerbosity, [777](#)
- sock, [780](#)
- Socket, [773](#), [774](#)
- state, [781](#)
- textForward, [781](#)
- textForwardBuf, [781](#)
- trType, [781](#)
- tx, [781](#)
- verbosity, [781](#)
- vprintf, [777](#)
- Wireless, [778](#)
- write, [777](#), [778](#)
- writeData, [781](#)
- writeSize, [782](#)
- socket
 - Wireless, [982](#), [983](#)
- Socket.cc, [1318](#)
 - serr, [1319](#)
 - sout, [1319](#)
- Socket.h, [1320](#)
 - serr, [1321](#)
 - sout, [1321](#)
- SocketNS, [128](#)
 - CONNECTION_CLOSED, [128](#)
 - CONNECTION_CLOSING, [128](#)
 - CONNECTION_CONNECTED, [128](#)
 - CONNECTION_-CONNECTING, [128](#)
 - CONNECTION_ERROR, [128](#)
 - CONNECTION_LISTENING, [128](#)
 - FLUSH_BLOCKING, [129](#)
 - FLUSH_NONBLOCKING, [129](#)
 - SOCK_DGRAM, [129](#)
 - SOCK_STREAM, [129](#)
- SocketNS
 - ConnectionState, [128](#)
 - FlushType_t, [128](#)
 - TransportType_t, [129](#)
- sockets
 - Wireless, [984](#)
- sockHM
 - WorldModel2, [995](#)
- sort
 - Util.h, [1374](#)
- sound
 - Config, [273](#)
- sound_config
 - Config::sound_config, [284](#)
- SOUND_NUM_BUFFER
 - SoundPlay, [816](#)
- SoundBufferTime
 - ERS210Info, [55](#)
 - ERS220Info, [86](#)
 - ERS2xxInfo, [113](#)
- SoundData
 - SoundManager::SoundData, [803](#)
- soundInfo
 - WAV, [973](#)
- SoundManager, [783](#)
 - Enqueue, [789](#)
 - Fast, [788](#)
 - Override, [789](#)
 - Pause, [789](#)

- Quality, 788
- SoundManager, 789
- SoundManagerMsg, 808
- Stop, 789
- SoundManager
 - Chain, 789
 - ChainBuffer, 789
 - ChainFile, 790
 - chanlist, 797
 - chanlist.t, 787
 - CopyTo, 790
 - endPlay, 790
 - GetNumPlaying, 791
 - GetRemainTime, 791
 - InitAccess, 791
 - initRegion, 791
 - invalid_Play_ID, 797
 - invalid_Snd_ID, 797
 - LoadBuffer, 792
 - LoadFile, 792
 - lock, 797
 - lookup, 792
 - lookupPath, 793
 - makePath, 793
 - max_chan, 797
 - MAX_NAME_LEN, 798
 - MAX_PLAY, 798
 - MAX_SND, 798
 - mix_mode, 798
 - MixMode.t, 788
 - operator=, 793
 - PausePlay, 793
 - Play, 793
 - Play_ID, 787
 - PlayBuffer, 794
 - PlayFile, 794
 - playlist, 798
 - playlist.t, 788
 - queue_mode, 798
 - QueueMode.t, 788
 - ReceivedMsg, 794
 - Release, 795
 - ReleaseFile, 795
 - ResumePlay, 795
 - selectChannels, 795
 - SetMode, 796
 - Snd_ID, 788
 - sndlist, 798
 - sndlist.t, 788
 - SoundManager, 789
 - StopPlay, 796
 - subjs, 798
 - updateChannels, 796
- SoundManager.cc, 1322
- SoundManager.cc
 - AutoLock, 1323
 - sndman, 1323
- SoundManager.h, 1325
- SoundManager.h
 - sndman, 1327
- SoundManager::PlayState, 800
- SoundManager::PlayState
 - cumulative, 801
 - next_id, 801
 - offset, 801
 - PlayState, 800
 - snd_id, 801
- SoundManager::SoundData, 802
- SoundManager::SoundData
 - data, 803
 - len, 803
 - name, 803
 - operator=, 803
 - rcr, 803
 - ref, 804
 - SoundData, 803
- soundManagerMemRgn
 - MMCombo, 570
 - SoundPlay, 817
- SoundManagerMsg, 805
 - add, 807
 - del, 807
 - SoundManagerMsg, 807
 - unknown, 807
 - wakeup, 807
- SoundManagerMsg
 - ~SoundManagerMsg, 807
 - getID, 807
 - id, 809
 - MSG_SIZE, 809
 - MsgType, 807
 - operator=, 808

- region, 809
- setAdd, 808
- setDelete, 808
- setWakeup, 808
- Snd_ID, 806
- SoundManager, 808
- SoundManagerMsg, 807
- type, 809
- SoundManagerMsg.h, 1328
- SoundPlay, 810
 - ProcessID, 700
 - SoundPlay, 813
- SoundPlay
 - ~SoundPlay, 813
 - active, 816
 - DoDestroy, 813
 - DoInit, 813
 - doSendSound, 813
 - DoStart, 814
 - DoStop, 814
 - etrans, 816
 - eventTranslatorQueueMemRgn, 816
 - FindFreeRegion, 814
 - GotEventTranslatorQueue, 814
 - GotSoundMsg, 814
 - InitRegion, 815
 - NewSoundVectorData, 815
 - observer, 816
 - OpenSpeaker, 815
 - ReadyRegisterSoundManager, 815
 - ReadySendSound, 815
 - region, 816
 - SetPowerAndVolume, 815
 - SOUND_NUM_BUFFER, 816
 - soundManagerMemRgn, 817
 - SoundPlay, 813
 - SPEAKER_LOCATOR, 817
 - speakerID, 817
 - subject, 817
- SoundPlay.cc, 1329
- SoundPlay.h, 1331
- SoundProcess
 - ProcessID, 699
- SoundTestBehavior, 818
 - SoundTestBehavior, 820
- SoundTestBehavior
 - Back, 821
 - curplay, 821
 - DoStart, 820
 - DoStop, 820
 - endtime, 821
 - getClassDescription, 820
 - getName, 820
 - LBk, 822
 - LFr, 822
 - pauseWhileChin, 822
 - play, 821
 - processEvent, 821
 - RBk, 822
 - RFr, 822
 - SoundTestBehavior, 820
- SoundTestBehavior.h, 1333
- soundUnitSize
 - WAV, 973
- sourceID
 - EventBase, 371
- sout
 - Socket.cc, 1319
 - Socket.h, 1321
- SP_LBK_JOINT
 - Poses.h, 1273
- SP_LBK_KNEE
 - Poses.h, 1273
- SP_LBK_SHLDR
 - Poses.h, 1273
- SP_LFR_JOINT
 - Poses.h, 1273
- SP_LFR_KNEE
 - Poses.h, 1273
- SP_LFR_SHLDR
 - Poses.h, 1273
- SP_RBK_JOINT
 - Poses.h, 1274
- SP_RBK_KNEE
 - Poses.h, 1274
- SP_RBK_SHLDR
 - Poses.h, 1274
- SP_RFR_JOINT
 - Poses.h, 1274
- SP_RFR_KNEE

- Poses.h, [1274](#)
- SP_RFR_SHLDR
 - Poses.h, [1274](#)
- SP_TOLERANCE
 - Poses.h, [1274](#)
- SPATH
 - Path.h, [1267](#)
- SPATH_TEM
 - Path.h, [1267](#)
- spawned
 - StartupBehavior, [833](#)
- SPEAKER_LOCATOR
 - SoundPlay, [817](#)
- speakerID
 - SoundPlay, [817](#)
- SpecialModeSID
 - PowerSourceID, [125](#)
- spin
 - MutexLock, [646](#)
- spline
 - WalkMC, [948](#)
- Spline.h, [1335](#)
 - EPS, [1336](#)
 - HSPLINE, [1336](#)
 - HSPLINE_TEM, [1336](#)
 - NUHSPLINE, [1336](#)
 - NUHSPLINE_TEM, [1336](#)
- SplinePath, [823](#)
 - SplinePath, [823](#)
- SplinePath
 - add, [824](#)
 - dx0, [824](#)
 - dx1, [824](#)
 - eval, [824](#)
 - eval_deriv, [824](#)
 - init, [824](#)
 - s, [824](#)
 - SplinePath, [823](#)
 - t0, [824](#)
 - t1, [824](#)
 - x0, [824](#)
 - x1, [824](#)
- splinepath
 - WalkMC, [948](#)
- sq_distort_coeff
 - Vision, [916](#)
- sqlength
 - GVector::vector2d, [895](#)
 - GVector::vector3d, [902](#)
- SQRT_2_PI
 - almMain.cc, [1055](#)
- squared
 - mathutils, [121](#)
- squareDistance
 - mathutils, [122](#)
- src
 - Transition, [877](#)
- SRLdelay
 - WorldModel2, [995](#)
- stampHM
 - ALM, [169](#)
- stand
 - DriveMeBehavior, [331](#)
- stand_id
 - DriveMeBehavior, [331](#)
 - WorldModel2Behavior::Gawk-Node, [1002](#)
 - WorldModel2Behavior::Wait-Node, [1005](#)
 - WorldModel2Behavior::Walk-Node, [1010](#)
- standUp
 - WorldModel2Behavior, [999](#)
- standUp_id
 - WorldModel2Behavior, [999](#)
- stare
 - BanditMachine, [178](#)
- StareAtBallBehavior, [826](#)
 - StareAtBallBehavior, [827](#)
- StareAtBallBehavior
 - ~StareAtBallBehavior, [827](#)
 - DoStart, [827](#)
 - DoStop, [827](#)
 - getClassDescription, [828](#)
 - getName, [828](#)
 - headpointer_id, [829](#)
 - processEvent, [828](#)
 - StareAtBallBehavior, [827](#)
- StareAtBallBehavior.cc, [1338](#)
- StareAtBallBehavior.cc
 - DtoR, [1338](#)
- StareAtBallBehavior.h, [1339](#)

- start
 - BanditMachine, 179
 - BehaviorActivatorControl, 225
 - BehaviorSwitchActivatorControl, 235
 - BehaviorSwitchControl, 242
 - BehaviorSwitchControlBase, 247
 - Profiler::Timer, 716
 - RemoteProcess, 735
 - WorldModel2Behavior, 999
- started
 - BehaviorBase, 232
 - MotionCommand, 579
- startmine
 - BehaviorSwitchControl, 242
- starts
 - MotionSequence, 632
- startTime
 - Profiler, 708
 - Profiler::Timer, 717
- starttime
 - LedEngine::LEDInfo, 490
 - MotionSequence::Move, 634
- StartupBehavior, 830
 - StartupBehavior, 831
- StartupBehavior
 - ~StartupBehavior, 831
 - DoStart, 832
 - DoStop, 832
 - getClassDescription, 832
 - getName, 832
 - pid_id, 833
 - processEvent, 833
 - SetupMenus, 833
 - spawned, 833
 - StartupBehavior, 831
 - stop_id, 833
- startupBehavior
 - MMCombo.cc, 1238
 - StartupBehavior.cc, 1344
 - StartupBehavior.h, 1345
- StartupBehavior.cc, 1341
- StartupBehavior.cc
 - gStartup, 1344
 - startupBehavior, 1344
- StartupBehavior.h, 1345
- StartupBehavior.h
 - startupBehavior, 1345
- startWarning
 - BatteryMonitorBehavior, 222
- state
 - _afsLandmarkLoc, 140
 - Socket, 781
 - WorldState.cc, 1438
 - WorldState.h, 1442
- stateMachineEGID
 - EventBase, 362
- StateNode, 835
 - StateNode, 838
- StateNode
 - ~StateNode, 838
 - addNode, 838
 - addTransition, 838
 - DoStart, 839
 - DoStop, 839
 - getDescription, 839
 - getName, 840
 - getNodes, 840
 - getTransitions, 840
 - issetup, 842
 - name, 842
 - nodes, 842
 - operator=, 840
 - parent, 842
 - processEvent, 840
 - retain, 842
 - setName, 841
 - setRetain, 841
 - setup, 841
 - StateNode, 838
 - transitionFrom, 841
 - transitions, 842
 - transitionTo, 841
- StateNode.cc, 1346
- StateNode.h, 1347
- StationConnectSID
 - PowerSourceID, 124
- statusETID
 - EventBase, 362
- std, 130
- stderr_port
 - Config::main_config, 280

- stilldown
 - EmergencyStopMC, [351](#)
- stim_id
 - EventBase, [371](#)
- Stop
 - SoundManager, [789](#)
- stop
 - BehaviorActivatorControl, [225](#)
 - BehaviorSwitchActivatorControl, [235](#)
 - BehaviorSwitchControl, [242](#)
 - BehaviorSwitchControlBase, [247](#)
- stop_id
 - StartupBehavior, [833](#)
- stopother
 - BehaviorSwitchControl, [243](#)
- StopPlay
 - SoundManager, [796](#)
- stopWarning
 - BatteryMonitorBehavior, [222](#)
- streambuf, [843](#)
- StringInputControl, [844](#)
 - StringInputControl, [845](#)
- StringInputControl
 - activate, [846](#)
 - doReadStdIn, [846](#)
 - getLastInput, [846](#)
 - lastInput, [847](#)
 - refresh, [846](#)
 - setPrompt, [847](#)
 - StringInputControl, [845](#)
 - takeInput, [847](#)
 - userPrompt, [847](#)
- StringInputControl.cc, [1349](#)
- StringInputControl.h, [1350](#)
- stringpad
 - LoadSave, [535](#)
- subject
 - MMCombo, [570](#)
 - RemoteProcess, [736](#)
 - SoundPlay, [817](#)
- subjs
 - MotionManager, [599](#)
 - SoundManager, [798](#)
- SuspendedSID
 - PowerSourceID, [124](#)
- swap
 - ListMemBuf, [506](#)
- sway
 - WalkMC::WalkParam, [964](#)
- sync
 - basic_netbuf, [206](#)
- SystemUtility.h, [1352](#)
- SystemUtility.h
 - DeleteLarge, [1353](#)
 - NewLarge, [1353](#)
- T
 - ListMemBuf, [499](#)
- t
 - HeadPointControllerBehavior, [446](#)
 - NonUniformHermiteSplineSegment, [653](#)
- t0
 - SplinePath, [824](#)
- t1
 - SplinePath, [824](#)
- tail_max
 - Kinematics.cc, [1205](#)
 - Kinematics.h, [1213](#)
- tail_min
 - Kinematics.cc, [1205](#)
 - Kinematics.h, [1213](#)
- TailCenterButOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [96](#)
- TailCenterButSID
 - ButtonSourceID, [30](#)
- TailCenterLEDMask
 - ERS220Info, [86](#)
 - ERS2xxInfo, [113](#)
- TailCenterLEDOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- TailLEDMask
 - ERS210Info, [56](#)
 - ERS220Info, [86](#)
 - ERS2xxInfo, [113](#)
- TailLeftButOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [96](#)

- TailLeftButSID
 - ButtonSourceID, [30](#)
- TailLeftLEDMask
 - ERS220Info, [86](#)
 - ERS2xxInfo, [113](#)
- TailLeftLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- TailOffset
 - ERS210Info, [56](#)
 - ERS2xxInfo, [113](#)
- TailRightButOffset
 - ERS220Info, [67](#)
 - ERS2xxInfo, [96](#)
- TailRightButSID
 - ButtonSourceID, [30](#)
- TailRightLEDMask
 - ERS220Info, [86](#)
 - ERS2xxInfo, [114](#)
- TailRightLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [67](#)
 - ERS2xxInfo, [97](#)
- tailwag_id
 - CameraBehavior, [258](#)
- TailWagMC, [849](#)
 - TailWagMC, [851](#)
- TailWagMC
 - ~TailWagMC, [851](#)
 - active, [854](#)
 - getActive, [851](#)
 - getMagnitude, [851](#)
 - getPeriod, [851](#)
 - getTilt, [851](#)
 - isAlive, [852](#)
 - isDirty, [852](#)
 - magnitude, [854](#)
 - pans, [854](#)
 - period, [854](#)
 - setActive, [852](#)
 - setMagnitude, [852](#)
 - setPeriod, [852](#)
 - setTilt, [853](#)
 - TailWagMC, [851](#)
 - tilt, [854](#)
 - unsetTilt, [853](#)
 - updateOutputs, [853](#)
- TailWagMC.h, [1354](#)
- takeInput
 - ControlBase, [304](#)
 - NullControl, [656](#)
 - SavePostureControl, [741](#)
 - StringInputControl, [847](#)
 - ValueEditControl, [884](#)
- takeLine
 - Controller, [318](#)
- takeSnapshot
 - EmergencyStopMC, [349](#)
 - PostureEngine, [688](#)
 - PostureMC, [696](#)
- target
 - BehaviorActivatorControl, [227](#)
 - ValueEditControl, [884](#)
 - ValueSetControl, [890](#)
- target_vel_xya
 - WalkMC, [956](#)
- term_width
 - HelpControl, [459](#)
- TermChargeSID
 - PowerSourceID, [124](#)
- TermDischargeSID
 - PowerSourceID, [124](#)
- Test.c, [1356](#)
 - landmarks, [1357](#)
 - main, [1357](#)
 - Particles, [1357](#)
- Test.t
 - CompareTrans, [267](#)
- textForward
 - Socket, [781](#)
- textForwardBuf
 - Socket, [781](#)
- textmsgEGID
 - EventBase, [362](#)
- TextMsgEvent, [855](#)
 - TextMsgEvent, [856](#)
- TextMsgEvent
 - _text, [858](#)
 - _token, [858](#)
 - getBinSize, [856](#)
 - getText, [857](#)

- getToken, [857](#)
- LoadBuffer, [857](#)
- SaveBuffer, [857](#)
- setText, [858](#)
- setToken, [858](#)
- TextMsgEvent, [856](#)
- TextMsgEvent.h, [1358](#)
- TextMsgEvent_ID
 - EventTranslator, [407](#)
- theLastOne
 - HeadPointControllerBehavior, [446](#)
 - WalkControllerBehavior, [943](#)
- theOne
 - EStopControllerBehavior, [356](#)
 - HeadPointControllerBehavior, [446](#)
 - WalkControllerBehavior, [943](#)
- theOneController
 - Controller, [321](#)
- ThermoOffset
 - ERS210Info, [42](#)
 - ERS220Info, [69](#)
 - ERS2xxInfo, [99](#)
- theta
 - _afsLastObservation, [142](#)
 - _afsPose, [146](#)
 - _afsRRP, [148](#)
- THING
 - VisionInterface, [135](#)
- thing
 - VisionInterface::ObjectInfo, [658](#)
- ThingSID
 - VisionEventNS, [131](#)
- THR_AHEAD
 - WorldModel2Behavior::Walk-Node, [1010](#)
- THR_TURN
 - WorldModel2Behavior::Walk-Node, [1010](#)
- thresh
 - Config::vision_config, [288](#)
- thresholdImage
 - Vision, [913](#)
- ThumbsupSID
 - VisionEventNS, [131](#)
- tilt
 - TailWagMC, [854](#)
- TiltOffset
 - ERS210Info, [42](#)
 - ERS220Info, [70](#)
 - ERS2xxInfo, [99](#)
- time
 - _FastSLAM_update, [153](#)
 - motionReshapeKludge.h, [1254](#)
 - WalkMC, [956](#)
- TimeET, [860](#)
- TimeET, [862](#)
- TimeET
 - Age, [863](#)
 - ms_per_sec, [865](#)
 - operator+, [863](#)
 - operator+=, [863](#)
 - operator-, [863](#)
 - operator-=, [863](#)
 - operator<, [863](#), [864](#)
 - operator<<, [865](#)
 - Set, [864](#)
 - TimeET, [862](#)
 - tv, [865](#)
 - tz, [865](#)
 - us_per_ms, [865](#)
 - us_per_sec, [866](#)
 - Value, [865](#)
- TimeET.cc, [1360](#)
- TimeET.h, [1361](#)
- TimeET.h
 - operator+, [1362](#)
 - operator-, [1362](#)
 - operator<<, [1362](#)
- timeoflastbtn
 - EmergencyStopMC, [351](#)
- timeoflastfreeze
 - EmergencyStopMC, [351](#)
- timeofthisbtn
 - EmergencyStopMC, [351](#)
- timeout
 - WalkToTargetMachine, [969](#)
- TimeOutTrans, [867](#)
- TimeOutTrans, [868](#)
- TimeOutTrans
 - d, [869](#)

- disable, [868](#)
 - enable, [868](#)
 - processEvent, [868](#)
 - resetTimer, [868](#)
 - TimeOutTrans, [868](#)
- TimeOutTrans.h, [1364](#)
- Timer
 - Profiler, [707](#)
 - Profiler::Timer, [715](#), [716](#)
- timer_it_t
 - EventRouter, [383](#)
- TIMER_SID_GPA
 - WorldModel2.cc, [1427](#)
- TIMER_SID_SRL
 - WorldModel2.cc, [1427](#)
- timerEGID
 - EventBase, [362](#)
- TimerEntry
 - EventRouter::TimerEntry, [401](#)
- timers
 - EventRouter, [393](#)
- timestamp
 - EventBase, [371](#)
- TimeStep
 - WalkMC, [956](#)
- timeval, [870](#)
 - tv_sec, [870](#)
 - tv_usec, [870](#)
- timezone, [871](#)
 - tz_dsttime, [871](#)
 - tz_minuteswest, [871](#)
- TIBluLEDMask
 - ERS210Info, [56](#)
 - ERS220Info, [86](#)
 - ERS2xxInfo, [114](#)
- TIBluLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- TIRedLEDMask
 - ERS210Info, [56](#)
 - ERS220Info, [87](#)
 - ERS2xxInfo, [114](#)
- TIRedLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
- ERS2xxInfo, [97](#)
- tmap
 - Vision, [917](#)
- to_int_type
 - char_traits, [260](#)
- TODEG
 - Util.h, [1374](#)
- toggle
 - BehaviorActivatorControl, [225](#)
 - BehaviorSwitchActivatorControl, [235](#)
 - BehaviorSwitchControl, [243](#)
 - BehaviorSwitchControlBase, [247](#)
- ToggleHeadLightBehavior, [872](#)
 - ToggleHeadLightBehavior, [873](#)
- ToggleHeadLightBehavior
 - DoStart, [873](#)
 - DoStop, [873](#)
 - getClassDescription, [873](#)
 - getName, [874](#)
 - light_id, [874](#)
 - ToggleHeadLightBehavior, [873](#)
- ToggleHeadLightBehavior.h, [1366](#)
- tolerance
 - PostureMC, [697](#)
- top
 - Controller, [318](#)
- TopBrLEDMask
 - ERS210Info, [56](#)
 - ERS220Info, [87](#)
 - ERS2xxInfo, [114](#)
- TopBrLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- TopLLEDMask
 - ERS210Info, [56](#)
 - ERS220Info, [87](#)
 - ERS2xxInfo, [114](#)
- TopLLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- TopRLEDMask
 - ERS210Info, [57](#)
 - ERS220Info, [87](#)

- ERS2xxInfo, [114](#)
- TopRLEDOffset
 - ERS210Info, [40](#)
 - ERS220Info, [68](#)
 - ERS2xxInfo, [97](#)
- totalInterval
 - Profiler::SectionInfo, [712](#)
- totalTime
 - Profiler::SectionInfo, [713](#)
- TPROffset_t
 - ERS210Info, [42](#)
 - ERS220Info, [70](#)
 - ERS2xxInfo, [99](#)
- tracking
 - WalkToTargetMachine, [969](#)
- traits_type
 - basic_iNetStream, [191](#)
 - basic_ioNetStream, [196](#)
 - basic_netbuf, [202](#)
 - basic_oNetStream, [209](#)
- Transition, [875](#)
 - ~Transition, [876](#)
 - activate, [876](#)
 - disable, [877](#)
 - dst, [877](#)
 - enable, [877](#)
 - operator=, [877](#)
 - src, [877](#)
 - Transition, [876](#)
- Transition.cc, [1368](#)
- Transition.h, [1369](#)
- transitionFrom
 - StateNode, [841](#)
- transitions
 - StateNode, [842](#)
- transitionTo
 - StateNode, [841](#)
- translateEvents
 - EventTranslator, [408](#)
- TransportType_t
 - SocketNS, [129](#)
- trapEvent
 - Controller, [318](#)
 - EventTranslator, [408](#)
 - EventTrapper, [411](#)
- trappers
 - EventRouter, [393](#)
- trav
 - _hm_cell, [155](#)
- travelTime
 - WalkMC, [956](#)
- triangulate
 - afsTriangulator.cc, [1044](#)
- trType
 - Socket, [781](#)
- try_lock
 - MutexLock, [646](#)
- trylock
 - MotionManager, [595](#)
- tst
 - CompareTrans, [269](#)
- TURN
 - World-
 - Model2Behavior::WalkNode, [1008](#)
- turn
 - MutexLock::door_t, [651](#)
- tv
 - TimeET, [865](#)
- tv_sec
 - timeval, [870](#)
- tv_usec
 - timeval, [870](#)
- twodigit
 - LedEngine, [480](#)
- tx
 - Socket, [781](#)
- type
 - _FastSLAM_update, [153](#)
 - MotionManagerMsg, [611](#)
 - MotionRequest, [613](#)
 - SoundManagerMsg, [809](#)
- typeID
 - EventBase, [371](#)
- TypeID_t
 - EventTranslator, [407](#)
- tz
 - TimeET, [865](#)
- tz_dsttime
 - timezone, [871](#)
- tz_minuteswest
 - timezone, [871](#)

- uflow
 - basic_netbuf, 206
- uint
 - MotionManager.cc, 1245
- underflow
 - basic_netbuf, 206
- unknown
 - MotionManagerMsg, 610
 - SoundManagerMsg, 807
- unknownEGID
 - EventBase, 361
- UNLESS
 - WorldModel2.cc, 1428
- unlock
 - MutexLock, 647
- unset
 - OutputCmd, 663
 - OutputPID, 670
- unsetTilt
 - TailWagMC, 853
- unused
 - HeadPointerMC, 456
- updateChannels
 - SoundManager, 796
- UpdatedSID
 - PowerSourceID, 125
 - SensorSourceID, 127
- updateLEDFrames
 - LedEngine, 486
- updateLEDs
 - LedEngine, 486
- updateOutputs
 - DynamicMotionSequence, 342
 - EmergencyStopMC, 349
 - HeadPointerMC, 454
 - LedMC, 494
 - MotionCommand, 578
 - MotionSequence, 629
 - MotionSequenceMC, 640
 - PIDMC, 676
 - PostureMC, 696
 - RemoteControllerMC, 730
 - TailWagMC, 853
 - WalkMC, 954
- updatePIDs
 - MotionManager, 596
- updateWorldState
 - MotionManager, 596
- us_per_ms
 - TimeET, 865
- us_per_sec
 - TimeET, 866
- use_VT100
 - Config::main_config, 280
- userPrompt
 - StringInputControl, 847
- using_buf_in
 - basic_netbuf, 207
- using_buf_out
 - basic_netbuf, 207
- Util.h, 1370
 - angle_wrap, 1372
 - atan2a, 1372
 - atan2b, 1372
 - avg_angle, 1372
 - bound, 1372
 - clearbits, 1372
 - fmodt, 1372
 - gaussian_constant, 1374
 - gaussian_prob, 1373
 - gaussian_with_min, 1373
 - list_length, 1373
 - max3, 1373
 - mcopy, 1373
 - min3, 1373
 - mset, 1373
 - mzero, 1373
 - norm_angle, 1374
 - saw, 1374
 - setbits, 1374
 - sign, 1374
 - sort, 1374
 - TODEG, 1374
- V2COMP
 - gvector.h, 1179
- V3COMP
 - gvector.h, 1179
- val
 - Aibo3DControllerBehavior, 163
 - CompareTrans, 269
- validInput

- ControlBase, [304](#)
- Value
 - TimeET, [865](#)
- value
 - LedEngine::LEDInfo, [490](#)
 - OutputCmd, [664](#)
- ValueEditControl, [878](#)
 - ValueEditControl, [880](#), [881](#)
- ValueEditControl
 - ~ValueEditControl, [881](#)
 - activate, [881](#)
 - addCopy, [881](#)
 - copies, [884](#)
 - cur, [884](#)
 - doNextItem, [882](#)
 - doPrevItem, [882](#)
 - doSelect, [882](#)
 - getCopies, [882](#)
 - getName, [882](#)
 - getTarget, [883](#)
 - operator=, [883](#)
 - pause, [883](#)
 - processEvent, [883](#)
 - setTarget, [884](#)
 - takeInput, [884](#)
 - target, [884](#)
 - ValueEditControl, [880](#), [881](#)
- ValueEditControl.h, [1375](#)
- ValueSetControl, [886](#)
 - ValueSetControl, [887](#), [888](#)
- ValueSetControl
 - ~ValueSetControl, [888](#)
 - activate, [888](#)
 - def, [890](#)
 - getDefault, [889](#)
 - getTarget, [889](#)
 - operator=, [889](#)
 - setDefault, [889](#)
 - setTarget, [890](#)
 - target, [890](#)
 - ValueSetControl, [887](#), [888](#)
- ValueSetControl.h, [1377](#)
- variance
 - _afsLandmarkLoc, [140](#)
- vector2d
 - Geometry.h, [1172](#)
- GVector::vector2d, [892](#)
- VECTOR2D.BINARY_OPERATOR
 - gvector.h, [1179](#)
- VECTOR2D.EQUAL.BINARY_OPERATOR
 - GVector, [119](#)
 - gvector.h, [1179](#)
- VECTOR2D.EQUAL.SCALAR_OPERATOR
 - gvector.h, [1179](#)
- VECTOR2D.LOGIC_OPERATOR
 - GVector, [120](#)
 - gvector.h, [1180](#)
- VECTOR2D.SCALAR_OPERATOR
 - gvector.h, [1180](#)
- vector2f
 - Geometry.h, [1172](#)
- vector3d
 - Geometry.h, [1172](#)
 - GVector::vector3d, [898](#)
- VECTOR3D.BINARY_OPERATOR
 - gvector.h, [1180](#)
- VECTOR3D.EQUAL.BINARY_OPERATOR
 - GVector, [120](#)
 - gvector.h, [1180](#)
- VECTOR3D.EQUAL.SCALAR_OPERATOR
 - gvector.h, [1181](#)
- VECTOR3D.LOGIC_OPERATOR
 - GVector, [120](#)
 - gvector.h, [1181](#)
- VECTOR3D.SCALAR_OPERATOR
 - gvector.h, [1181](#)
- vector3f
 - Geometry.h, [1172](#)
- vel_xya
 - WalkMC, [956](#)
- verbose_level
 - Config::main_config, [280](#)
- verbosity
 - EventLogger, [377](#)
 - Socket, [781](#)
- VertFOV
 - VisionInterface, [135](#)
- vevent_spec

- Vision, 917
- VibrationSID
 - PowerSourceID, 124
- vis_markers
 - Vision, 917
- Vision, 904
 - ~Vision, 906
 - addToHistHorizStrip, 907
 - addToHistVertStrip, 907
 - body_angle, 913
 - body_height, 913
 - calcEdgeMask, 907
 - calcTotalArea, 913
 - camera_dir, 913
 - camera_loc, 914
 - camera_right, 914
 - camera_up, 914
 - close, 907
 - cmap, 914
 - color, 914
 - createEvent, 907
 - cur_tmap, 914
 - disableEvents, 908
 - enableEvents, 908
 - findBall, 908
 - findGesture, 908
 - findHand, 908
 - findMarkers, 909
 - findSpan, 909
 - findThing, 909
 - frame_count, 914
 - frameTimestamp, 914
 - generateEvent, 909
 - get_camera_dir, 909
 - get_camera_loc, 909
 - getColor, 910
 - getColorUnsafe, 910
 - getHeight, 910
 - getNearColor, 910
 - getPixelDirection, 910
 - getWidth, 910
 - head_angles, 914
 - height, 914
 - identifyMarker, 910
 - img, 915
 - initialize, 911
 - initializeEventSpecs, 911
 - isAdjacent, 911
 - isIn, 911
 - lin_distort_coeff, 915
 - markers, 915
 - max_height, 915
 - max_regions, 915
 - max_runs, 915
 - max_width, 915
 - num_colors, 915
 - num_regions, 915
 - num_runs, 915
 - num_tmaps, 916
 - obj_info, 916
 - operator=, 911
 - outCountAvgColor, 916
 - outCountColorArea, 916
 - outCountRaw, 916
 - outCountRLE, 916
 - processFrame, 911
 - reg, 916
 - rmap, 916
 - rmap2, 916
 - runHighLevelVision, 912
 - runLowLevelVision, 912
 - saveThresholdImage, 912
 - sendColorArea, 912
 - sendRawImage, 912
 - sendRLEImage, 912
 - setCameraParam, 912
 - setNoiseThreshold, 912
 - setThreshold, 912
 - sq_distort_coeff, 916
 - thresholdImage, 913
 - tmap, 917
 - vevent_spec, 917
 - vis_markers, 917
 - Vision, 906
 - VisionSerializer, 913
 - vobj_info, 917
 - vser, 917
 - width, 917
 - yindex, 917
- vision
 - Config, 273
 - Vision.cc, 1382

- Vision.h, 1387
- Vision.cc, 1379
 - pct_from_mean, 1381
 - vision, 1382
 - WritePPM, 1381
- Vision.h, 1383
 - bits_u, 1386
 - bits_v, 1386
 - bits_y, 1386
 - cmap_t, 1386
 - color_class_state, 1386
 - FocalDist, 1387
 - FocalDistV, 1387
 - image, 1386
 - MAX_TMAPS, 1385
 - MIN_EXP_REGION_SIZE, 1385
 - MIN_EXP_RUN_LENGTH, 1385
 - pixel, 1386
 - region, 1386
 - RobotArea, 1387
 - run, 1386
 - vision, 1387
 - YPixelSize, 1387
- vision_config
 - Config::vision_config, 287
- Visiondefines.h, 1388
 - COLOR_BACKGROUND, 1388
 - COLOR_BGREEN, 1388
 - COLOR_BLACK, 1388
 - COLOR_BLUE, 1389
 - COLOR_GRAY, 1389
 - COLOR_GREEN, 1389
 - COLOR_ORANGE, 1389
 - COLOR_PINK, 1389
 - COLOR_PURPLE, 1389
 - COLOR_RED, 1389
 - COLOR_SKIN, 1389
 - COLOR_YELLOW, 1389
 - MAX_COLORS, 1389
 - NUM_VEVENTS, 1390
- visionEGID
 - EventBase, 361
- VisionEvent, 918
 - VisionEvent, 920
- VisionEvent
 - _cenX, 923
 - _cenY, 923
 - _distance, 923
 - _property, 923
 - getBinSize, 920
 - getCenterX, 920
 - getCenterY, 921
 - getDistance, 921
 - getProperty, 921
 - LoadBuffer, 921
 - SaveBuffer, 922
 - setCenterX, 922
 - setCenterY, 922
 - setDistance, 922
 - setProperty, 922
 - VisionEvent, 920
- VisionEvent.h, 1391
- VisionEvent_ID
 - EventTranslator, 407
- VisionEventNS, 131
 - HandSID, 131
 - MarkersSID, 131
 - PinkBallSID, 131
 - RedBallSID, 131
 - ThingSID, 131
 - ThumbsupSID, 131
- VisionEventNS
 - VisionSourceID_t, 131
- VisionEventSpec, 924
- VisionEventSpec
 - confidence, 924
 - count, 924
 - cx, 924
 - cy, 924
 - filter, 924
 - listeners, 924
 - present, 924
- VisionInterface, 132
- VisionInterface
 - HAND, 133
 - HorzFOV, 133
 - MARKER_GOG, 133
 - MARKER_GOP, 133
 - MARKER_GPG, 133
 - MARKER_GPO, 133
 - MARKER_OGO, 134
 - MARKER_OGP, 134

- MARKER_OPG, [134](#)
- MARKER_OPO, [134](#)
- MARKER_PGO, [134](#)
- MARKER_PGP, [134](#)
- MARKER_POG, [134](#)
- MARKER_POP, [134](#)
- NUM_MARKERS, [134](#)
- NUM_VISION_OBJECTS, [134](#)
- OFF_EDGE_BOTTOM, [135](#)
- OFF_EDGE_LEFT, [135](#)
- OFF_EDGE_RIGHT, [135](#)
- OFF_EDGE_TOP, [135](#)
- PBALL, [135](#)
- RBALL, [135](#)
- SendRawImage, [133](#)
- SendRLEImage, [133](#)
- SetThreshold, [133](#)
- THING, [135](#)
- VertFOV, [135](#)
- WriteThresholdImage, [133](#)
- VisionInterface.h, [1393](#)
- VisionInterface::ObjectInfo, [658](#)
- VisionInterface::ObjectInfo
 - hand, [658](#)
 - marker, [658](#)
 - pball, [658](#)
 - rball, [658](#)
 - thing, [658](#)
- VisionInterface::VObject, [933](#)
- VisionInterface::VObject
 - confidence, [933](#)
 - distance, [933](#)
 - edge, [933](#)
 - left, [933](#)
 - loc, [933](#)
 - right, [933](#)
- VisionObjectInfo, [926](#)
- VisionObjectInfo
 - reg, [926](#)
 - reg2, [926](#)
- VisionSerializer, [927](#)
 - Vision, [913](#)
 - VisionSerializer, [928](#)
- VisionSerializer
 - encodeVisionRaw, [928](#)
 - encodeVisionRLE, [928](#)
 - encodeVisionRun, [929](#)
 - operator=, [929](#)
 - serialize, [929](#)
 - VisionSerializer, [928](#)
 - visRaw, [929](#)
 - visRLE, [929](#)
- VisionSerializer.cc, [1395](#)
- VisionSerializer.h, [1397](#)
- VisionSourceID.t
 - VisionEventNS, [131](#)
- visRaw
 - VisionSerializer, [929](#)
- visRLE
 - VisionSerializer, [929](#)
- VisualTargetCloseTrans, [930](#)
 - VisualTargetCloseTrans, [931](#)
- VisualTargetCloseTrans
 - disable, [931](#)
 - enable, [931](#)
 - processEvent, [931](#)
 - sid, [932](#)
 - VisualTargetCloseTrans, [931](#)
- VisualTargetCloseTrans.h, [1399](#)
- vobj_info
 - Vision, [917](#)
- vprintf
 - Socket, [777](#)
- vser
 - Vision, [917](#)
- w
 - karmedbanditExp3, [472](#)
- waiting
 - AutoGetupBehavior, [174](#)
- WaitNode
 - BanditMachine::WaitNode, [187](#)
 - WorldModel2Behavior::WaitNode, [1005](#)
- wakeup
 - SoundManagerMsg, [807](#)
- walk_id
 - LoadWalkControl, [538](#)
 - SaveWalkControl, [743](#)
- walkControl_port
 - Config::main_config, [280](#)
- WalkControllerBehavior, [935](#)

- WalkControllerBehavior, 938
- WalkControllerBehavior
 - ~WalkControllerBehavior, 938
 - CMD.fwd, 940
 - CMD.opt0, 940
 - CMD.opt1, 940
 - CMD.opt2, 940
 - CMD.opt3, 941
 - CMD.opt4, 941
 - CMD.opt5, 941
 - CMD.opt6, 941
 - CMD.opt7, 941
 - CMD.opt8, 941
 - CMD.opt9, 942
 - CMD.roto, 942
 - CMD.side, 942
 - cmdsock, 942
 - da, 942
 - DoStart, 938
 - DoStop, 938
 - dx, 942
 - dy, 942
 - getClassDescription, 939
 - getName, 939
 - mechacmd_callback, 939
 - operator=, 939
 - processEvent, 939
 - runCommand, 940
 - theLastOne, 943
 - theOne, 943
 - WalkControllerBehavior, 938
 - walker_id, 943
- WalkControllerBehavior.cc, 1401
- WalkControllerBehavior.h, 1402
- walker_id
 - ChaseBallBehavior, 264
 - DriveMeBehavior, 331
 - FollowHeadBehavior, 428
 - SimpleChaseBallBehavior, 765
 - WalkControllerBehavior, 943
 - WalkToTargetMachine, 969
 - WorldModel2Behavior::Walk-
Node, 1010
- WalkMC, 944
 - WalkMC, 949
- WalkMC
 - angle_delta, 954
 - body_angle, 954
 - body_loc, 954
 - cmds, 955
 - DoStart, 949
 - DoStop, 949
 - getAngle, 949
 - getCurVelocity, 950
 - getHeight, 950
 - getHop, 950
 - getPaused, 950
 - getPeriod, 950
 - getSway, 950
 - getTargetVelocity, 951
 - getTravelTime, 951
 - init, 951
 - isAlive, 951
 - isDirty, 951
 - isPaused, 955
 - legpos, 955
 - legw, 955
 - load, 952
 - max_accel_xya, 955
 - MAX_DA, 955
 - MAX_DX, 955
 - MAX_DY, 956
 - pos_delta, 956
 - resetLegPos, 952
 - save, 952
 - setAngle, 952
 - setHeight, 952
 - setHop, 953
 - setPaused, 953
 - setPeriod, 953
 - setSway, 953
 - setTargetVelocity, 953
 - spline, 948
 - splinepath, 948
 - target_vel_xya, 956
 - time, 956
 - TimeStep, 956
 - travelTime, 956
 - updateOutputs, 954
 - vel_xya, 956
 - WalkMC, 949
 - wp, 957

- WalkMC.cc, [1404](#)
- WalkMC.cc
 - BOUND_MOTION, [1406](#)
 - checksum, [1406](#)
 - read_file, [1406](#)
 - save_file, [1406](#)
- WalkMC.h, [1407](#)
- WalkMC::LegParam, [958](#)
- WalkMC::LegParam
 - down_time, [959](#)
 - down_vel, [959](#)
 - LegParam, [958](#)
 - lift_time, [959](#)
 - lift_vel, [959](#)
 - neutral, [959](#)
- WalkMC::LegWalkState, [960](#)
- WalkMC::LegWalkState
 - air, [960](#)
 - airpath, [960](#)
 - LegWalkState, [960](#)
- WalkMC::WalkParam, [962](#)
- WalkMC::WalkParam
 - body_angle, [963](#)
 - body_height, [963](#)
 - hop, [963](#)
 - leg, [963](#)
 - period, [963](#)
 - reserved, [963](#)
 - sway, [964](#)
 - WalkParam, [963](#)
- WalkMotionModel
 - WalkMotionModel.cc, [1410](#)
 - WalkMotionModel.h, [1411](#)
- WalkMotionModel.cc, [1410](#)
- WalkMotionModel.cc
 - WalkMotionModel, [1410](#)
- WalkMotionModel.h, [1411](#)
- WalkMotionModel.h
 - WalkMotionModel, [1411](#)
- WalkNode
 - WorldModel2Behavior::Walk-
Node, [1008](#)
- WalkParam
 - WalkMC::WalkParam, [963](#)
- walkstate
 - WorldModel2Behavior::Walk-
Node, [1010](#)
- WalkToTargetMachine, [965](#)
 - WalkToTargetMachine, [967](#)
- WalkToTargetMachine
 - close, [968](#)
 - DoStart, [967](#)
 - DoStop, [967](#)
 - getClassDescription, [967](#)
 - headpointer_id, [969](#)
 - lost, [969](#)
 - operator=, [968](#)
 - processEvent, [968](#)
 - setup, [968](#)
 - timeout, [969](#)
 - tracking, [969](#)
 - walker_id, [969](#)
 - WalkToTargetMachine, [967](#)
- WalkToTargetMachine.cc, [1412](#)
- WalkToTargetMachine.cc
 - DtoR, [1413](#)
- WalkToTargetMachine.h, [1414](#)
- WAV, [970](#)
 - ~WAV, [971](#)
 - CopyTo, [971](#)
 - dataCurrent, [972](#)
 - dataEnd, [972](#)
 - dataStart, [972](#)
 - FMTSIZE_WITHOUT_-
EXTINFO, [973](#)
 - get_longword, [971](#)
 - get_word, [971](#)
 - GetBitsPerSample, [971](#)
 - GetDataEnd, [971](#)
 - GetDataStart, [971](#)
 - GetSamplingRate, [972](#)
 - GetSoundUnitSize, [972](#)
 - MONO16K16B_UNIT_SIZE,
[973](#)
 - MONO8K8B_UNIT_SIZE, [973](#)
 - Rewind, [972](#)
 - Set, [972](#)
 - soundInfo, [973](#)
 - soundUnitSize, [973](#)
 - WAV, [970](#)
- WAV.cc, [1416](#)

- WAV.h, [1418](#)
 - WAV_BITSPERSAMPLE_-NOT_SUPPORTED, [1419](#)
 - WAV_CHANNEL_NOT_SUPPORTED, [1419](#)
 - WAV_FAIL, [1419](#)
 - WAV_FORMAT_NOT_SUPPORTED, [1419](#)
 - WAV_NOT_RIFF, [1419](#)
 - WAV_NOT_WAV, [1419](#)
 - WAV_SAMPLINGRATE_NOT_SUPPORTED, [1419](#)
 - WAV_SIZE_NOT_ENOUGH, [1419](#)
 - WAV_SUCCESS, [1419](#)
 - WAVError, [1419](#)
- WAV_BITSPERSAMPLE_NOT_SUPPORTED
 - WAV.h, [1419](#)
- WAV_CHANNEL_NOT_SUPPORTED
 - WAV.h, [1419](#)
- WAV_FAIL
 - WAV.h, [1419](#)
- WAV_FORMAT_NOT_SUPPORTED
 - WAV.h, [1419](#)
- WAV_NOT_RIFF
 - WAV.h, [1419](#)
- WAV_NOT_WAV
 - WAV.h, [1419](#)
- WAV_SAMPLINGRATE_NOT_SUPPORTED
 - WAV.h, [1419](#)
- WAV_SIZE_NOT_ENOUGH
 - WAV.h, [1419](#)
- WAV_SUCCESS
 - WAV.h, [1419](#)
- WAVError
 - WAV.h, [1419](#)
- weight
 - _afsParticle, [144](#)
 - OutputCmd, [664](#)
 - OutputPID, [670](#)
- Weights
 - afsMain.cc, [1031](#)
- white_balance
 - Config::vision_config, [288](#)
- width
 - Vision, [917](#)
- Wireless, [974](#)
 - ~Wireless, [977](#)
 - BindCont, [977](#)
 - blockingSend, [977](#)
 - close, [978](#)
 - CloseCont, [978](#)
 - connect, [978](#), [979](#)
 - ConnectCont, [979](#)
 - hasData, [979](#)
 - ipstackRef, [983](#)
 - isConnected, [979](#)
 - isReady, [980](#)
 - listen, [980](#)
 - ListenCont, [981](#)
 - myOID, [983](#)
 - operator=, [981](#)
 - receive, [981](#)
 - ReceiveCont, [981](#)
 - send, [981](#)
 - SendCont, [982](#)
 - setReceiver, [982](#)
 - sock_num, [983](#)
 - Socket, [778](#)
 - socket, [982](#), [983](#)
 - sockets, [984](#)
 - Wireless, [977](#)
 - WIRELESS_DEF_RECV_SIZE, [984](#)
 - WIRELESS_DEF_SEND_SIZE, [984](#)
 - WIRELESS_MAX_SOCKETS, [984](#)
- wireless
 - Config, [273](#)
 - Wireless.cc, [1422](#)
 - Wireless.h, [1425](#)
- Wireless.cc, [1420](#)
 - wireless, [1422](#)
- Wireless.h, [1423](#)
 - wireless, [1425](#)
- wireless_config
 - Config::wireless_config, [289](#)
- WIRELESS_DEF_RECV_SIZE

- Wireless, [984](#)
- WIRELESS_DEF_SEND_SIZE
 - Wireless, [984](#)
- WIRELESS_MAX_SOCKETS
 - Wireless, [984](#)
- WM2
 - DumbWM2Behavior, [334](#)
 - WorldModel2Behavior, [999](#)
- WM2Kludge, [136](#)
 - IgnoreGreenItems, [136](#)
 - IgnoreZLessThanZero, [136](#)
 - LazyFastSLAM, [136](#)
- WM2ref
 - WorldModel2Behavior::GawkNode, [1002](#)
- WorldModel2, [985](#)
 - AGM, [157](#)
 - ALM, [170](#)
 - WorldModel2, [988](#), [989](#)
- WorldModel2
 - ~WorldModel2, [988](#)
 - da, [993](#)
 - disableGPA, [989](#)
 - disableIR, [989](#)
 - disableKludge, [989](#)
 - dx, [993](#)
 - dy, [993](#)
 - enabledGPA, [993](#)
 - enabledIR, [994](#)
 - enableGPA, [989](#)
 - enableIR, [990](#)
 - enableKludge, [990](#)
 - getRequests, [990](#)
 - GPAdelay, [994](#)
 - invadeDMData, [991](#)
 - invadeHMData, [991](#)
 - kludges, [994](#)
 - locked, [994](#)
 - moving, [994](#)
 - movingSince, [994](#)
 - operator=, [991](#)
 - pickDMData, [991](#)
 - pickHMData, [991](#)
 - processEvent, [991](#)
 - processGround, [992](#)
 - processLocomotion, [992](#)
 - processSensors, [992](#)
 - resetWorldModel, [992](#)
 - serialize, [993](#)
 - sockDM, [995](#)
 - sockHM, [995](#)
 - SRLdelay, [995](#)
 - WorldModel2, [988](#), [989](#)
- worldmodel2
 - Config, [273](#)
- WorldModel2.cc, [1426](#)
- WorldModel2.cc
 - aiboIsErect, [1428](#)
 - aiboIsLevelHeaded, [1428](#)
 - aiboStaresDeadAhead, [1428](#)
 - TIMER_SID_GPA, [1427](#)
 - TIMER_SID_SRL, [1427](#)
 - UNLESS, [1428](#)
- WorldModel2.h, [1429](#)
- WorldModel2.h
 - aiboIsErect, [1431](#)
 - aiboIsLevelHeaded, [1431](#)
 - aiboStaresDeadAhead, [1431](#)
 - FastSLAM_update, [1430](#)
 - FSUdeque, [1430](#)
 - MRvector, [1430](#)
- worldmodel2_config
 - Config::worldmodel2_config, [290](#)
- WorldModel2Behavior, [996](#)
 - WorldModel2Behavior, [997](#), [998](#)
- WorldModel2Behavior
 - ~WorldModel2Behavior, [997](#)
 - DoStart, [998](#)
 - DoStop, [998](#)
 - getClassDescription, [998](#)
 - operator=, [998](#)
 - setup, [998](#)
 - standUp, [999](#)
 - standUp_id, [999](#)
 - start, [999](#)
 - WM2, [999](#)
 - WorldModel2Behavior, [997](#), [998](#)
- WorldModel2Behavior.cc, [1432](#)
- WorldModel2Behavior.h, [1434](#)
- WorldModel2Behavior::GawkNode, [1000](#)
- WorldModel2Behavior::GawkNode

- DoStart, [1001](#)
- DoStop, [1001](#)
- GawkNode, [1001](#)
- head_id, [1002](#)
- operator=, [1002](#)
- processEvent, [1002](#)
- stand_id, [1002](#)
- WM2ref, [1002](#)
- WorldModel2Behavior::WaitNode, [1004](#)
- WorldModel2Behavior::WaitNode
 - DoStart, [1005](#)
 - DoStop, [1005](#)
 - head_id, [1005](#)
 - stand_id, [1005](#)
 - WaitNode, [1005](#)
- WorldModel2Behavior::WalkNode, [1007](#)
 - AHEAD, [1008](#)
 - TURN, [1008](#)
- WorldModel2Behavior::WalkNode~WalkNode, [1008](#)
 - DoStart, [1009](#)
 - DoStop, [1009](#)
 - head_id, [1010](#)
 - processEvent, [1009](#)
 - stand_id, [1010](#)
 - THR_AHEAD, [1010](#)
 - THR_TURN, [1010](#)
 - walker_id, [1010](#)
 - WalkNode, [1008](#)
 - walkstate, [1010](#)
- worldModelEGID
 - EventBase, [361](#)
- WorldState, [1011](#)
 - WorldState, [1014](#)
- WorldState
 - alwaysGenerateStatus, [1016](#)
 - batteryStatus, [1016](#)
 - button_times, [1016](#)
 - buttons, [1016](#)
 - calcDers, [1014](#)
 - chkEvent, [1014](#)
 - chkPowerEvent, [1014](#)
 - curtime, [1016](#)
 - ERS210Mask, [1016](#)
 - ERS220Mask, [1017](#)
 - g, [1017](#)
 - IROORDist, [1017](#)
 - lastSensorUpdateTime, [1017](#)
 - mainProfile, [1017](#)
 - motionProfile, [1017](#)
 - operator=, [1015](#)
 - outputs, [1017](#)
 - pidduties, [1018](#)
 - pids, [1018](#)
 - powerFlags, [1018](#)
 - read, [1015](#)
 - robotDesign, [1018](#)
 - robotStatus, [1018](#)
 - sensors, [1018](#)
 - WorldState, [1014](#)
 - wsser, [1019](#)
- WorldState.cc, [1436](#)
- WorldState.cc
 - GETB, [1438](#)
 - GETD, [1438](#)
 - GETDUTY, [1438](#)
 - GETSIG, [1438](#)
 - state, [1438](#)
- WorldState.h, [1439](#)
- WorldState.h
 - state, [1442](#)
- worldStateMemRgn
 - MMCombo, [570](#)
- WorldStateSerializer, [1020](#)
 - WorldStateSerializer, [1020](#)
- WorldStateSerializer
 - operator=, [1021](#)
 - serialize, [1021](#)
 - WorldStateSerializer, [1020](#)
 - wsJoints, [1021](#)
 - wsPIDs, [1021](#)
- WorldStateSerializer.cc, [1443](#)
- WorldStateSerializer.h, [1444](#)
- wp
 - WalkMC, [957](#)
- write
 - Socket, [777](#), [778](#)
- writeData
 - Socket, [781](#)
- WritePPM

- Vision.cc, [1381](#)
- writeSize
 - Socket, [782](#)
- WriteThresholdImage
 - VisionInterface, [133](#)
- wsJoints
 - WorldStateSerializer, [1021](#)
- wsjoints_port
 - Config::main_config, [280](#)
- wsPIDs
 - WorldStateSerializer, [1021](#)
- wspids_port
 - Config::main_config, [281](#)
- wsser
 - WorldState, [1019](#)
- x
 - _afsLandmarkLoc, [140](#)
 - _afsLastObservation, [142](#)
 - _afsPose, [146](#)
 - _afsXY, [149](#)
 - GVector::vector2d, [895](#)
 - GVector::vector3d, [902](#)
 - LocomotionEvent, [544](#)
 - MotionRequest, [613](#)
- x0
 - SplinePath, [824](#)
- x1
 - SplinePath, [824](#)
- xy
 - _afsLandmarkLoc, [140](#)
 - MotionRequest, [613](#)
- xy2gm_index
 - almUtility.cc, [1061](#)
 - almUtility.h, [1064](#)
- xy2hm_index
 - almUtility.cc, [1061](#)
 - almUtility.h, [1064](#)
- xyz2neck_range
 - almUtility.cc, [1061](#)
 - almUtility.h, [1064](#)
- y
 - _afsLandmarkLoc, [141](#)
 - _afsLastObservation, [143](#)
 - _afsPose, [146](#)
 - _afsXY, [149](#)
 - GVector::vector2d, [895](#)
 - GVector::vector3d, [902](#)
 - LocomotionEvent, [544](#)
 - MotionRequest, [613](#)
- yindex
 - Vision, [917](#)
- YPixelSize
 - Vision.h, [1387](#)
- z
 - GVector::vector3d, [903](#)