ProCK-KI

0.1

Erzeugt von Doxygen 1.8.0

Sam Sep 22 2012 17:00:23

Inhaltsverzeichnis

1 Ausstehende Aufgaben **1**

2 Klassen-Verzeichnis **3**

2.1 Klassenhierarchie 3

3 Klassen-Verzeichnis **5**

3.1 Auflistung der Klassen 5

4 Datei-Verzeichnis **7**

4.1 Auflistung der Dateien 7

5 Klassen-Dokumentation **9**

5.1 AxisAgent Klassenreferenz 9

5.1.1 Ausführliche Beschreibung 11

5.1.2 Beschreibung der Konstruktoren und Destruktoren 11

5.1.2.1 AxisAgent 11

5.1.2.2 ∼AxisAgent 11

5.1.2.3 AxisAgent 11

5.1.2.4 ∼AxisAgent 12

5.1.3 Dokumentation der Elementfunktionen 12

5.1.3.1 advanceShoot 12

5.1.3.2 advanceShoot 12

5.1.3.3 getRunAgentInstance 12

5.1.3.4 getRunAgentInstance 12

5.1.3.5 runAgent 13

5.1.3.6 runAgent 13

5.1.3.7 shoot 14

5.1.3.8 shoot 14

5.1.3.9 start 15

5.1.3.10 start 15

5.1.3.11 stop 16

5.1.3.12 stop 16

5.1.4 Dokumentation der Datenelemente 16

5.1.4.1 afTargetPos . . . 16
5.1.4.2 axis . . . 16
5.1.4.3 firstFunc . . . 16
5.1.4.4 m_vecRotTargetPos . . . 17
5.1.4.5 mutexHandle . . . 17
5.1.4.6 myStopped . . . 17
5.1.4.7 myThreadHandle . . . 17
5.1.4.8 pConf . . . 17
5.1.4.9 pfComputerTargetPos . . . 18
5.1.4.10 pKB . . . 18
5.2 Ball Klassenreferenz . . . 18
5.2.1 Ausführliche Beschreibung . . . 21
5.2.2 Beschreibung der Konstruktoren und Destruktoren . . . 21
5.2.2.1 Ball . . . 21
5.2.2.2 ∼Ball . . . 22
5.2.3 Dokumentation der Elementfunktionen . . . 22
5.2.3.1 getPos . . . 22
5.2.3.2 getPosAndVelo . . . 22
5.2.3.3 getVelo . . . 23
5.2.3.4 setBallPos . . . 24
5.2.3.5 start . . . 24
5.2.4 Dokumentation der Datenelemente . . . 25
5.2.4.1 aBallPos . . . 25
5.2.4.2 client . . . 25
5.2.4.3 mutexHandle . . . 25
5.2.4.4 pConf . . . 25
5.2.4.5 pKB . . . 26
5.2.4.6 usBallPosIterator . . . 26
5.3 BallClient Klassenreferenz . . . 26
5.3.1 Ausführliche Beschreibung . . . 29
5.3.2 Beschreibung der Konstruktoren und Destruktoren . . . 29
5.3.2.1 BallClient . . . 29
5.3.2.2 ∼BallClient . . . 30
5.3.3 Dokumentation der Elementfunktionen . . . 30
5.3.3.1 requestBallPos . . . 30
5.3.3.2 start . . . 31
5.3.3.3 subscribeBallPos . . . 31
5.3.3.4 update . . . 32
5.3.3.5 update . . . 33
5.3.3.6 update . . . 33

5.3.4 Dokumentation der Datenelemente . . . 34
5.3.4.1 myRecvData . . . 34
5.3.4.2 myServerAddr . . . 34
5.3.4.3 mySocket . . . 34
5.3.4.4 myUser . . . 34
5.4 BallClientInterface Klassenreferenz . . . 34
5.4.1 Ausführliche Beschreibung . . . 35
5.4.2 Beschreibung der Konstruktoren und Destruktoren . . . 35
5.4.2.1 ~BallClientInterface . . . 35
5.4.3 Dokumentation der Elementfunktionen . . . 36
5.4.3.1 requestBallPos . . . 36
5.4.3.2 start . . . 36
5.4.3.3 subscribeBallPos . . . 36
5.5 BallClientUserInterface Schnittstellenreferenz . . . 37
5.5.1 Ausführliche Beschreibung . . . 38
5.5.2 Beschreibung der Konstruktoren und Destruktoren . . . 38
5.5.2.1 ~BallClientUserInterface . . . 38
5.5.3 Dokumentation der Elementfunktionen . . . 38
5.5.3.1 setBallPos . . . 38
5.5.3.2 start . . . 38
5.6 ballCoordinate_t Strukturreferenz . . . 39
5.6.1 Ausführliche Beschreibung . . . 39
5.6.2 Dokumentation der Datenelemente . . . 39
5.6.2.1 bCoordinateOk . . . 39
5.6.2.2 timestamp . . . 39
5.6.2.3 ulFrameCnt . . . 39
5.6.2.4 x . . . 40
5.6.2.5 y . . . 40
5.7 ballPacket Strukturreferenz . . . 40
5.7.1 Ausführliche Beschreibung . . . 41
5.7.2 Dokumentation der Datenelemente . . . 41
5.7.2.1 ballPos . . . 41
5.7.2.2 id . . . 41
5.8 ubf16_request::bitfeld_16 Strukturreferenz . . . 41
5.8.1 Ausführliche Beschreibung . . . 41
5.8.2 Dokumentation der Datenelemente . . . 42
5.8.2.1 ballSqueeze_05 . . . 42
5.8.2.2 bothAxisPos_02 . . . 42
5.8.2.3 computerAxisPos_01 . . . 42
5.8.2.4 computerAxisRamp_03 . . . 42

5.8.2.5 computerAxisStackmode_04 . . . 42
5.8.2.6 humanAxisPos_00 . . . 42
5.8.2.7 reserved_08 . . . 42
5.8.2.8 reserved_09 . . . 42
5.8.2.9 reserved_10 . . . 42
5.8.2.10 reserved_11 . . . 42
5.8.2.11 reserved_12 . . . 42
5.8.2.12 reserved_13 . . . 43
5.8.2.13 reserved_14 . . . 43
5.8.2.14 reserved_15 . . . 43
5.8.2.15 safety_07 . . . 43
5.8.2.16 score_06 . . . 43
5.9 bool_1 Strukturreferenz . . . 43
5.9.1 Ausführliche Beschreibung . . . 43
5.9.2 Dokumentation der Datenelemente . . . 43
5.9.2.1 bl . . . 43
5.9.2.2 id . . . 44
5.10 bool_2 Strukturreferenz . . . 44
5.10.1 Ausführliche Beschreibung . . . 44
5.10.2 Dokumentation der Datenelemente . . . 44
5.10.2.1 bl . . . 44
5.10.2.2 id . . . 44
5.11 ClientInterface Schnittstellenreferenz . . . 44
5.11.1 Ausführliche Beschreibung . . . 44
5.12 Config Klassenreferenz . . . 45
5.12.1 Ausführliche Beschreibung . . . 48
5.12.2 Beschreibung der Konstruktoren und Destruktoren . . . 48
5.12.2.1 Config . . . 48
5.12.2.2 ∼Config . . . 49
5.12.3 Dokumentation der Elementfunktionen . . . 49
5.12.3.1 getAxisDistance . . . 49
5.12.3.2 getAxisOffset . . . 50
5.12.3.3 getAxisPosX . . . 50
5.12.3.4 getAxisShift . . . 51
5.12.3.5 getAxisVal . . . 51
5.12.3.6 getBallAddr . . . 52
5.12.3.7 getBallAddr . . . 53
5.12.3.8 getBallAddr . . . 53
5.12.3.9 getBallDiameter . . . 54
5.12.3.10 getBallToAxisReachable . . . 54

5.12.3.11 getBallToAxisShootable . . . 55
5.12.3.12 getConfFromFile . . . 56
5.12.3.13 getConnectionInfo . . . 58
5.12.3.14 getConnectionInfo . . . 58
5.12.3.15 getDimension . . . 59
5.12.3.16 getFieldSize . . . 59
5.12.3.17 getFigureFieldDistance . . . 60
5.12.3.18 getFigureFootBreadth . . . 61
5.12.3.19 getFigureFootHeight . . . 61
5.12.3.20 getFigureFootThickness . . . 62
5.12.3.21 getFigurePos . . . 63
5.12.3.22 getFigureVal . . . 63
5.12.3.23 getGoalSize . . . 64
5.12.3.24 getGuiAddr . . . 64
5.12.3.25 getGuiAddr . . . 65
5.12.3.26 getGuiAddr . . . 66
5.12.3.27 getKB . . . 66
5.12.3.28 getKiBallAddr . . . 66
5.12.3.29 getKiBallAddr . . . 67
5.12.3.30 getKiBallAddr . . . 67
5.12.3.31 getKiGuiAddr . . . 68
5.12.3.32 getKiGuiAddr . . . 69
5.12.3.33 getKiGuiAddr . . . 69
5.12.3.34 getKiSPServerAddr . . . 70
5.12.3.35 getKiSPServerAddr . . . 70
5.12.3.36 getKiSPServerAddr . . . 71
5.12.3.37 getShootAngleInfo . . . 71
5.12.3.38 getSPServerAddr . . . 72
5.12.3.39 getSPServerAddr . . . 73
5.12.3.40 getSPServerAddr . . . 73
5.12.3.41 getState . . . 73
5.12.3.42 getState . . . 74
5.12.3.43 getStateInfo . . . 74
5.12.3.44 getStatesInGroup . . . 75
5.12.3.45 getVar . . . 75
5.12.3.46 getVar . . . 76
5.12.3.47 getVar . . . 76
5.12.3.48 getVar . . . 77
5.12.3.49 getVar . . . 77
5.12.3.50 getVar . . . 77

5.12.3.51 getVar . . . 78
5.12.3.52 getVar . . . 78
5.12.3.53 getVar . . . 78
5.12.3.54 getVar . . . 79
5.12.3.55 getVelocityMaxRot . . . 79
5.12.3.56 getVelocityMaxTrans . . . 80
5.12.3.57 setState . . . 80
5.12.3.58 start . . . 81
5.12.3.59 writeConfToFile . . . 83
5.12.4 Dokumentation der Datenelemente . . . 83
5.12.4.1 pKB . . . 83
5.12.4.2 pProfileHandle . . . 83
5.12.4.3 profileDoc . . . 83
5.12.4.4 pVar . . . 83
5.13 ConfigException Klassenreferenz . . . 83
5.13.1 Ausführliche Beschreibung . . . 85
5.13.2 Beschreibung der Konstruktoren und Destruktoren . . . 86
5.13.2.1 ConfigException . . . 86
5.13.3 Dokumentation der Elementfunktionen . . . 86
5.13.3.1 getAttrName . . . 86
5.13.3.2 getValName . . . 86
5.13.4 Dokumentation der Datenelemente . . . 87
5.13.4.1 myAttr . . . 87
5.13.4.2 myVal . . . 87
5.14 Exception Klassenreferenz . . . 87
5.14.1 Ausführliche Beschreibung . . . 88
5.14.2 Beschreibung der Konstruktoren und Destruktoren . . . 88
5.14.2.1 Exception . . . 88
5.14.3 Dokumentation der Elementfunktionen . . . 89
5.14.3.1 getCode . . . 89
5.14.3.2 getDesc . . . 89
5.14.3.3 getErrStr . . . 90
5.14.3.4 getExDesc . . . 90
5.14.3.5 operator= . . . 91
5.14.3.6 setDesc . . . 91
5.14.4 Dokumentation der Datenelemente . . . 91
5.14.4.1 myCode . . . 91
5.14.4.2 myDesc . . . 92
5.15 FileException Klassenreferenz . . . 92
5.15.1 Ausführliche Beschreibung . . . 93

5.15.2 Beschreibung der Konstruktoren und Destruktoren . . . 93
5.15.2.1 FileException . . . 93
5.15.3 Dokumentation der Elementfunktionen . . . 94
5.15.3.1 getFileName . . . 94
5.15.4 Dokumentation der Datenelemente . . . 94
5.15.4.1 myFile . . . 94
5.16 float_1 Strukturreferenz . . . 94
5.16.1 Ausführliche Beschreibung . . . 95
5.16.2 Dokumentation der Datenelemente . . . 95
5.16.2.1 flt . . . 95
5.16.2.2 id . . . 95
5.17 float_2_2_4 Strukturreferenz . . . 95
5.17.1 Ausführliche Beschreibung . . . 95
5.17.2 Dokumentation der Datenelemente . . . 95
5.17.2.1 flt . . . 95
5.17.2.2 id . . . 95
5.18 float_2_4 Strukturreferenz . . . 96
5.18.1 Ausführliche Beschreibung . . . 96
5.18.2 Dokumentation der Datenelemente . . . 96
5.18.2.1 flt . . . 96
5.18.2.2 id . . . 96
5.19 float_4 Strukturreferenz . . . 96
5.19.1 Ausführliche Beschreibung . . . 96
5.19.2 Dokumentation der Datenelemente . . . 97
5.19.2.1 flt . . . 97
5.19.2.2 id . . . 97
5.20 funcElement Strukturreferenz . . . 97
5.20.1 Ausführliche Beschreibung . . . 98
5.20.2 Dokumentation der Datenelemente . . . 98
5.20.2.1 blocking . . . 98
5.20.2.2 func . . . 98
5.20.2.3 next . . . 98
5.20.2.4 timeout . . . 98
5.20.2.5 values . . . 98
5.21 FuncList Klassenreferenz . . . 98
5.21.1 Ausführliche Beschreibung . . . 100
5.21.2 Beschreibung der Konstruktoren und Destruktoren . . . 100
5.21.2.1 FuncList . . . 100
5.21.2.2 ~FuncList . . . 100
5.21.2.3 FuncList . . . 100

5.21.2.4 ∼FuncList . . . 100
5.21.3 Dokumentation der Elementfunktionen . . . 100
5.21.3.1 attach . . . 100
5.21.3.2 attach . . . 101
5.21.3.3 execFunc . . . 101
5.21.3.4 execFunc . . . 101
5.21.4 Dokumentation der Datenelemente . . . 102
5.21.4.1 axis . . . 102
5.21.4.2 firstFunc . . . 102
5.21.4.3 lastFunc . . . 102
5.21.4.4 pKB . . . 102
5.22 Gui Klassenreferenz . . . 102
5.22.1 Ausführliche Beschreibung . . . 103
5.22.2 Beschreibung der Konstruktoren und Destruktoren . . . 103
5.22.2.1 Gui . . . 103
5.22.2.2 ∼Gui . . . 103
5.23 HostException Klassenreferenz . . . 103
5.23.1 Ausführliche Beschreibung . . . 105
5.23.2 Beschreibung der Konstruktoren und Destruktoren . . . 105
5.23.2.1 HostException . . . 105
5.23.3 Dokumentation der Elementfunktionen . . . 106
5.23.3.1 getHostName . . . 106
5.23.4 Dokumentation der Datenelemente . . . 106
5.23.4.1 myHost . . . 106
5.24 Kicker Klassenreferenz . . . 106
5.24.1 Ausführliche Beschreibung . . . 110
5.24.2 Beschreibung der Konstruktoren und Destruktoren . . . 110
5.24.2.1 Kicker . . . 110
5.24.2.2 ∼Kicker . . . 110
5.24.3 Dokumentation der Elementfunktionen . . . 110
5.24.3.1 getComputerAxisPos . . . 111
5.24.3.2 getComputerAxisPos . . . 111
5.24.3.3 getHumanAxisPos . . . 111
5.24.3.4 getHumanAxisPos . . . 112
5.24.3.5 getScore . . . 112
5.24.3.6 requestAxisPos . . . 112
5.24.3.7 requestScore . . . 113
5.24.3.8 sendAxisPos . . . 113
5.24.3.9 sendScore . . . 114
5.24.3.10 setAxisPos . . . 114

5.24.3.11 setComputerAxisPos . . . 115
5.24.3.12 setHumanAxisPos . . . 116
5.24.3.13 setScore . . . 116
5.24.3.14 start . . . 117
5.24.3.15 subscribeScore . . . 117
5.24.4 Dokumentation der Datenelemente . . . 118
5.24.4.1 aafComputerSendPosRot . . . 118
5.24.4.2 aafComputerSendPosTrans . . . 118
5.24.4.3 afComputerAxisPosRot . . . 118
5.24.4.4 afComputerAxisPosTrans . . . 118
5.24.4.5 afHumanAxisPosRot . . . 118
5.24.4.6 afHumanAxisPosTrans . . . 118
5.24.4.7 allAxisSet . . . 119
5.24.4.8 ausScore . . . 119
5.24.4.9 client . . . 119
5.24.4.10 lastSend . . . 119
5.24.4.11 mutexHandle . . . 119
5.24.4.12 pConf . . . 119
5.24.4.13 pKB . . . 120
5.25 KnowledgeBase Klassenreferenz . . . 120
5.25.1 Ausführliche Beschreibung . . . 121
5.25.2 Beschreibung der Konstruktoren und Destruktoren . . . 122
5.25.2.1 KnowledgeBase . . . 122
5.25.2.2 ∼KnowledgeBase . . . 122
5.25.3 Dokumentation der Elementfunktionen . . . 123
5.25.3.1 axisUpdate . . . 123
5.25.3.2 ballUpdate . . . 123
5.25.3.3 deleteStates . . . 124
5.25.3.4 getBall . . . 124
5.25.3.5 getConf . . . 125
5.25.3.6 getKicker . . . 125
5.25.3.7 getRunKBInstance . . . 126
5.25.3.8 initStates . . . 127
5.25.3.9 linkStates . . . 127
5.25.3.10 runKB . . . 127
5.25.3.11 start . . . 128
5.25.3.12 unlinkStates . . . 128
5.25.4 Dokumentation der Datenelemente . . . 129
5.25.4.1 aaStates . . . 129
5.25.4.2 myStopped . . . 129

5.25.4.3 myThreadHandle . . . 129
5.25.4.4 pBall . . . 129
5.25.4.5 pConf . . . 129
5.25.4.6 pKicker . . . 130
5.26 KnowlegeBase Klassenreferenz . . . 130
5.26.1 Ausführliche Beschreibung . . . 130
5.27 Master Klassenreferenz . . . 130
5.27.1 Ausführliche Beschreibung . . . 131
5.27.2 Beschreibung der Konstruktoren und Destruktoren . . . 131
5.27.2.1 Master . . . 131
5.27.2.2 ∼Master . . . 132
5.27.3 Dokumentation der Elementfunktionen . . . 132
5.27.3.1 run . . . 132
5.27.4 Dokumentation der Datenelemente . . . 132
5.27.4.1 apAA . . . 133
5.27.4.2 pKB . . . 133
5.28 MutexException Klassenreferenz . . . 133
5.28.1 Ausführliche Beschreibung . . . 134
5.28.2 Beschreibung der Konstruktoren und Destruktoren . . . 134
5.28.2.1 MutexException . . . 134
5.28.3 Dokumentation der Elementfunktionen . . . 135
5.28.3.1 getWinError . . . 135
5.28.4 Dokumentation der Datenelemente . . . 135
5.28.4.1 myWinErr . . . 135
5.29 Observer Klassenreferenz . . . 135
5.29.1 Ausführliche Beschreibung . . . 137
5.29.2 Beschreibung der Konstruktoren und Destruktoren . . . 137
5.29.2.1 ∼Observer . . . 137
5.29.2.2 Observer . . . 137
5.29.3 Dokumentation der Elementfunktionen . . . 137
5.29.3.1 update . . . 137
5.29.3.2 update . . . 137
5.29.3.3 update . . . 138
5.30 ObsListKnot Strukturreferenz . . . 138
5.30.1 Ausführliche Beschreibung . . . 139
5.30.2 Dokumentation der Datenelemente . . . 139
5.30.2.1 next . . . 139
5.30.2.2 obs . . . 139
5.30.2.3 prev . . . 139
5.31 Publish Klassenreferenz . . . 139

5.31.1 Ausführliche Beschreibung . . . 142
5.31.2 Beschreibung der Konstruktoren und Destruktoren . . . 142
5.31.2.1 ~Publish . . . 142
5.31.2.2 Publish . . . 142
5.31.3 Dokumentation der Elementfunktionen . . . 142
5.31.3.1 attach . . . 142
5.31.3.2 detach . . . 143
5.31.3.3 notify . . . 143
5.31.3.4 notify . . . 144
5.31.3.5 notify . . . 144
5.31.4 Dokumentation der Datenelemente . . . 145
5.31.4.1 firstObs . . . 145
5.31.4.2 lastObs . . . 145
5.32 requestPacket Strukturreferenz . . . 145
5.32.1 Ausführliche Beschreibung . . . 146
5.32.2 Dokumentation der Datenelemente . . . 146
5.32.2.1 id . . . 147
5.32.2.2 request . . . 147
5.33 SpsClient Klassenreferenz . . . 147
5.33.1 Ausführliche Beschreibung . . . 150
5.33.2 Beschreibung der Konstruktoren und Destruktoren . . . 150
5.33.2.1 SpsClient . . . 150
5.33.2.2 ~SpsClient . . . 151
5.33.3 Dokumentation der Elementfunktionen . . . 151
5.33.3.1 requestAxisPos . . . 151
5.33.3.2 requestScore . . . 151
5.33.3.3 sendAxisPos . . . 151
5.33.3.4 sendScore . . . 152
5.33.3.5 start . . . 152
5.33.3.6 subscribeScore . . . 152
5.33.3.7 update . . . 153
5.33.3.8 update . . . 153
5.33.3.9 update . . . 153
5.33.4 Dokumentation der Datenelemente . . . 153
5.33.4.1 myRecvData . . . 153
5.33.4.2 myServerAddr . . . 154
5.33.4.3 mySocket . . . 154
5.33.4.4 myUser . . . 154
5.34 SpsClientInterface Klassenreferenz . . . 154
5.34.1 Ausführliche Beschreibung . . . 156

5.34.2 Beschreibung der Konstruktoren und Destruktoren . . . 156
5.34.2.1 ~SpsClientInterface . . . 156
5.34.3 Dokumentation der Elementfunktionen . . . 156
5.34.3.1 requestAxisPos . . . 156
5.34.3.2 requestScore . . . 156
5.34.3.3 sendAxisPos . . . 157
5.34.3.4 sendScore . . . 157
5.34.3.5 start . . . 158
5.34.3.6 subscribeScore . . . 158
5.35 SpsClientUserInterface Schnittstellenreferenz . . . 159
5.35.1 Ausführliche Beschreibung . . . 160
5.35.2 Beschreibung der Konstruktoren und Destruktoren . . . 160
5.35.2.1 ~SpsClientUserInterface . . . 160
5.35.3 Dokumentation der Elementfunktionen . . . 160
5.35.3.1 setComputerAxisPos . . . 160
5.35.3.2 setHumanAxisPos . . . 160
5.35.3.3 setScore . . . 161
5.35.3.4 start . . . 161
5.36 State Klassenreferenz . . . 161
5.36.1 Ausführliche Beschreibung . . . 164
5.36.2 Beschreibung der Konstruktoren und Destruktoren . . . 164
5.36.2.1 State . . . 164
5.36.3 Dokumentation der Elementfunktionen . . . 165
5.36.3.1 dec . . . 165
5.36.3.2 diffToMax . . . 165
5.36.3.3 diffToMin . . . 165
5.36.3.4 get . . . 165
5.36.3.5 getMax . . . 166
5.36.3.6 getMin . . . 166
5.36.3.7 inc . . . 166
5.36.3.8 set . . . 166
5.36.3.9 update . . . 167
5.36.3.10 update . . . 167
5.36.3.11 update . . . 167
5.36.4 Dokumentation der Datenelemente . . . 168
5.36.4.1 iGroup . . . 168
5.36.4.2 iState . . . 168
5.36.4.3 iStateMax . . . 168
5.36.4.4 iStateMin . . . 168
5.36.4.5 mutexHandle . . . 169

5.36.4.6 pConf . . . 169
5.36.4.7 pKB . . . 169
5.37 StateException Klassenreferenz . . . 169
5.37.1 Ausführliche Beschreibung . . . 171
5.37.2 Beschreibung der Konstruktoren und Destruktoren . . . 172
5.37.2.1 StateException . . . 172
5.37.3 Dokumentation der Elementfunktionen . . . 172
5.37.3.1 getState . . . 172
5.37.3.2 getStateCategory . . . 172
5.37.4 Dokumentation der Datenelemente . . . 172
5.37.4.1 myState . . . 172
5.37.4.2 myStateCategory . . . 172
5.38 ThreadException Klassenreferenz . . . 173
5.38.1 Ausführliche Beschreibung . . . 174
5.38.2 Beschreibung der Konstruktoren und Destruktoren . . . 174
5.38.2.1 ThreadException . . . 174
5.38.3 Dokumentation der Elementfunktionen . . . 175
5.38.3.1 getThreadClassName . . . 175
5.38.4 Dokumentation der Datenelemente . . . 175
5.38.4.1 myThreadClass . . . 175
5.39 tObsListKnot Strukturreferenz . . . 175
5.39.1 Ausführliche Beschreibung . . . 175
5.40 ubf16_request Variantenreferenz . . . 175
5.40.1 Ausführliche Beschreibung . . . 176
5.40.2 Dokumentation der Datenelemente . . . 176
5.40.2.1 data . . . 176
5.40.2.2 value . . . 177
5.41 UDPsocket Klassenreferenz . . . 177
5.41.1 Ausführliche Beschreibung . . . 180
5.41.2 Beschreibung der Konstruktoren und Destruktoren . . . 181
5.41.2.1 UDPsocket . . . 181
5.41.2.2 ∼UDPsocket . . . 181
5.41.3 Dokumentation der Elementfunktionen . . . 181
5.41.3.1 getAddrFromString . . . 181
5.41.3.2 getData . . . 182
5.41.3.3 getRecvDataInstance . . . 182
5.41.3.4 pause . . . 183
5.41.3.5 recvData . . . 183
5.41.3.6 sendData . . . 184
5.41.3.7 start . . . 184

5.41.4 Dokumentation der Datenelemente . . . 185
5.41.4.1 myAddr . . . 185
5.41.4.2 myBuffer . . . 185
5.41.4.3 myBufferSize . . . 185
5.41.4.4 myDataSize . . . 186
5.41.4.5 myPaused . . . 186
5.41.4.6 mySenderAddr . . . 186
5.41.4.7 mySock . . . 186
5.41.4.8 myStopped . . . 187
5.41.4.9 myThreadHandle . . . 187
5.41.4.10 myWsa . . . 187
5.41.4.11 myWsaErr . . . 187
5.42 unsignedShort_2 Strukturreferenz . . . 187
5.42.1 Ausführliche Beschreibung . . . 188
5.42.2 Dokumentation der Datenelemente . . . 188
5.42.2.1 id . . . 188
5.42.2.2 us . . . 188

6 Datei-Dokumentation 189

6.1 AxisAgent.cpp-Dateireferenz . . . 189
6.1.1 Ausführliche Beschreibung . . . 189
6.2 AxisAgent.h-Dateireferenz . . . 189
6.2.1 Ausführliche Beschreibung . . . 190
6.2.2 Dokumentation der benutzerdefinierten Typen . . . 191
6.2.2.1 funcElement . . . 191
6.2.2.2 moveFunc . . . 191
6.3 AxisAgent_wobbeln.cpp-Dateireferenz . . . 191
6.4 AxisAgent_wobbeln.h-Dateireferenz . . . 191
6.4.1 Dokumentation der benutzerdefinierten Typen . . . 192
6.4.1.1 funcElement . . . 192
6.4.1.2 moveFunc . . . 192
6.5 AxisKinetics.h-Dateireferenz . . . 192
6.5.1 Ausführliche Beschreibung . . . 194
6.5.2 Dokumentation der Funktionen . . . 194
6.5.2.1 calcPosWhereBallCrossesAxis . . . 194
6.5.2.2 checkBallShootable . . . 195
6.5.2.3 checkMaxRotVelocity . . . 196
6.5.2.4 checkMaxTransVelocity . . . 197
6.5.2.5 checkPosReached . . . 198
6.5.2.6 chooseFigure . . . 199

6.5.2.7 findAngleFromBallToGoal . . . 200
6.5.2.8 moveAxisPTP . . . 201
6.5.2.9 moveAxisTo . . . 202
6.5.2.10 moveFeetInCircularMotion . . . 203
6.5.2.11 prepareShoot . . . 204
6.5.2.12 shootBall . . . 205
6.6 Ball.cpp-Dateireferenz . . . 206
6.6.1 Ausführliche Beschreibung . . . 207
6.7 Ball.h-Dateireferenz . . . 207
6.7.1 Ausführliche Beschreibung . . . 208
6.8 BallClient.cpp-Dateireferenz . . . 208
6.8.1 Ausführliche Beschreibung . . . 209
6.9 BallClient.h-Dateireferenz . . . 209
6.9.1 Ausführliche Beschreibung . . . 210
6.10 BallClientInterface.h-Dateireferenz . . . 210
6.10.1 Ausführliche Beschreibung . . . 211
6.11 BallClientUserInterface.h-Dateireferenz . . . 211
6.11.1 Ausführliche Beschreibung . . . 212
6.12 Config.cpp-Dateireferenz . . . 212
6.12.1 Ausführliche Beschreibung . . . 213
6.13 Config.h-Dateireferenz . . . 213
6.13.1 Ausführliche Beschreibung . . . 214
6.14 ConfigDefines.h-Dateireferenz . . . 214
6.14.1 Ausführliche Beschreibung . . . 215
6.14.2 Dokumentation der Aufzählungstypen . . . 215
6.14.2.1 ConfigVar . . . 215
6.15 defines.h-Dateireferenz . . . 216
6.15.1 Ausführliche Beschreibung . . . 217
6.15.2 Makro-Dokumentation . . . 217
6.15.2.1 BUFFER_SIZE . . . 217
6.15.2.2 MAX_SHOOT_OFFSETS . . . 218
6.15.3 Dokumentation der Aufzählungstypen . . . 218
6.15.3.1 Axis . . . 218
6.15.3.2 Coordinate . . . 218
6.15.3.3 Data_Type . . . 218
6.15.3.4 Data_Value . . . 218
6.15.3.5 Figure . . . 219
6.15.3.6 Gamer . . . 219
6.15.3.7 Safety . . . 219
6.15.3.8 ShootOffset . . . 219

6.15.4 Variablen-Dokumentation . . . 219
6.15.4.1 FIELDSIZE_X . . . 219
6.15.4.2 FIELDSIZE_Z . . . 220
6.15.4.3 GOAL_POSITION_NEGATIVE_Z . . . 220
6.15.4.4 GOAL_POSITION_POSITIVE_Z . . . 220
6.15.4.5 GOAL_WIDTH . . . 220
6.15.4.6 PI . . . 220
6.16 ErrorCodes.h-Dateireferenz . . . 220
6.16.1 Ausführliche Beschreibung . . . 221
6.16.2 Makro-Dokumentation . . . 221
6.16.2.1 ERROR_AXIS_BLOCK . . . 221
6.16.2.2 ERROR_BALL . . . 221
6.16.2.3 ERROR_CONFIG . . . 221
6.16.2.4 ERROR_FILE . . . 221
6.16.2.5 ERROR_KICKER . . . 221
6.16.2.6 ERROR_KNOWLEDGEBASE . . . 222
6.16.2.7 ERROR_MUTEX . . . 222
6.16.2.8 ERROR_MUTEX_WAIT . . . 222
6.16.2.9 ERROR_NONE . . . 222
6.16.2.10 ERROR_STATE . . . 222
6.16.2.11 ERROR_THREAD . . . 222
6.16.2.12 ERROR_UDP_NO_HOST . . . 222
6.16.2.13 ERROR_UNKNOWN . . . 222
6.17 Exception.h-Dateireferenz . . . 222
6.17.1 Ausführliche Beschreibung . . . 224
6.17.2 Dokumentation der benutzerdefinierten Typen . . . 224
6.17.2.1 DWORD . . . 224
6.18 Gui.cpp-Dateireferenz . . . 224
6.18.1 Ausführliche Beschreibung . . . 224
6.19 Gui.h-Dateireferenz . . . 225
6.19.1 Ausführliche Beschreibung . . . 225
6.20 Kicker.cpp-Dateireferenz . . . 226
6.20.1 Ausführliche Beschreibung . . . 226
6.21 Kicker.h-Dateireferenz . . . 226
6.21.1 Ausführliche Beschreibung . . . 227
6.22 KnowledgeBase.cpp-Dateireferenz . . . 228
6.22.1 Ausführliche Beschreibung . . . 228
6.23 KnowledgeBase.h-Dateireferenz . . . 228
6.23.1 Ausführliche Beschreibung . . . 229
6.24 main.cpp-Dateireferenz . . . 230

6.24.1 Ausführliche Beschreibung . . . 230
6.24.2 Dokumentation der Funktionen . . . 230
6.24.2.1 main . . . 230
6.25 Master.cpp-Dateireferenz . . . 231
6.25.1 Ausführliche Beschreibung . . . 232
6.26 Master.h-Dateireferenz . . . 232
6.26.1 Ausführliche Beschreibung . . . 233
6.27 observer.h-Dateireferenz . . . 233
6.27.1 Ausführliche Beschreibung . . . 234
6.28 publish.h-Dateireferenz . . . 234
6.28.1 Ausführliche Beschreibung . . . 235
6.28.2 Makro-Dokumentation . . . 235
6.28.2.1 NULL . . . 235
6.28.3 Dokumentation der benutzerdefinierten Typen . . . 235
6.28.3.1 ObsListKnot_t . . . 235
6.29 SpsClient.cpp-Dateireferenz . . . 236
6.29.1 Ausführliche Beschreibung . . . 236
6.30 SpsClient.h-Dateireferenz . . . 236
6.30.1 Ausführliche Beschreibung . . . 237
6.31 SpsClientInterface.h-Dateireferenz . . . 238
6.31.1 Ausführliche Beschreibung . . . 238
6.32 SpsClientUserInterface.h-Dateireferenz . . . 238
6.32.1 Ausführliche Beschreibung . . . 239
6.33 StateDefines.h-Dateireferenz . . . 239
6.33.1 Ausführliche Beschreibung . . . 241
6.33.2 Makro-Dokumentation . . . 241
6.33.2.1 MAX_STATES_PER_GROUP . . . 241
6.33.3 Dokumentation der Aufzählungstypen . . . 241
6.33.3.1 AxisStates . . . 241
6.33.3.2 BallPosStates . . . 242
6.33.3.3 BallPosToAxisStates . . . 242
6.33.3.4 BallStates . . . 242
6.33.3.5 ConfigStates . . . 242
6.33.3.6 GameStates . . . 243
6.33.3.7 GlobalStates . . . 243
6.33.3.8 KickerStates . . . 243
6.33.3.9 KnowledgeBaseStates . . . 243
6.33.3.10 StateIDs . . . 243
6.33.3.11 StateVals . . . 244
6.34 States.h-Dateireferenz . . . 244

6.34.1 Ausführliche Beschreibung 245
6.35 UdpDefines.h-Dateireferenz 246
6.35.1 Ausführliche Beschreibung 248
6.35.2 Dokumentation der benutzerdefinierten Typen 248
6.35.2.1 ballCoordinate_t 248
6.35.2.2 ballPacket 248
6.35.2.3 bool_1 248
6.35.2.4 bool_2 248
6.35.2.5 float_1 248
6.35.2.6 float_2_2_4 248
6.35.2.7 float_2_4 249
6.35.2.8 float_4 249
6.35.2.9 requestPacket 249
6.35.2.10 ubf16_request 249
6.35.2.11 unsignedShort_2 249
6.35.3 Dokumentation der Aufzählungstypen 249
6.35.3.1 PacketID 249
6.35.4 Variablen-Dokumentation 250
6.35.4.1 MAX_VELOCITY_ROT 250
6.35.4.2 MAX_VELOCITY_TRANS 250
6.35.4.3 TRAVERSE_DISTANCE_1 250
6.35.4.4 TRAVERSE_DISTANCE_2 250
6.35.4.5 TRAVERSE_DISTANCE_3 250
6.35.4.6 TRAVERSE_DISTANCE_5 250
6.36 UdpSocket.cpp-Dateireferenz 251
6.36.1 Ausführliche Beschreibung 251
6.37 UdpSocket.h-Dateireferenz 251
6.37.1 Ausführliche Beschreibung 252

Kapitel 1

Ausstehende Aufgaben

Element AxisAgent::mutexHandle

Benutzung vom Mutex beschreiben

Benutzung vom Mutex beschreiben

Benutzung vom Mutex beschreiben

Klasse Ball

Exception-Handling.

Element Ball::mutexHandle

Benutzung vom Mutex beschreiben

Element chooseFigure (unsigned int axis, float &posY, KnowledgeBase *pKB)

Nähe der Spielfigur zur Zielposition im Gütefaktor überprüfen!

Klasse Config

writeConfToFile() und Werte zuweisen umsetzen.

Exception-Handling.

Element Config::getState (unsigned int group, unsigned int state, int &val)

Als getVar() umschreiben

Element Config::getState (unsigned int group, unsigned int state, int(&val)[3])

Als getVar() umschreiben

Element Config::getStatesInGroup (unsigned int group, unsigned int &states)

Als getVar() umschreiben

Element Config::setState (unsigned int group, unsigned int state, int val)

Als setVar() umschreiben

Element Config::~Config (void)

pVar löschen!

Element findAngleFromBallToGoal (unsigned int axis, float &angle, KnowledgeBase *pKB)

Bei der Schnittpunktberechnung auf Werte ausserhalb des Spielfeldes überprüfen!

Element Kicker::mutexHandle

Benutzung vom Mutex beschreiben

Element Kicker::setAxisPos (unsigned int axis, float axisPos[2], float axisAngle[2])

Bedingung mit allAxisSet[] überdenken!!! Führt manchmal zu seltsamen Verhalten.

Element KnowledgeBase::axisUpdate (Gamer gamer, Axis axis)

Funktion umsetzen

Element KnowledgeBase::ballUpdate (void)

Funktion umsetzen

Element shootBall (unsigned int axis, float values[4], KnowledgeBase *pKB, bool blocking=true, int timeout=1000)

Andere Schussrichtungen wie nach vorne implementieren (seitlich und nach hinten).

Kapitel 2

Klassen-Verzeichnis

2.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

AxisAgent 9
BallClientInterface 34
  BallClient 26
BallClientUserInterface 37
  Ball 18
ballCoordinate_t 39
ballPacket 40
ubf16_request::bitfeld_16 41
bool_1 43
bool_2 44
ClientInterface 44
Config 45
Exception 87
  ConfigException 83
  FileException 92
  HostException 103
  MutexException 133
  StateException 169
  ThreadException 173
float_1 94
float_2_2_4 95
float_2_4 96
float_4 96
funcElement 97
FuncList 98
Gui 102
KnowledgeBase 120
KnowlegeBase 130
Master 130
Observer 135
  BallClient 26
  SpsClient 147
  State 161
ObsListKnot 138
Publish 139
  State 161
  UDPsocket 177

requestPacket 145
SpsClientInterface 154
SpsClient 147
SpsClientUserInterface 159
Kicker 106
tObsListKnot 175
ubf16_request 175
unsignedShort_2 187

Kapitel 3

Klassen-Verzeichnis

3.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

AxisAgent
Ein Agent zur Ansteuerung einer Spielstange 9
Ball
Zugriff zur Ballerkennung 18
BallClient
Ein UDP-BallClient 26
BallClientInterface 34
BallClientUserInterface
Interface zur Kommunikation mit dem Benutzer des Ball-Clients 37
ballCoordinate_t
Struktur der Ballposition 39
ballPacket
Paket mit der Ballposition 40
ubf16_request::bitfeld_16 41
bool_1
Anfrage-Paket für bestimmte Daten 43
bool_2
Anfrage-Paket für bestimmte Daten 44
ClientInterface
Interface zur Kommunikation mit dem UDP-Client 44
Config
Configurationsverwaltung 45
ConfigException
Exception-Klasse für die Konfigurationsdateien 83
Exception
Basis-Klasse von der alle Exception-Klassen abgeleitet werden sollten 87
FileException
Exception-Klasse für Dateizugriff 92
float_1
Anfrage-Paket für bestimmte Daten 94
float_2_2_4
Anfrage-Paket für bestimmte Daten 95
float_2_4
Anfrage-Paket für bestimmte Daten 96
float_4
Anfrage-Paket für bestimmte Daten 96
funcElement
Struktur für ein Listenelement 97

FuncList
Eine einfach verkettete Liste für Funktionspointer . . . 98
Gui . . . 102
HostException
Exception-Klasse für die Host-Auflösung . . . 103
Kicker
Eine Klasse, die den UDP-Client anlegt und benutzt . . . 106
KnowledgeBase . . . 120
KnowlegeBase
Diese Klasse stellt die Wissensbasis dar . . . 130
Master . . . 130
MutexException
Exception-Klasse für die Mutexzugriffe . . . 133
Observer
Observer Klasse für das Observer Entwurfsmodell . . . 135
ObsListKnot . . . 138
Publish
Publisher Klasse für das Observer Entwurfsmodell . . . 139
requestPacket
Anfrage-Paket für bestimmte Daten . . . 145
SpsClient
Ein UDP-Client . . . 147
SpsClientInterface . . . 154
SpsClientUserInterface
Interface zur Kommunikation mit dem Benutzer des UDP-Clients . . . 159
State
Status einer Eigenschaft . . . 161
StateException
Exception-Klasse für Zustände . . . 169
ThreadException
Exception-Klasse für Threads . . . 173
tObsListKnot
Struktur für ein Listenelement . . . 175
ubf16_request
Bitfeld, bei dem jedes Bit bestimmte Daten anfordert . . . 175
UDPsocket
Stellt ein UDP-Socket dar . . . 177
unsignedShort_2
Anfrage-Paket für bestimmte Daten . . . 187

Kapitel 4

Datei-Verzeichnis

4.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

AxisAgent.cpp
Enthält die Implementierung der Klasse AxisAgent 189
AxisAgent.h
Enthält die Definition der Klasse AxisAgent 189
AxisAgent_wobbeln.cpp 191
AxisAgent_wobbeln.h 191
AxisKinetics.h
Enthält Funktionen zur Bewegungssteuerung der Spielstangen 192
Ball.cpp
Enthält die Implementierung der Klasse Ball 206
Ball.h
Enthält die Definition der Klasse Ball 207
BallClient.cpp
Enthält die Implementierung der Klasse BallClient 208
BallClient.h
Enthält die Definition der Klasse BallClient 209
BallClientInterface.h
Enthält das Interface zur Kommunikation mit dem UDP-Client 210
BallClientUserInterface.h
Enthält das Interface zur Kommunikation mit dem Benutzer des UDP-Clients 211
Config.cpp
Enthält die Implementierung der Klasse Config 212
Config.h
Enthält die Definition der Klasse Config 213
ConfigDefines.h
Enthält die Defines der Klasse Config 214
defines.h
Enthält alle nötigen Defines 216
ErrorCodes.h
Enthält die Defines der Fehlercodes 220
Exception.h
Enthält die Definition der Exception-Klassen 222
Gui.cpp
Enthält die Implementierung der Klasse Gui 224
Gui.h
Enthält die Definition der Klasse Gui 225
Kicker.cpp
Enthält die Implementierung der Klasse Kicker 226

Kicker.h
Enthält die Definition der Klasse Kicker 226
KnowledgeBase.cpp
Enthält die Implementierung der Klasse KnowledgeBase 228
KnowledgeBase.h
Enthält die Definition der Klasse KnowledgeBase 228
main.cpp
Enthält die main-Funktion 230
Master.cpp
Enthält die Implementierung der Klasse Master 231
Master.h
Enthält die Definition der Klasse Master 232
observer.h
Enthält die Definition und Implementierung der Klasse Observer 233
publish.h
Enthält die Definition und Implementierung der Klasse Publish 234
SpsClient.cpp
Enthält die Implementierung der Klasse Client 236
SpsClient.h
Enthält die Definition der Klasse Client 236
SpsClientInterface.h
Enthält das Interface zur Kommunikation mit dem UDP-Client 238
SpsClientUserInterface.h
Enthält das Interface zur Kommunikation mit dem Benutzer des UDP-Clients 238
StateDefines.h
Enthält die Defines der Klasse State 239
States.h
Enthält die Definition der einzelnen Zustände 244
UdpDefines.h
Enthält die defines der verschiedenen Strukturen 246
UdpSocket.cpp
Enthält die Implementierung der Klasse UDPsocket 251
UdpSocket.h
Enthält die Definition der Klasse UDPsocket 251

Kapitel 5

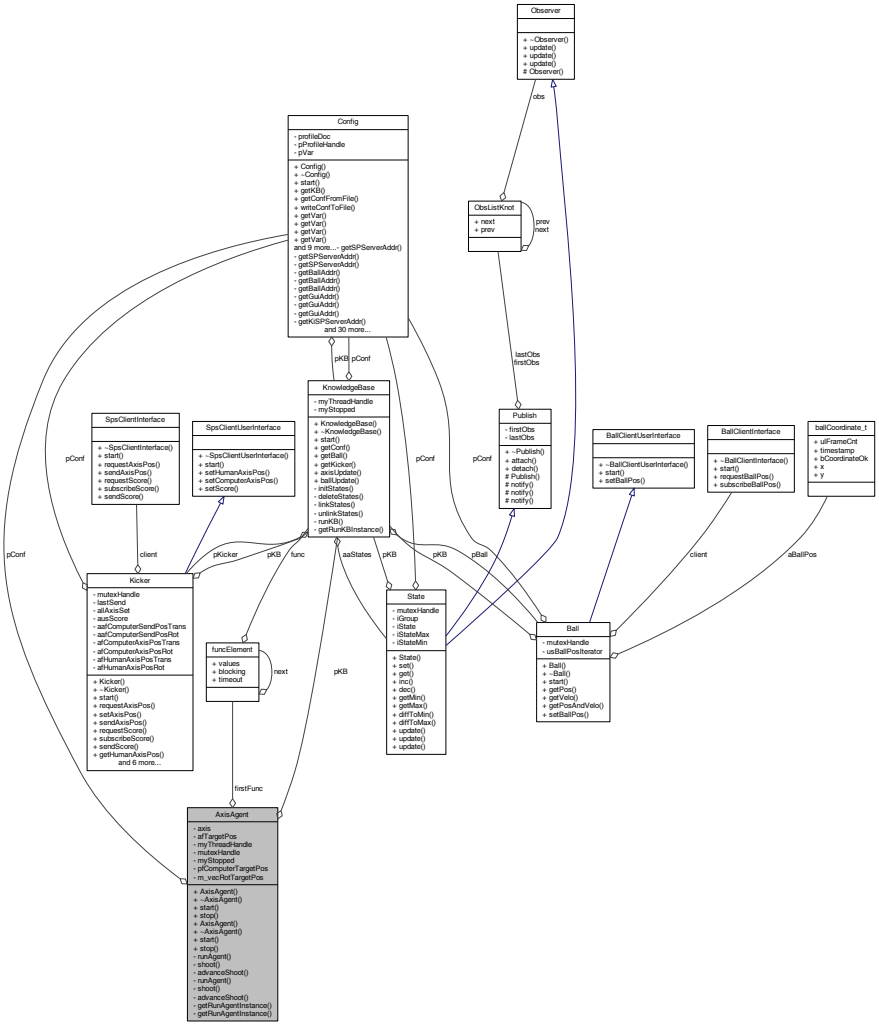
Klassen-Dokumentation

5.1 AxisAgent Klassenreferenz

Ein Agent zur Ansteuerung einer Spielstange.

`#include <AxisAgent.h>`

Zusammengehörigkeiten von AxisAgent:



Öffentliche Methoden

- AxisAgent (Axis ax, KnowledgeBase *kB)

  Konstruktor.
- ~AxisAgent (void)

  Destruktor.
- void start (HANDLE &threadHandle)

  Startet das System.
- void stop (void)

  Stoppt das System.
- AxisAgent (Axis ax, KnowledgeBase *kB)

  Konstruktor.
- ~AxisAgent (void)

  Destruktor.
- void start (HANDLE &threadHandle)

  Startet das System.
- void stop (void)

  Stoppt das System.

Private Methoden

- DWORD runAgent (void)

  Eigener Thread, welcher eine Spielstange ansteuert.
- void shoot (float fAxisRotAct, float fFromAngle=-45.f, float fToAngle=+45.f)
- bool advanceShoot (float fAxisRotAct)
- DWORD runAgent (void)

  Eigener Thread, welcher eine Spielstange ansteuert.
- void shoot (float fAxisRotAct, float fFromAngle=-45.f, float fToAngle=+45.f)
- bool advanceShoot (float fAxisRotAct)

Private, statische Methoden

- static DWORD WINAPI getRunAgentInstance (void *instance)

  Gibt den Funktionspointer auf die Funktion runAgent() zurück.
- static DWORD WINAPI getRunAgentInstance (void *instance)

  Gibt den Funktionspointer auf die Funktion runAgent() zurück.

Private Attribute

- Axis axis

  Spielstange, welche der Agent ansteuern soll.
- KnowledgeBase * pKB

  Zeiger auf die Wissensbasis, zur Kommunikation.
- Config * pConf

  Zugriff auf die Konfigurationen.
- funcElement * firstFunc

  Zugriff auf die SPS.
- float afTargetPos [4]

  Zielposition und Geschwindigkeit.
- HANDLE myThreadHandle

Handle vom Thread.

- HANDLE mutexHandle

  Mutex-Handle.
- bool myStopped

  Wird auf true gesetzt wenn der Agent beendet werden soll.
- float pfComputerTargetPos [2][2]
- std::deque< float > m_vecRotTargetPos

5.1.1 Ausführliche Beschreibung

Ein Agent zur Ansteuerung einer Spielstange.

Definiert in Zeile 139 der Datei AxisAgent.h.

5.1.2 Beschreibung der Konstruktoren und Destruktoren

5.1.2.1 AxisAgent::AxisAgent (Axis *ax*, KnowledgeBase * *kB*)

Konstruktor.

Parameter

in	*ax*	Spielstange, welche der Agent ansteuern soll
in	*kB*	Zeiger auf die Wissensbasis, zur Kommunikation

Definiert in Zeile 25 der Datei AxisAgent.cpp.

Benutzt axis, AXIS_FOUR, AXIS_ONE, firstFunc, KnowledgeBase::getConf(), mutexHandle, myStopped, NULL, pConf und pKB.

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.1.2.2 AxisAgent::~AxisAgent (void)

Destruktor.

Definiert in Zeile 57 der Datei AxisAgent.cpp.

Benutzt axis, mutexHandle, myStopped und myThreadHandle.

5.1.2.3 AxisAgent::AxisAgent (Axis *ax*, KnowledgeBase * *kB*)

Konstruktor.

Parameter

in	*ax*	Spielstange, welche der Agent ansteuern soll
in	*kB*	Zeiger auf die Wissensbasis, zur Kommunikation

5.1.2.4 AxisAgent::~AxisAgent (void)

Destruktor.

5.1.3 Dokumentation der Elementfunktionen

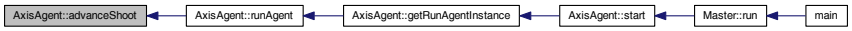
5.1.3.1 bool AxisAgent::advanceShoot (float *fAxisRotAct*) [private]

Definiert in Zeile 581 der Datei AxisAgent.cpp.

Benutzt ANGLE, m_vecRotTargetPos, pfComputerTargetPos und VALUE.

Wird benutzt von runAgent().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.1.3.2 bool AxisAgent::advanceShoot (float *fAxisRotAct*) [private]

5.1.3.3 static DWORD WINAPI AxisAgent::getRunAgentInstance (void * *instance*) [static, private]

Gibt den Funktionspointer auf die Funktion runAgent() zurück.

Parameter

in	*instance*	Pointer auf die Instanz dieser Klasse

Rückgabe

Funktionspointer auf die Funktion runAgent() der aktuellen Instanz

Trick um einen neuen Thread mit einer Memberfunktion zu erzeugen.

5.1.3.4 DWORD WINAPI AxisAgent::getRunAgentInstance (void * *instance*) [static, private]

Gibt den Funktionspointer auf die Funktion runAgent() zurück.

Parameter

in	*instance*	Pointer auf die Instanz dieser Klasse

Rückgabe

Funktionspointer auf die Funktion runAgent() der aktuellen Instanz

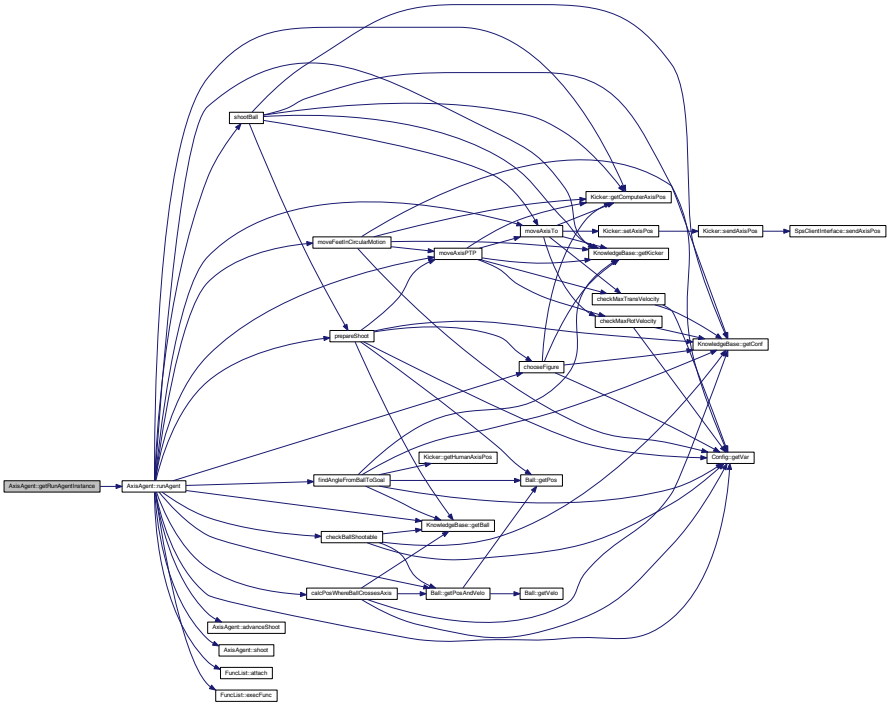
Trick um einen neuen Thread mit einer Memberfunktion zu erzeugen.

Definiert in Zeile 567 der Datei AxisAgent.cpp.

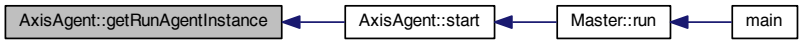
Benutzt NULL und runAgent().

Wird benutzt von start().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.1.3.5 DWORD AxisAgent::runAgent (void) [private]

Eigener Thread, welcher eine Spielstange ansteuert.

5.1.3.6 DWORD AxisAgent::runAgent (void) [private]

Eigener Thread, welcher eine Spielstange ansteuert.

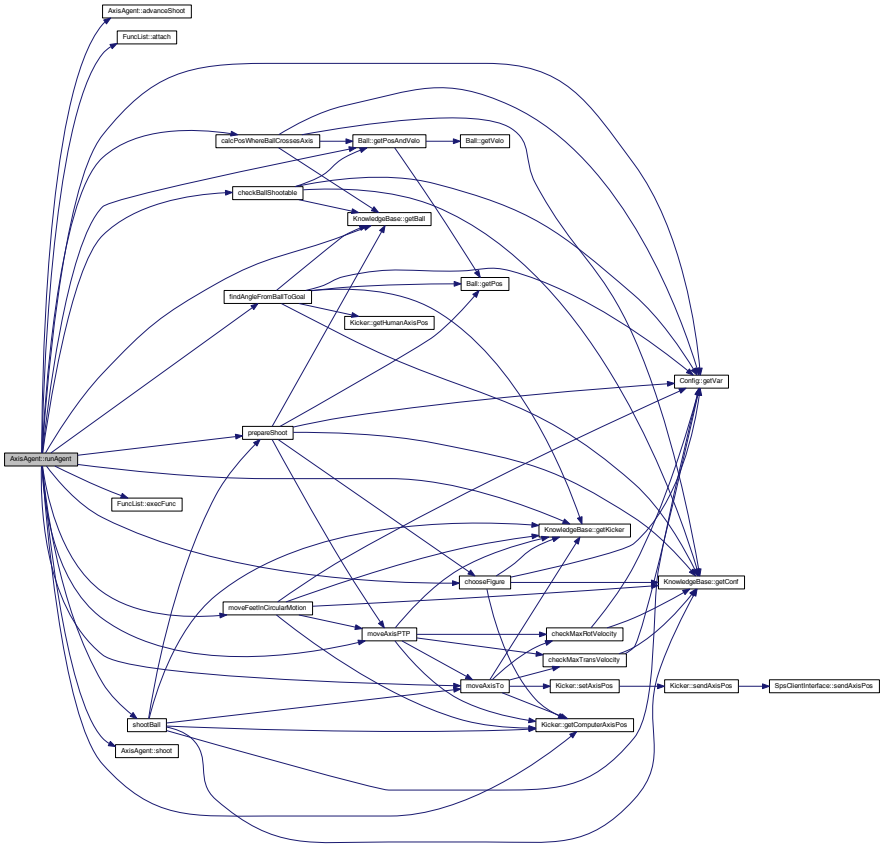
Definiert in Zeile 170 der Datei AxisAgent.cpp.

Benutzt advanceShoot(), ANGLE, FuncList::attach(), axis, AXIS_FOUR, AXIS_ONE, AXIS_THREE, AXIS_TWO, AxisPosX, calcPosWhereBallCrossesAxis(), checkBallShootable(), chooseFigure(), COORDINATE_Y, FuncList::execFunc(), FieldSize, FIELDSIZE_X, FIELDSIZE_Z, findAngleFromBallToGoal(), FuncList::firstFunc, KnowledgeBase::getBall(), Kicker::getComputerAxisPos(), KnowledgeBase::getKicker(), Ball::getPosAndVelo(), Config::getVar(), GOAL_POSITION_NEGATIVE_Z, GOAL_POSITION_POSITIVE_Z, GoalSize, moveAxisPTP(), moveAxis-

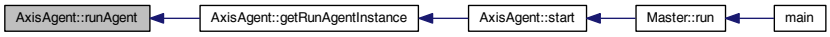
To(), moveFeetInCircularMotion(), mutexHandle, myStopped, funcElement::next, NULL, pConf, pfComputerTargetPos, pKB, POSITION, prepareShoot(), shoot(), shootBall(), TRAVERSE_DISTANCE_1, TRAVERSE_DISTANCE_2, TRAVERSE_DISTANCE_3, TRAVERSE_DISTANCE_5, VALUE, VELOCITY, VelocityMaxRot und VelocityMaxTrans.

Wird benutzt von getRunAgentInstance().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.1.3.7 void AxisAgent::shoot (float *fAxisRotAct*, float *fFromAngle* = -45.f, float *fToAngle* = +45.f) [private]

5.1.3.8 void AxisAgent::shoot (float *fAxisRotAct*, float *fFromAngle* = -45.f, float *fToAngle* = +45.f) [private]

Definiert in Zeile 594 der Datei AxisAgent.cpp.

Benutzt m_vecRotTargetPos.

Wird benutzt von runAgent().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.1.3.9 void AxisAgent::start (HANDLE & *threadHandle*)

Startet das System.

Parameter

out	*threadHandle*	Objekt des erzeugten Treads

5.1.3.10 void AxisAgent::start (HANDLE & *threadHandle*)

Startet das System.

Parameter

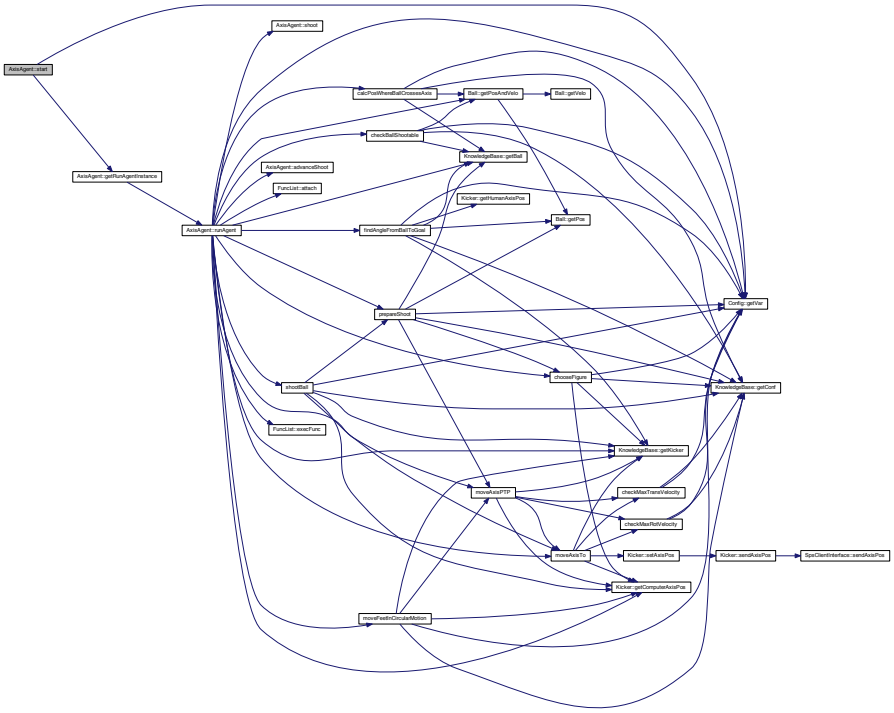
out	*threadHandle*	Objekt des erzeugten Treads

Definiert in Zeile 103 der Datei AxisAgent.cpp.

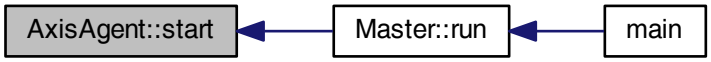
Benutzt afTargetPos, axis, AXIS_FOUR, AXIS_ONE, AxisShift, getRunAgentInstance(), Config::getVar(), mutexHandle, myStopped, NULL, pConf, VALUE, VELOCITY, VelocityMaxRot und VelocityMaxTrans.

Wird benutzt von Master::run().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.1.3.11 void AxisAgent::stop (void)

Stoppt das System.

5.1.3.12 void AxisAgent::stop (void)

Stoppt das System.

Definiert in Zeile 158 der Datei AxisAgent.cpp.

Benutzt mutexHandle und myStopped.

5.1.4 Dokumentation der Datenelemente

5.1.4.1 float AxisAgent::afTargetPos [private]

Zielposition und Geschwindigkeit.

Definiert in Zeile 196 der Datei AxisAgent.h.

Wird benutzt von start().

5.1.4.2 Axis AxisAgent::axis [private]

Spielstange, welche der Agent ansteuern soll.

Definiert in Zeile 169 der Datei AxisAgent.h.

Wird benutzt von AxisAgent(), runAgent(), start() und ~AxisAgent().

5.1.4.3 funcElement * AxisAgent::firstFunc [private]

Zugriff auf die SPS.

Siehe auch

Kicker

Erstes Listenelement der Funktionen

Definiert in Zeile 191 der Datei AxisAgent.h.

Wird benutzt von AxisAgent().

5.1.4.4 std::deque< float > AxisAgent::m_vecRotTargetPos [private]

Definiert in Zeile 242 der Datei AxisAgent.h.

Wird benutzt von advanceShoot() und shoot().

5.1.4.5 HANDLE AxisAgent::mutexHandle [private]

Mutex-Handle.

Noch zu erledigen Benutzung vom Mutex beschreiben

Mutex-Handle zum sicheren Datenzugriff auf myStopped von anderen Threads.

Noch zu erledigen Benutzung vom Mutex beschreiben

Mutex-Handle zum sicheren Datenzugriff auf myStopped von anderen Threads.

Definiert in Zeile 213 der Datei AxisAgent.h.

Wird benutzt von AxisAgent(), runAgent(), start(), stop() und ~AxisAgent().

5.1.4.6 bool AxisAgent::myStopped [private]

Wird auf true gesetzt wenn der Agent beendet werden soll.

Siehe auch

AxisAgent::~AxisAgent(void)
AxisAgent::runAgent(void)

Definiert in Zeile 220 der Datei AxisAgent.h.

Wird benutzt von AxisAgent(), runAgent(), start(), stop() und ~AxisAgent().

5.1.4.7 HANDLE AxisAgent::myThreadHandle [private]

Handle vom Thread.

Handle vom Thread welcher zum Ansteuern einer Spielstange zuständig ist. Wird in AxisAgent::start() beim Erzeugen des Threads gesetzt. Wird benutzt um im Destruktor auf das Ende des Threads zu warten.

Definiert in Zeile 205 der Datei AxisAgent.h.

Wird benutzt von ~AxisAgent().

5.1.4.8 Config * AxisAgent::pConf [private]

Zugriff auf die Konfigurationen.

Siehe auch

Config

Definiert in Zeile 180 der Datei AxisAgent.h.

Wird benutzt von AxisAgent(), runAgent() und start().

5.1.4.9 float AxisAgent::pfComputerTargetPos [private]

Definiert in Zeile 241 der Datei AxisAgent.h.

Wird benutzt von advanceShoot() und runAgent().

5.1.4.10 KnowledgeBase * AxisAgent::pKB [private]

Zeiger auf die Wissensbasis, zur Kommunikation.

Definiert in Zeile 174 der Datei AxisAgent.h.

Wird benutzt von AxisAgent() und runAgent().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

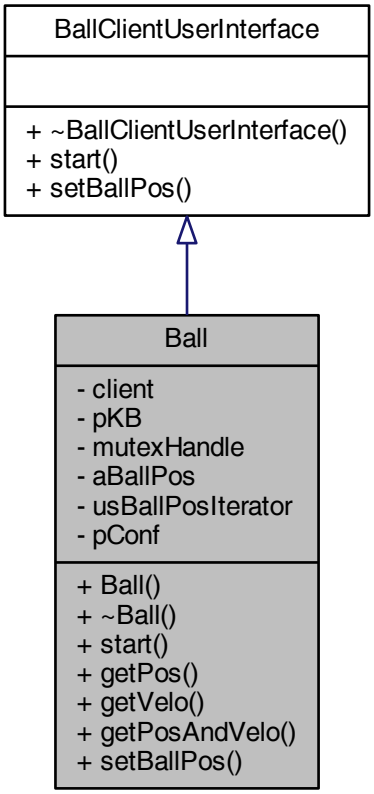
- AxisAgent.h
- AxisAgent_wobbeln.h
- AxisAgent.cpp
- AxisAgent_wobbeln.cpp

5.2 Ball Klassenreferenz

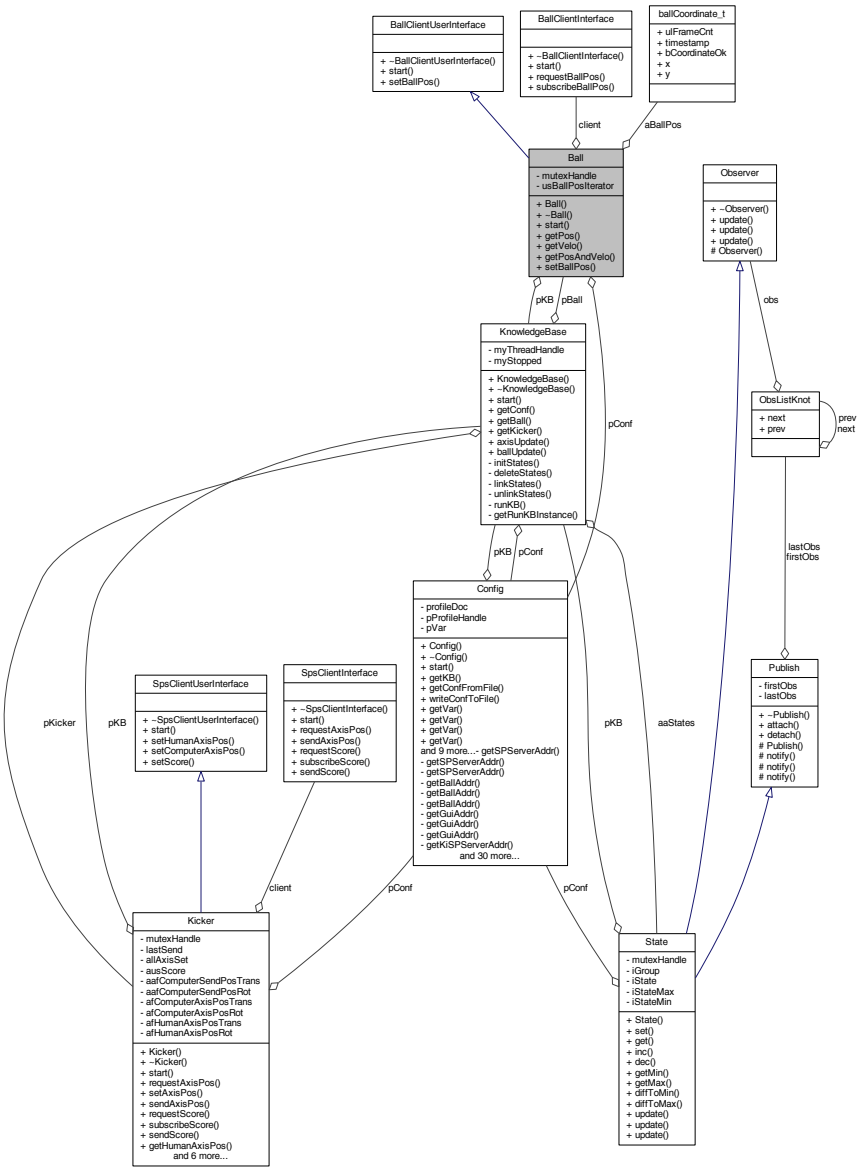
Zugriff zur Ballerkennung.

`#include <Ball.h>`

Klassendiagramm für Ball:



Zusammengehörigkeiten von Ball:



Öffentliche Methoden

- Ball (Config *pConfig)

  Konstruktor.
- ~Ball (void)

  Destruktor.
- void start (void)

  Startet die Benutzung des Clients.
- bool getPos (float &x, float &y)

  Bestimmt die Position des Balles.
- bool getVelo (float &xv, float &yv)

  Bestimmt die Geschwindigkeit des Balles.
- bool getPosAndVelo (float &x, float &y, float &xv, float &yv)

Bestimmt die Position des Balles.

- void setBallPos (ballCoordinate_t ballPos)

  setzt die Ballposition

Private Attribute

- BallClientInterface ∗ client

  Zeiger auf den Client.
- KnowledgeBase ∗ pKB

  Zeiger auf die Wissensbasis, zur Kommunikation.
- HANDLE mutexHandle

  Mutex-Handle.
- ballCoordinate_t aBallPos [8]

  Speicher für die letzten 8 Ballpositionen.
- unsigned short usBallPosIterator

  Iterator für die Ballpositionen.
- Config ∗ pConf

  Zugriff auf die Konfigurationen.

5.2.1 Ausführliche Beschreibung

Zugriff zur Ballerkennung.

Verwaltet die Ballposition und stellt sie für andere Klassen zur Verfügung.

Noch zu erledigen Exception-Handling.

Definiert in Zeile 29 der Datei Ball.h.

5.2.2 Beschreibung der Konstruktoren und Destruktoren

5.2.2.1 Ball::Ball (Config ∗ *pConfig*)

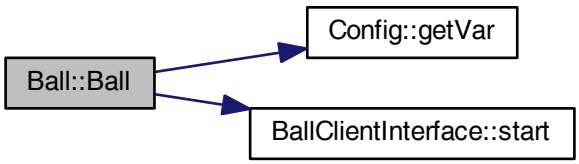
Konstruktor.

Stellt die Verbindung zur Ballerkennung her.

Definiert in Zeile 20 der Datei Ball.cpp.

Benutzt BallHost, BallPort, client, ERROR_BALL, ERROR_CONFIG, Config::getVar(), KiBallPort, mutexHandle, NULL, pConf, BallClientInterface::start() und usBallPosIterator.

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



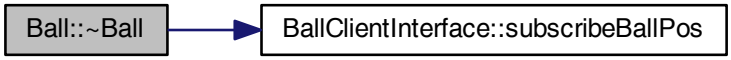
5.2.2.2 Ball::~Ball (void)

Destruktor.

Definiert in Zeile 50 der Datei Ball.cpp.

Benutzt client, mutexHandle und BallClientInterface::subscribeBallPos().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.2.3 Dokumentation der Elementfunktionen

5.2.3.1 bool Ball::getPos (float & *x*, float & *y*)

Bestimmt die Position des Balles.

Die Position wird in mm vom Ursprung (untel links von der Computerseite aus gesehen) angegeben.

Parameter

out	*x*	Referenz auf die X-Koordinate
out	*y*	Referenz auf die Y-Koordinate

Rückgabe

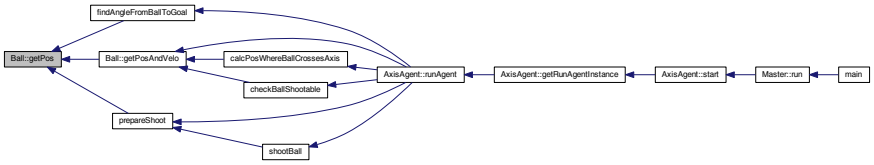
true bei Erfolg, ansonsten false

Definiert in Zeile 85 der Datei Ball.cpp.

Benutzt aBallPos, mutexHandle, usBallPosIterator, ballCoordinate_t::x und ballCoordinate_t::y.

Wird benutzt von findAngleFromBallToGoal(), getPosAndVelo() und prepareShoot().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.2.3.2 bool Ball::getPosAndVelo (float & *x*, float & *y*, float & *xv*, float & *yv*)

Bestimmt die Position des Balles.

Die Position wird in mm vom Ursprung (untel links von der Computerseite aus gesehen) angegeben.

Parameter

out	*x*	Referenz auf die X-Koordinate
out	*y*	Referenz auf die Y-Koordinate
out	*xv*	Referenz auf den X-Vektor
out	*yv*	Referenz auf den Y-Vektor

Rückgabe

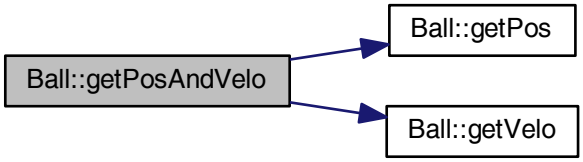
true bei Erfolg, ansonsten false

Definiert in Zeile 162 der Datei Ball.cpp.

Benutzt getPos() und getVelo().

Wird benutzt von calcPosWhereBallCrossesAxis(), checkBallShootable() und AxisAgent::runAgent().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.2.3.3 bool Ball::getVelo (float & *xv*, float & *yv*)

Bestimmt die Geschwindigkeit des Balles.

Die Geschwindigkeit wird in mm/ms angegeben.

Parameter

out	*xv*	Referenz auf die X-Koordinate
out	*yv*	Referenz auf die Y-Koordinate

Rückgabe

true bei Erfolg, ansonsten false

Definiert in Zeile 105 der Datei Ball.cpp.

Benutzt aBallPos, mutexHandle, ballCoordinate_t::timestamp und usBallPosIterator.

Wird benutzt von getPosAndVelo().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.2.3.4 void Ball::setBallPos (ballCoordinate_t *ballPos*) [virtual]

setzt die Ballposition

Parameter

in	*ballPos*	Ballposition die vom Client empfangen wurde

Wird vom Client aufgerufen wenn eine Ballposition vom Server eingetroffen ist.

Implementiert BallClientUserInterface.

Definiert in Zeile 167 der Datei Ball.cpp.

Benutzt aBallPos, KnowledgeBase::ballUpdate(), mutexHandle, pKB, usBallPosIterator, ballCoordinate_t::x und ballCoordinate_t::y.

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.2.3.5 void Ball::start (void) [virtual]

Startet die Benutzung des Clients.

Rückgabe

true wenn alles beim Starten des Clients geklappt hat

Implementiert BallClientUserInterface.

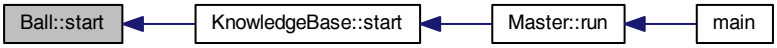
Definiert in Zeile 70 der Datei Ball.cpp.

Benutzt client und BallClientInterface::subscribeBallPos().

Wird benutzt von KnowledgeBase::start().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.2.4 Dokumentation der Datenelemente

5.2.4.1 ballCoordinate_t Ball::aBallPos[8] [private]

Speicher für die letzten 8 Ballpositionen.

Definiert in Zeile 125 der Datei Ball.h.

Wird benutzt von getPos(), getVelo() und setBallPos().

5.2.4.2 BallClientInterface* Ball::client [private]

Zeiger auf den Client.

Definiert in Zeile 106 der Datei Ball.h.

Wird benutzt von Ball(), start() und ~Ball().

5.2.4.3 HANDLE Ball::mutexHandle [private]

Mutex-Handle.

Noch zu erledigen Benutzung vom Mutex beschreiben

Mutex-Handle zum sicheren Datenzugriff da die Funktionen aus ClientUserInterface (ausser start()) von einem anderen Thread aufgerugen werden.

Definiert in Zeile 120 der Datei Ball.h.

Wird benutzt von Ball(), getPos(), getVelo(), setBallPos() und ~Ball().

5.2.4.4 Config* Ball::pConf [private]

Zugriff auf die Konfigurationen.

Siehe auch

Config

Definiert in Zeile 137 der Datei Ball.h.

Wird benutzt von Ball().

5.2.4.5 KnowledgeBase* Ball::pKB [private]

Zeiger auf die Wissensbasis, zur Kommunikation.

Definiert in Zeile 111 der Datei Ball.h.

Wird benutzt von setBallPos().

5.2.4.6 unsigned short Ball::usBallPosIterator [private]

Iterator für die Ballpositionen.

Siehe auch

myBallPos

Definiert in Zeile 131 der Datei Ball.h.

Wird benutzt von Ball(), getPos(), getVelo() und setBallPos().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

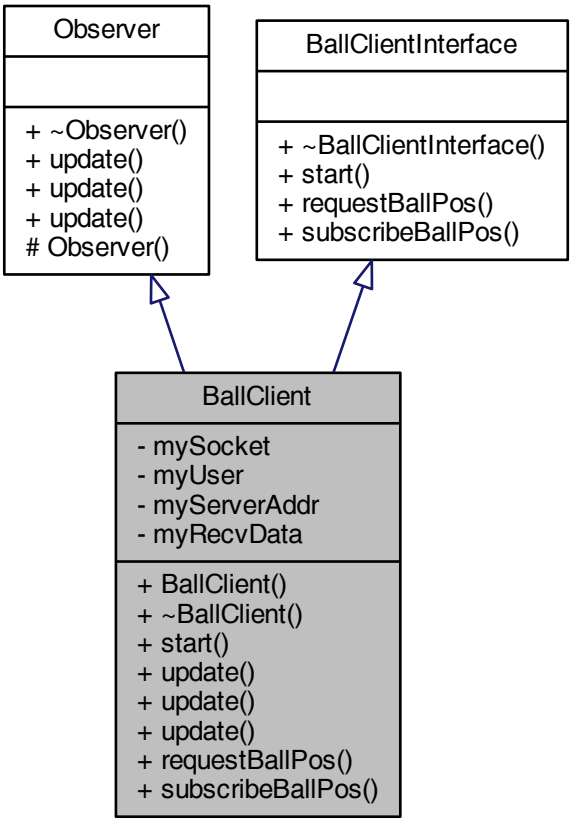
- Ball.h
- Ball.cpp

5.3 BallClient Klassenreferenz

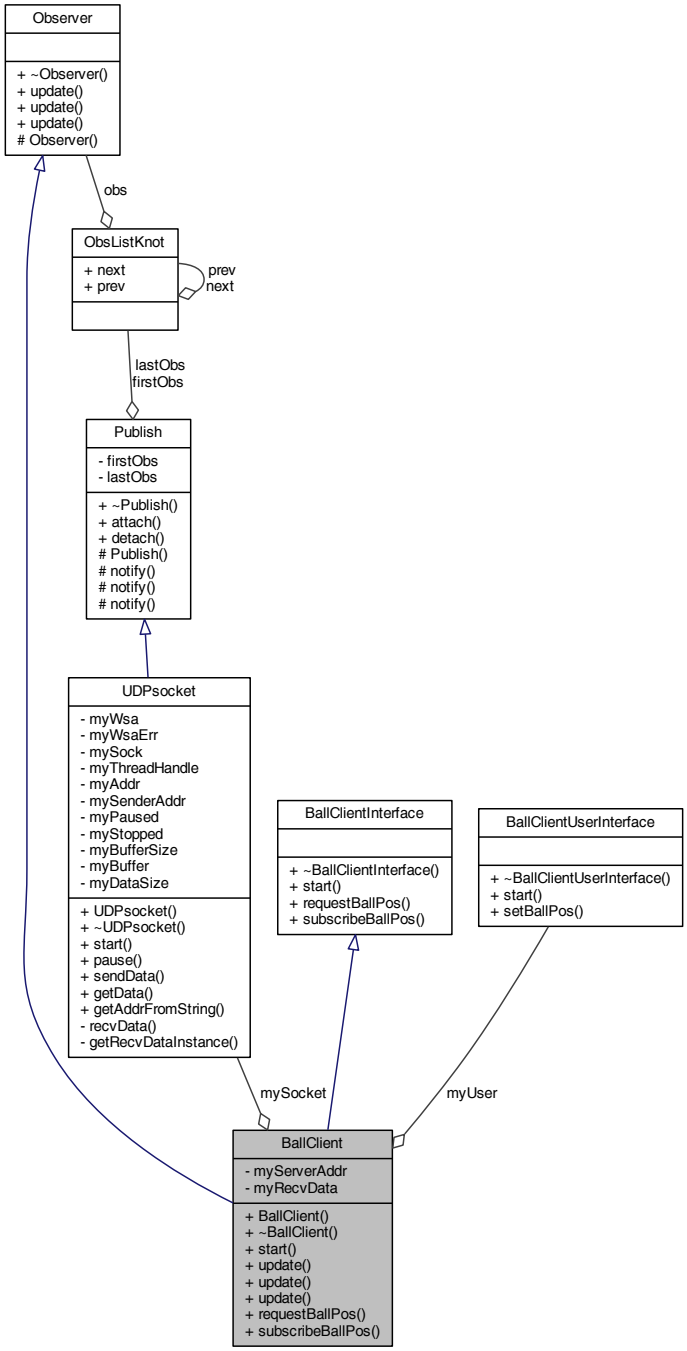
ein UDP-BallClient

```
#include <BallClient.h>
```

Klassendiagramm für BallClient:



Zusammengehörigkeiten von BallClient:



Öffentliche Methoden

- BallClient (unsigned short localPort, const char ∗serverHost, unsigned short serverPort)

  Konstruktor.
- ∼BallClient ()

  Destruktor.
- bool start (BallClientUserInterface ∗userInterface)

Startet den BallClient.

- void update (Publish ∗pub)

  update-Funktion der Observer Klasse
- void update (Publish ∗pub, int ∗iValues, unsigned int iValuesSize)

  update-Funktion der Observer Klasse
- void update (Publish ∗pub, float ∗fValues, unsigned int fValuesSize, int ∗iValues=NULL, unsigned int iValues-Size=0)

  update-Funktion der Observer Klasse
- bool requestBallPos ()

  Fragt die Ballposition beim Server an.
- bool subscribeBallPos (bool subs=true)

  Abonniert die Ballposition beim Server.

Private Attribute

- UDPsocket ∗ mySocket

  der UDP-Socket
- BallClientUserInterface ∗ myUser

  Zeiger auf den Benutzer dieser Klasse.
- sockaddr_in myServerAddr

  Adresse des Servers.
- char ∗ myRecvData

  Speicher für die empfangenen Daten.

5.3.1 Ausführliche Beschreibung

ein UDP-BallClient

Beispielimplementierung eines UDP-BallClient unter Verwendung der Klasse UDPsocket.

Definiert in Zeile 26 der Datei BallClient.h.

5.3.2 Beschreibung der Konstruktoren und Destruktoren

5.3.2.1 BallClient::BallClient (unsigned short *localPort*, const char ∗ *serverHost*, unsigned short *serverPort*)

Konstruktor.

Parameter

in	*localPort*	eigenen IP-Port
in	*serverHost*	Rechnername oder IP-Adresse des Servers
in	*serverPort*	Port des Servers

Ermittelt die Server-Adresse und legt den Empfangsspeicher und die Klasse UDPsocket an.

Definiert in Zeile 19 der Datei BallClient.cpp.

Benutzt BUFFER_SIZE, UDPsocket::getAddrFromString(), myServerAddr, mySocket und NULL.

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.3.2.2 BallClient::∼BallClient ( )

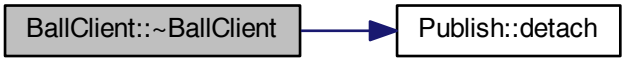
Destruktor.

Schließt den UDP-Socket und gibt den Empfangsspeicher wieder frei.

Definiert in Zeile 35 der Datei BallClient.cpp.

Benutzt Publish::detach(), mySocket, myUser und NULL.

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.3.3 Dokumentation der Elementfunktionen

5.3.3.1 bool BallClient::requestBallPos ( ) [virtual]

Fragt die Ballposition beim Server an.

Rückgabe

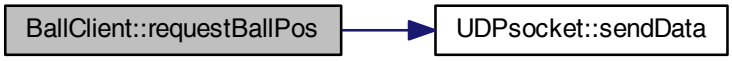
true wenn die Anfrage gesendet wurde

Implementiert BallClientInterface.

Definiert in Zeile 118 der Datei BallClient.cpp.

Benutzt myServerAddr, mySocket, REQUEST_BALLPOS und UDPsocket::sendData().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.3.3.2 bool BallClient::start (BallClientUserInterface * *userInterface*) [virtual]

Startet den BallClient.

Parameter

in	*userInterface*	Zeiger auf den Benutzer des BallClients

Rückgabe

true wenn alles beim Starten des BallClients geklappt hat

Fügt den BallClient als Abonnent zum Socket hinzu und startet den Socket.

Implementiert BallClientInterface.

Definiert in Zeile 46 der Datei BallClient.cpp.

Benutzt Publish::attach(), mySocket, myUser und UDPsocket::start().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.3.3.3 bool BallClient::subscribeBallPos (bool *subs* = true) [virtual]

Abonniert die Ballposition beim Server.

Parameter

in	*subs*	true zum Abonnieren, false zum Kündigen

Rückgabe

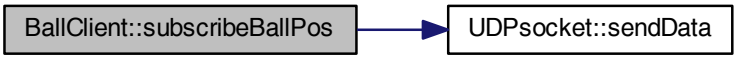
true wenn die Anfrage gesendet wurde

Implementiert BallClientInterface.

Definiert in Zeile 135 der Datei BallClient.cpp.

Benutzt bool_1::bl, bool_1::id, myServerAddr, mySocket, UDPsocket::sendData() und SUBSCRIBE_BALLPOS.

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.3.3.4 void BallClient::update (Publish * *pub*) [virtual]

update-Funktion der Observer Klasse

Parameter

in	*pub*	Publisher Objekt, welches über ein Ereignis benachichtigt (hier UDPsocket)

Wird von UDPsocket aufgerufen wenn Daten eingetroffen sind.

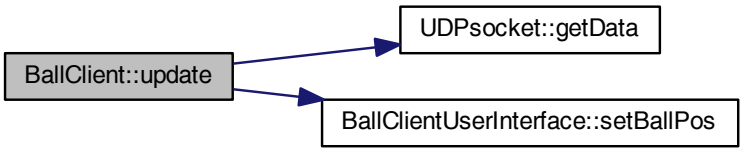
Implementiert Observer.

Definiert in Zeile 63 der Datei BallClient.cpp.

Benutzt ballPacket::ballPos, GET_BALLPOS, UDPsocket::getData(), myRecvData, mySocket, myUser und BallClientUserInterface::setBallPos().

Wird benutzt von update().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.3.3.5 void BallClient::update (Publish * *pub*, int * *iValues*, unsigned int *iValuesSize*) [virtual]

update-Funktion der Observer Klasse

Parameter

in	*pub*	Publisher Objekt, welches über ein Ereignis benachichtigt (hier UDPsocket)
in	*iValues*	wird ignoriert
in	*iValuesSize*	wird ignoriert

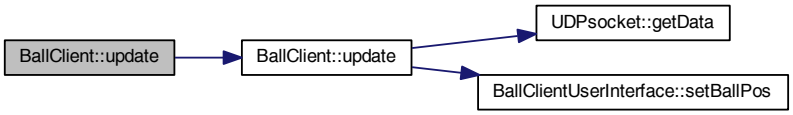
Wird von UDPsocket aufgerufen wenn Daten eingetroffen sind.

Implementiert Observer.

Definiert in Zeile 102 der Datei BallClient.cpp.

Benutzt update().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.3.3.6 void BallClient::update (Publish * *pub*, float * *fValues*, unsigned int *fValuesSize*, int * *iValues* = NULL, unsigned int *iValuesSize* = 0) [virtual]

update-Funktion der Observer Klasse

Parameter

in	*pub*	Publisher Objekt, welches über ein Ereignis benachichtigt (hier UDPsocket)
in	*fValues*	wird ignoriert
in	*fValuesSize*	wird ignoriert
in	*iValues*	wird ignoriert
in	*iValuesSize*	wird ignoriert

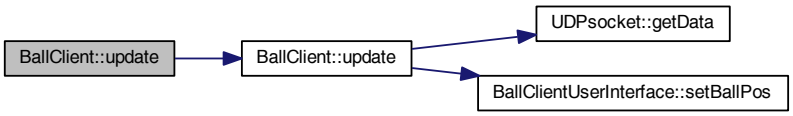
Wird von UDPsocket aufgerufen wenn Daten eingetroffen sind.

Implementiert Observer.

Definiert in Zeile 107 der Datei BallClient.cpp.

Benutzt update().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.3.4 Dokumentation der Datenelemente

5.3.4.1 char* BallClient::myRecvData [private]

Speicher für die empfangenen Daten.

Definiert in Zeile 127 der Datei BallClient.h.

Wird benutzt von update().

5.3.4.2 sockaddr_in BallClient::myServerAddr [private]

Adresse des Servers.

Definiert in Zeile 122 der Datei BallClient.h.

Wird benutzt von BallClient(), requestBallPos() und subscribeBallPos().

5.3.4.3 UDPsocket* BallClient::mySocket [private]

der UDP-Socket

Definiert in Zeile 112 der Datei BallClient.h.

Wird benutzt von BallClient(), requestBallPos(), start(), subscribeBallPos(), update() und ~BallClient().

5.3.4.4 BallClientUserInterface* BallClient::myUser [private]

Zeiger auf den Benutzer dieser Klasse.

Definiert in Zeile 117 der Datei BallClient.h.

Wird benutzt von start(), update() und ~BallClient().

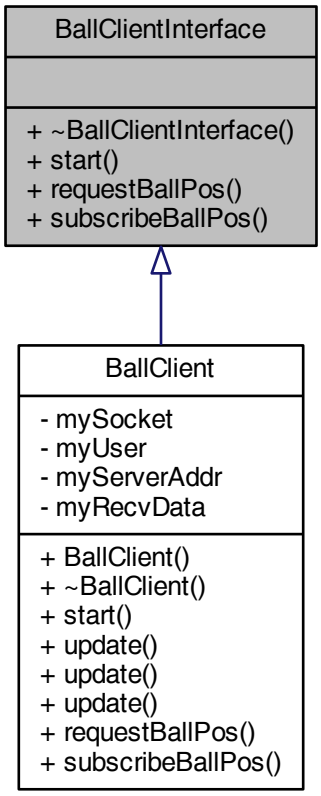
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- BallClient.h
- BallClient.cpp

5.4 BallClientInterface Klassenreferenz

```
#include <BallClientInterface.h>
```

Klassendiagramm für BallClientInterface:



Öffentliche Methoden

- virtual ~BallClientInterface (void)

  Virtueller Destruktor.
- virtual bool start (BallClientUserInterface *userInterface)=0

  Startet den Client.
- virtual bool requestBallPos ()=0

  Fragt die Ballposition beim Server an.
- virtual bool subscribeBallPos (bool subs=true)=0

  Abonniert die Ballposition beim Server.

5.4.1 Ausführliche Beschreibung

Definiert in Zeile 18 der Datei BallClientInterface.h.

5.4.2 Beschreibung der Konstruktoren und Destruktoren

5.4.2.1 virtual BallClientInterface::~BallClientInterface (void) [inline, virtual]

Virtueller Destruktor.

Definiert in Zeile 24 der Datei BallClientInterface.h.

5.4.3 Dokumentation der Elementfunktionen

5.4.3.1 virtual bool BallClientInterface::requestBallPos ( ) [pure virtual]

Fragt die Ballposition beim Server an.

Rückgabe

true wenn die Anfrage gesendet wurde

Implementiert in BallClient.

5.4.3.2 virtual bool BallClientInterface::start (BallClientUserInterface * *userInterface*) [pure virtual]

Startet den Client.

Parameter

in	*userInterface*	Zeiger auf den Benutzer des Clients

Rückgabe

true wenn alles beim Starten des Clients geklappt hat

Implementiert in BallClient.

Wird benutzt von Ball::Ball().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.4.3.3 virtual bool BallClientInterface::subscribeBallPos (bool *subs* = true) [pure virtual]

Abonniert die Ballposition beim Server.

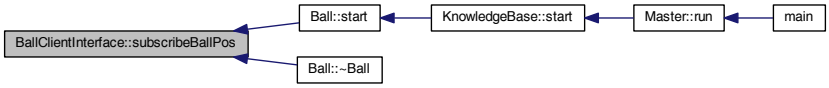
Rückgabe

true wenn die Anfrage gesendet wurde

Implementiert in BallClient.

Wird benutzt von Ball::start() und Ball::~Ball().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

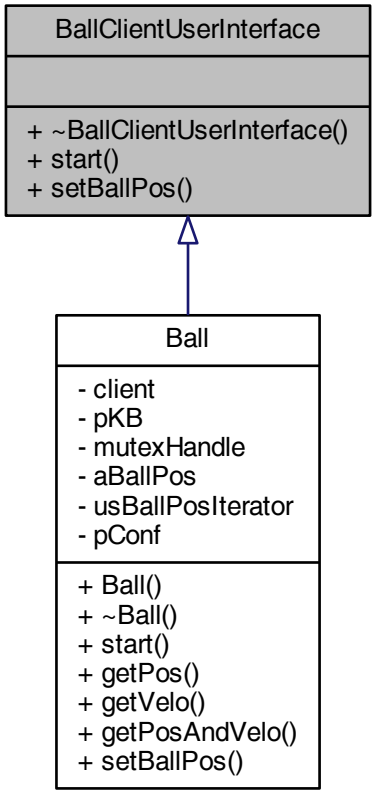
- BallClientInterface.h

5.5 BallClientUserInterface Schnittstellenreferenz

Interface zur Kommunikation mit dem Benutzer des Ball-Clients.

```
#include <BallClientUserInterface.h>
```

Klassendiagramm für BallClientUserInterface:



Öffentliche Methoden

- virtual ~BallClientUserInterface (void)

  Virtueller Destruktor.
- virtual void start ()=0

  Startet die Benutzung des Clients.
- virtual void setBallPos (ballCoordinate_t ballPos)=0

  setzt die Ballposition

5.5.1 Ausführliche Beschreibung

Interface zur Kommunikation mit dem Benutzer des Ball-Clients.

Definiert in Zeile 18 der Datei BallClientUserInterface.h.

5.5.2 Beschreibung der Konstruktoren und Destruktoren

5.5.2.1 virtual BallClientUserInterface::~BallClientUserInterface (void) [inline, virtual]

Virtueller Destruktor.

Definiert in Zeile 24 der Datei BallClientUserInterface.h.

5.5.3 Dokumentation der Elementfunktionen

5.5.3.1 virtual void BallClientUserInterface::setBallPos (ballCoordinate_t *ballPos*) [pure virtual]

setzt die Ballposition

Parameter

in	*ballPos*	Ballposition die vom Client empfangen wurde

Wird vom Client aufgerufen wenn eine Ballposition vom Server eingetroffen ist.

Implementiert in Ball.

Wird benutzt von BallClient::update().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.5.3.2 virtual void BallClientUserInterface::start ( ) [pure virtual]

Startet die Benutzung des Clients.

Rückgabe

true wenn alles beim Starten des Clients geklappt hat

Implementiert in Ball.

Die Dokumentation für diese Schnittstelle wurde erzeugt aufgrund der Datei:

- BallClientUserInterface.h

5.6 ballCoordinate_t Strukturreferenz

Struktur der Ballposition.

```
#include <UdpDefines.h>
```

Öffentliche Attribute

- unsigned long ulFrameCnt
- double timestamp
- bool bCoordinateOk
- float x
- float y

5.6.1 Ausführliche Beschreibung

Struktur der Ballposition.

Definiert in Zeile 214 der Datei UdpDefines.h.

5.6.2 Dokumentation der Datenelemente

5.6.2.1 bool ballCoordinate_t::bCoordinateOk

True wenn Ballposition erfolgreich bestimmt weden konnte

Definiert in Zeile 221 der Datei UdpDefines.h.

5.6.2.2 double ballCoordinate_t::timestamp

Zeitstempel der linken Kamera

Definiert in Zeile 219 der Datei UdpDefines.h.

Wird benutzt von Ball::getVelo().

5.6.2.3 unsigned long ballCoordinate_t::ulFrameCnt

Bildpaare seit Start

Definiert in Zeile 217 der Datei UdpDefines.h.

5.6.2.4 float ballCoordinate_t::x

Ballposition in cm (0,0 - 120,7)

Definiert in Zeile 223 der Datei UdpDefines.h.

Wird benutzt von Ball::getPos() und Ball::setBallPos().

5.6.2.5 float ballCoordinate_t::y

Ballposition in cm (0,0 - 68,5)

Definiert in Zeile 225 der Datei UdpDefines.h.

Wird benutzt von Ball::getPos() und Ball::setBallPos().

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

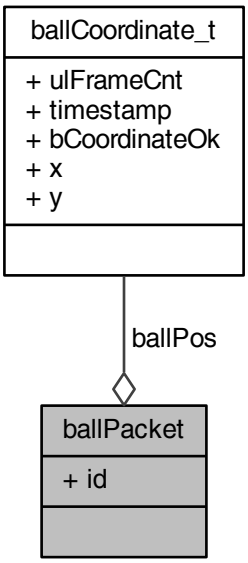
- UdpDefines.h

5.7 ballPacket Strukturreferenz

Paket mit der Ballposition.

`#include <UdpDefines.h>`

Zusammengehörigkeiten von ballPacket:



Öffentliche Attribute

- unsigned int id: 16
- ballCoordinate_t ballPos

5.7.1 Ausführliche Beschreibung

Paket mit der Ballposition.

Beinhaltet die Paket-ID und eine Struktur mit der Ballposition

Definiert in Zeile 233 der Datei UdpDefines.h.

5.7.2 Dokumentation der Datenelemente

5.7.2.1 ballCoordinate_t ballPacket::ballPos

Definiert in Zeile 236 der Datei UdpDefines.h.

Wird benutzt von BallClient::update().

5.7.2.2 unsigned int ballPacket::id

Definiert in Zeile 235 der Datei UdpDefines.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- UdpDefines.h

5.8 ubf16_request::bitfeld_16 Strukturreferenz

```
#include <UdpDefines.h>
```

Öffentliche Attribute

- unsigned int humanAxisPos_00: 1
- unsigned int computerAxisPos_01: 1
- unsigned int bothAxisPos_02: 1
- unsigned int computerAxisRamp_03: 1
- unsigned int computerAxisStackmode_04: 1
- unsigned int ballSqueeze_05: 1
- unsigned int score_06: 1
- unsigned int safety_07: 1
- unsigned int reserved_08: 1
- unsigned int reserved_09: 1
- unsigned int reserved_10: 1
- unsigned int reserved_11: 1
- unsigned int reserved_12: 1
- unsigned int reserved_13: 1
- unsigned int reserved_14: 1
- unsigned int reserved_15: 1

5.8.1 Ausführliche Beschreibung

Definiert in Zeile 102 der Datei UdpDefines.h.

5.8.2 Dokumentation der Datenelemente

5.8.2.1 unsigned int ubf16_request::bitfeld_16::ballSqueeze_05

Definiert in Zeile 109 der Datei UdpDefines.h.

5.8.2.2 unsigned int ubf16_request::bitfeld_16::bothAxisPos_02

Definiert in Zeile 106 der Datei UdpDefines.h.

Wird benutzt von SpsClient::requestAxisPos().

5.8.2.3 unsigned int ubf16_request::bitfeld_16::computerAxisPos_01

Definiert in Zeile 105 der Datei UdpDefines.h.

5.8.2.4 unsigned int ubf16_request::bitfeld_16::computerAxisRamp_03

Definiert in Zeile 107 der Datei UdpDefines.h.

5.8.2.5 unsigned int ubf16_request::bitfeld_16::computerAxisStackmode_04

Definiert in Zeile 108 der Datei UdpDefines.h.

5.8.2.6 unsigned int ubf16_request::bitfeld_16::humanAxisPos_00

Definiert in Zeile 104 der Datei UdpDefines.h.

5.8.2.7 unsigned int ubf16_request::bitfeld_16::reserved_08

Definiert in Zeile 112 der Datei UdpDefines.h.

5.8.2.8 unsigned int ubf16_request::bitfeld_16::reserved_09

Definiert in Zeile 113 der Datei UdpDefines.h.

5.8.2.9 unsigned int ubf16_request::bitfeld_16::reserved_10

Definiert in Zeile 114 der Datei UdpDefines.h.

5.8.2.10 unsigned int ubf16_request::bitfeld_16::reserved_11

Definiert in Zeile 115 der Datei UdpDefines.h.

5.8.2.11 unsigned int ubf16_request::bitfeld_16::reserved_12

Definiert in Zeile 116 der Datei UdpDefines.h.

5.8.2.12 unsigned int ubf16_request::bitfeld_16::reserved_13

Definiert in Zeile 117 der Datei UdpDefines.h.

5.8.2.13 unsigned int ubf16_request::bitfeld_16::reserved_14

Definiert in Zeile 118 der Datei UdpDefines.h.

5.8.2.14 unsigned int ubf16_request::bitfeld_16::reserved_15

Definiert in Zeile 119 der Datei UdpDefines.h.

5.8.2.15 unsigned int ubf16_request::bitfeld_16::safety_07

Definiert in Zeile 111 der Datei UdpDefines.h.

5.8.2.16 unsigned int ubf16_request::bitfeld_16::score_06

Definiert in Zeile 110 der Datei UdpDefines.h.

Wird benutzt von SpsClient::requestScore().

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- UdpDefines.h

5.9 bool_1 Strukturreferenz

Anfrage-Paket für bestimmte Daten.

```
#include <UdpDefines.h>
```

Öffentliche Attribute

- unsigned int id: 16
- bool bl

5.9.1 Ausführliche Beschreibung

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein bool-Wert.

Definiert in Zeile 139 der Datei UdpDefines.h.

5.9.2 Dokumentation der Datenelemente

5.9.2.1 bool bool_1::bl

Definiert in Zeile 142 der Datei UdpDefines.h.

Wird benutzt von BallClient::subscribeBallPos() und SpsClient::subscribeScore().

5.9.2.2 unsigned int bool_1::id

Definiert in Zeile 141 der Datei UdpDefines.h.

Wird benutzt von BallClient::subscribeBallPos() und SpsClient::subscribeScore().

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- UdpDefines.h

5.10 bool_2 Strukturreferenz

Anfrage-Paket für bestimmte Daten.

```
#include <UdpDefines.h>
```

Öffentliche Attribute

- unsigned int id: 16
- bool bl [2]

5.10.1 Ausführliche Beschreibung

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und zwei bool-Werte.

Definiert in Zeile 150 der Datei UdpDefines.h.

5.10.2 Dokumentation der Datenelemente

5.10.2.1 bool bool_2::bl[2]

Definiert in Zeile 153 der Datei UdpDefines.h.

5.10.2.2 unsigned int bool_2::id

Definiert in Zeile 152 der Datei UdpDefines.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- UdpDefines.h

5.11 ClientInterface Schnittstellenreferenz

Interface zur Kommunikation mit dem UDP-Client.

```
#include <BallClientInterface.h>
```

5.11.1 Ausführliche Beschreibung

Interface zur Kommunikation mit dem UDP-Client.

Definiert in Zeile 12 der Datei BallClientInterface.h.

Die Dokumentation für diese Schnittstelle wurde erzeugt aufgrund der Datei:

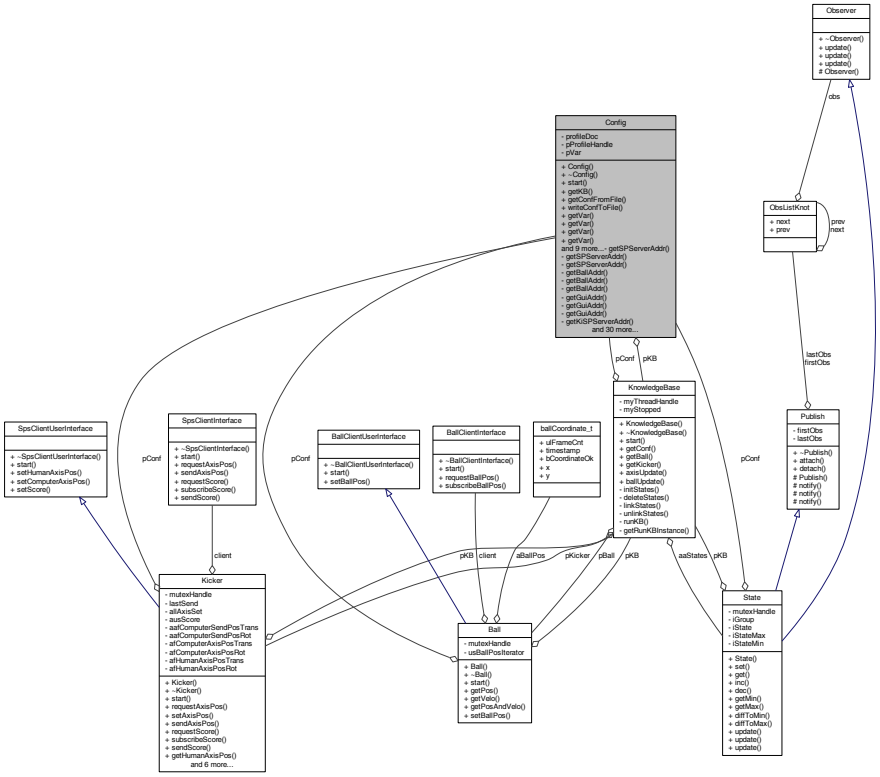
- BallClientInterface.h

5.12 Config Klassenreferenz

Configurationsverwaltung.

```
#include <Config.h>
```

Zusammengehörigkeiten von Config:



Öffentliche Methoden

- Config (KnowledgeBase *kb)

  Konstruktor.
- ~Config (void)

  Destruktor.
- void start (void)

  Lädt die Konfiguration aus den entsprechenden Dateien.
- KnowledgeBase * getKB (void)

  Gibt den Zeiger auf die Wissensbasis zurück.
- bool getConfFromFile (ConfigVar var=All)

  Liest die Konfiguration aus den Dateien.
- bool writeConfToFile (ConfigVar var=All)

  Schreibt die Konfiguration in die Dateien.
- bool getVar (ConfigVar var, unsigned short &usVal)

  Gibt den Wert von var zurück.
- bool getVar (ConfigVar var, unsigned short(&ausVal)[2])

Gibt den Wert von var zurück.

- bool getVar (ConfigVar var, char ∗&pcVal)

  Gibt den Wert von var zurück.
- bool getVar (ConfigVar var, float &fVal)

  Gibt den Wert von var zurück.
- bool getVar (ConfigVar var, float(&afVal)[2])

  Gibt den Wert von var zurück.
- bool getVar (ConfigVar var, float(&afVal)[4])

  Gibt den Wert von var zurück.
- bool getVar (ConfigVar var, float(&afVal)[4][5])

  Gibt den Wert von var zurück.
- bool getVar (ConfigVar var, bool(&abVal)[4])

  Gibt den Wert von var zurück.
- bool getVar (ConfigVar var, int &iVal)

  Gibt den Wert von var zurück.
- bool getVar (ConfigVar var, int iArraySize1, int iArraySize2, float ∗∗(&afVal))

  Gibt den Wert von var zurück.
- bool getState (unsigned int group, unsigned int state, int &val)

  Liefert den Wert eines Statusses.
- bool getState (unsigned int group, unsigned int state, int(&val)[3])

  Liefert die Werte eines Statusses.
- bool getStatesInGroup (unsigned int group, unsigned int &states)

  Liefert die Anzahl der Statusse der Gruppe zurück.
- bool setState (unsigned int group, unsigned int state, int val)

  Setzt den Wert eines Statusses.

Private Methoden

- bool getSPServerAddr (unsigned short &port)

  Liest den Port des TCP-Servers aus der Konfigurationsdatei.
- bool getSPServerAddr (char ∗&host)

  Liest die Adresse des SPServer aus der Konfigurationsdatei.
- bool getSPServerAddr (unsigned short &port, char ∗&host)

  Liest die Adresse des SPServer aus der Konfigurationsdatei.
- bool getBallAddr (unsigned short &port)

  Liest die Adresse der Ballerkennung aus der Konfigurationsdatei.
- bool getBallAddr (char ∗&host)

  Liest die Adresse der Ballerkennung aus der Konfigurationsdatei.
- bool getBallAddr (unsigned short &port, char ∗&host)

  Liest die Adresse der Ballerkennung aus der Konfigurationsdatei.
- bool getGuiAddr (unsigned short &port)

  Liest die Adresse der GUI aus der Konfigurationsdatei.
- bool getGuiAddr (char ∗&host)

  Liest die Adresse der GUI aus der Konfigurationsdatei.
- bool getGuiAddr (unsigned short &port, char ∗&host)

  Liest die Adresse der GUI aus der Konfigurationsdatei.
- bool getKiSPServerAddr (unsigned short &port)

  Liest die Adresse der Ki für den SPServer aus der Konfigurationsdatei.
- bool getKiSPServerAddr (char ∗&host)

  Liest die Adresse der Ki für den SPServer aus der Konfigurationsdatei.

- bool getKiSPServerAddr (unsigned short &port, char ∗&host)

  Liest die Adresse der für den SPServer Ki aus der Konfigurationsdatei.
- bool getKiBallAddr (unsigned short &port)

  Liest die Adresse der Ki für die Ballerkennung aus der Konfigurationsdatei.
- bool getKiBallAddr (char ∗&host)

  Liest die Adresse der Ki für die Ballerkennung aus der Konfigurationsdatei.
- bool getKiBallAddr (unsigned short &port, char ∗&host)

  Liest die Adresse der für die Ballerkennung Ki aus der Konfigurationsdatei.
- bool getKiGuiAddr (unsigned short &port)

  Liest die Adresse der Ki für die GUI aus der Konfigurationsdatei.
- bool getKiGuiAddr (char ∗&host)

  Liest die Adresse der Ki für die GUI aus der Konfigurationsdatei.
- bool getKiGuiAddr (unsigned short &port, char ∗&host)

  Liest die Adresse der Ki aus der Konfigurationsdatei.
- bool getAxisShift (const unsigned int axis, float &shift)

  Liest den Fahrweg einer Spielstange aus der Konfigurationsdatei.
- bool getAxisPosX (const unsigned int axis, float &pos)

  Liest X-Position einer Spielstange aus der Konfigurationsdatei.
- bool getAxisOffset (float &offset)

  Liest den Abstand, den die Spielstangen von den Grenzwerten einhalten sollen aus der Konfigurationsdatei.
- bool getAxisDistance (float &distance)

  Liest den Abstand zwischen den einzelnen Spielstangen aus der Konfigurationsdatei.
- bool getFigurePos (const unsigned int axis, const unsigned int figure, float &pos)

  Liest die Position der Spielfigur an der Spielstange aus der Konfigurationsdatei.
- bool getFieldSize (float &depth, float &breadth)

  Liest die Spielfeldgröße aus der Konfigurationsdatei.
- bool getGoalSize (float &width)

  Liest die Torgröße aus der Konfigurationsdatei.
- bool getVelocityMaxTrans (float &transVelo)

  Liest die maxinale Geschwindigkeit der Translation aus der Konfigurationsdatei.
- bool getVelocityMaxRot (float &rotVelo)

  Liest die maxinale Geschwindigkeit der Rotation aus der Konfigurationsdatei.
- bool getFigureFootBreadth (float &breadth)

  Liest die Breite des Fußes der Spielfigur aus der Konfigurationsdatei.
- bool getFigureFootHeight (float &height)

  Liest die Höhe des Fußes der Spielfigur aus der Konfigurationsdatei.
- bool getFigureFootThickness (float &thickness)

  Liest die Dicke des Fußes der Spielfigur aus der Konfigurationsdatei.
- bool getFigureFieldDistance (float &distance)

  Liest den Abstand des Fußes der Spielfigur zum Spielfeld aus der Konfigurationsdatei.
- bool getBallDiameter (float &diameter)

  Liest den Durchmesser des Balles aus der Konfigurationsdatei.
- bool getBallToAxisShootable (float &distance)

  Liest die Distanz des Balles zur Spielstange, aus der der Ball von der Figur noch geschossen werden kann aus der Konfigurationsdatei.
- bool getBallToAxisReachable (float &distance)

  Liest die Distanz des Balles zur Spielstange, aus der der Ball von der Figur noch erreicht werden kann aus der Konfigurationsdatei.
- bool getConnectionInfo (const char ∗service, const char ∗attr, char ∗&addr)

  Liest Werte aus der Konfigurationsdatei, die für verschiedene Verbindungen nötig sind.
- bool getConnectionInfo (const char ∗service, unsigned short &port)

Liest Port aus der Konfigurationsdatei, über den die Verbindung läuft.

- bool getDimension (const char ∗child, const char ∗attr, float &val)

  Liest Werte aus der Konfigurationsdatei, die sich auf die Dimensionierung beziehen.
- bool getAxisVal (const unsigned int axis, const char ∗child1, const char ∗child2, const char ∗attr, float &val)

  Liest Werte aus der Konfigurationsdatei, die sich auf eine Spielstange beziehen.
- bool getFigureVal (const unsigned int axis, const unsigned int figure, const char ∗attr, float &val)

  Liest Werte aus der Konfigurationsdatei, die sich auf eine Spielfigur beziehen.
- bool getStateInfo (unsigned int group, unsigned int state, int ∗&vals)

  Liest Werte aus der Konfigurationsdatei, die für die Statusse benötigt werden.
- bool getShootAngleInfo (unsigned int shoot, unsigned int num, float ∗&val)

  Liest Werte aus der Konfigurationsdatei, die sich auf die Schusswinkel beziehen.

Private Attribute

- KnowledgeBase ∗ pKB

  Zeiger auf die Wissensbasis.
- TiXmlDocument profileDoc

  Die Datei, die die Grundkonfiguration beinhaltet.
- TiXmlHandle ∗ pProfileHandle

  Handle auf die Profil-Kongigurationsdatei.
- void ∗ pVar [All]

  Array mit den Zeigern auf die einzelnen Variablen.

5.12.1 Ausführliche Beschreibung

Configurationsverwaltung.

Verwaltet alle Konfigurationen in XML-Dateien und stellt sie für andere Klassen zur Verfügung.

Noch zu erledigen writeConfToFile() und Werte zuweisen umsetzen.
Exception-Handling.

Definiert in Zeile 26 der Datei Config.h.

5.12.2 Beschreibung der Konstruktoren und Destruktoren

5.12.2.1 Config::Config (KnowledgeBase ∗ *kb*)

Konstruktor.

Parameter

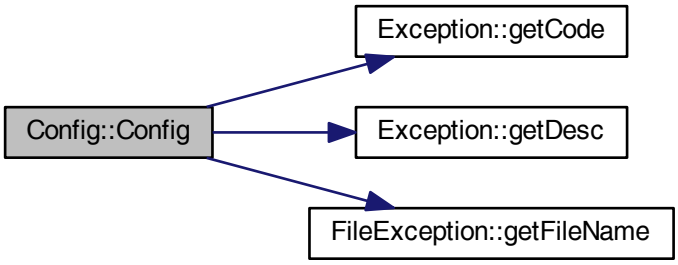
in	*kb*	Zeiger auf die Wissensbasis, zur Kommunikation mit dieser

Öffnet die Konfigurationsdatei und parst die Standardkonfiguration.

Definiert in Zeile 17 der Datei Config.cpp.

Benutzt All, BUFFER_SIZE, Exception::getCode(), Exception::getDesc(), FileException::getFileName(), NULL, pProfileHandle, profileDoc und pVar.

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.12.2.2 Config::~Config (void)

Destruktor.

Noch zu erledigen pVar löschen!

Definiert in Zeile 70 der Datei Config.cpp.

5.12.3 Dokumentation der Elementfunktionen

5.12.3.1 bool Config::getAxisDistance (float & *distance*) [private]

Liest den Abstand zwischen den einzelnen Spielstangen aus der Konfigurationsdatei.

Parameter

out	*distance*	Abstand

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 799 der Datei Config.cpp.

Benutzt NULL und pProfileHandle.

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.2 bool Config::getAxisOffset (float & *offset*) [private]

Liest den Abstand, den die Spielstangen von den Grenzwerten einhalten sollen aus der Konfigurationsdatei.

Parameter

out	*offset*	Abstand

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 775 der Datei Config.cpp.

Benutzt NULL und pProfileHandle.

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.3 bool Config::getAxisPosX (const unsigned int *axis*, float & *pos*) [private]

Liest X-Position einer Spielstange aus der Konfigurationsdatei.

Parameter

in	*axis*	Spielstange
out	*pos*	Position

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 767 der Datei Config.cpp.

Benutzt getAxisVal().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.4 bool Config::getAxisShift (const unsigned int *axis*, float & *shift*) [private]

Liest den Fahrweg einer Spielstange aus der Konfigurationsdatei.

Parameter

in	*axis*	Spielstange
out	*shift*	Fahrweg

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 759 der Datei Config.cpp.

Benutzt getAxisVal().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.5 bool Config::getAxisVal (const unsigned int *axis*, const char * *child1*, const char * *child2*, const char * *attr*, float & *val*) [private]

Liest Werte aus der Konfigurationsdatei, die sich auf eine Spielstange beziehen.

Parameter

in	*axis*	Spielstange
in	*child1*	Art der Eigenschaft (dimensions oder startpos)
in	*child2*	Eigenschaft (shift, pos, velo, ...)
in	*attr*	Das Attribut was zurückgegeben werden soll
out	*val*	Der Wert

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 1111 der Datei Config.cpp.

Benutzt AXIS_FOUR, AXIS_ONE, NULL und pProfileHandle.

Wird benutzt von getAxisPosX() und getAxisShift().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.6 bool Config::getBallAddr (unsigned short & *port*) [private]

Liest die Adresse der Ballerkennung aus der Konfigurationsdatei.

Parameter

out	*port*	Port der Ballerkennung

Rückgabe

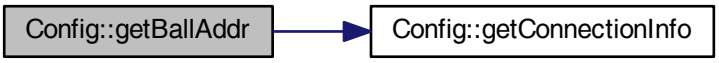
true wenn der Wert gelesen werden konnte

Definiert in Zeile 639 der Datei Config.cpp.

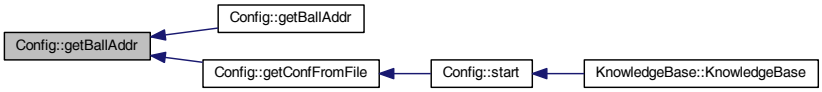
Benutzt getConnectionInfo().

Wird benutzt von getBallAddr() und getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.7 bool Config::getBallAddr (char *& *host*) [private]

Liest die Adresse der Ballerkennung aus der Konfigurationsdatei.

Parameter

out	*host*	Host der Ballerkennung

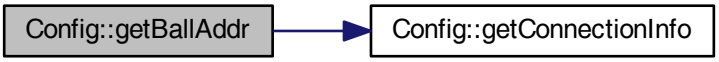
Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 647 der Datei Config.cpp.

Benutzt getConnectionInfo().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.12.3.8 bool Config::getBallAddr (unsigned short & *port*, char *& *host*) [private]

Liest die Adresse der Ballerkennung aus der Konfigurationsdatei.

Parameter

out	*port*	Port der Ballerkennung
out	*host*	Host der Ballerkennung

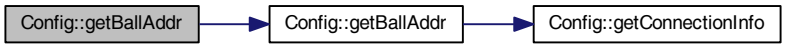
Rückgabe

true wenn die Werte gelesen werden konnten

Definiert in Zeile 655 der Datei Config.cpp.

Benutzt getBallAddr().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.12.3.9 bool Config::getBallDiameter (float & *diameter*) [private]

Liest den Durchmesser des Balles aus der Konfigurationsdatei.

Parameter

out	*diameter*	Durchmesser des Balles

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 910 der Datei Config.cpp.

Benutzt getDimension().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.10 bool Config::getBallToAxisReachable (float & *distance*) [private]

Liest die Distanz des Balles zur Spielstange, aus der der Ball von der Figur noch erreicht werden kann aus der Konfigurationsdatei.

Parameter

out	*distance*	Distanz des Balles zur Spielstange

Rückgabe

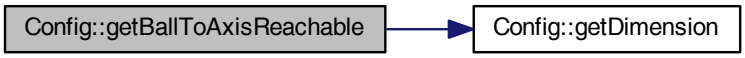
true wenn der Wert gelesen werden konnte

Definiert in Zeile 926 der Datei Config.cpp.

Benutzt getDimension().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.11 bool Config::getBallToAxisShootable (float & *distance*) [private]

Liest die Distanz des Balles zur Spielstange, aus der der Ball von der Figur noch geschossen werden kann aus der Konfigurationsdatei.

Parameter

out	*distance*	Distanz des Balles zur Spielstange

Rückgabe

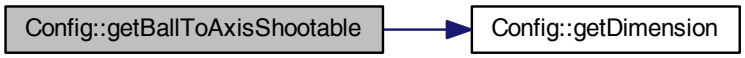
true wenn der Wert gelesen werden konnte

Definiert in Zeile 918 der Datei Config.cpp.

Benutzt getDimension().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.12 bool Config::getConfFromFile (ConfigVar *var* = All)

Liest die Konfiguration aus den Dateien.

Parameter

in	*var*	Angabe, welcher Wert gelesen werden soll

Rückgabe

true wenn der Wert gelesen werden konnte

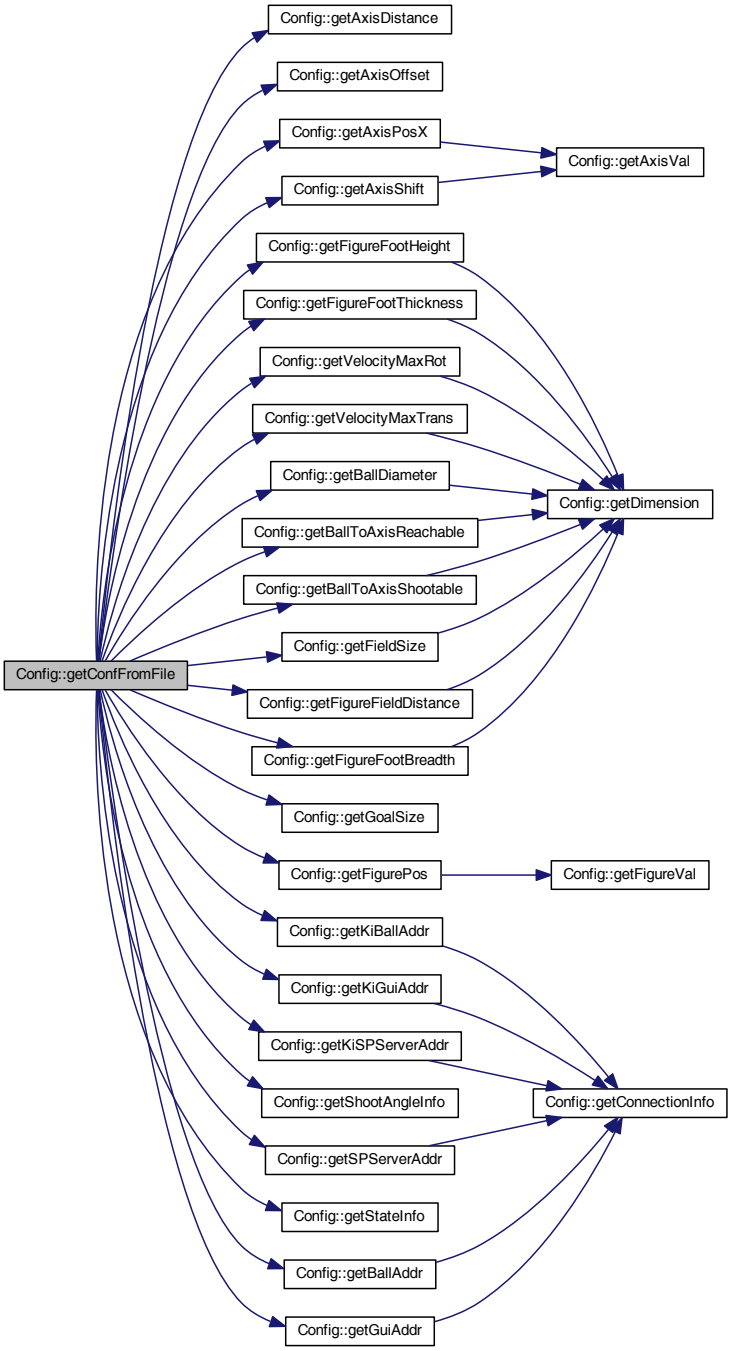
Bei All werden alle bekannten Werte gelesen.

Definiert in Zeile 111 der Datei Config.cpp.

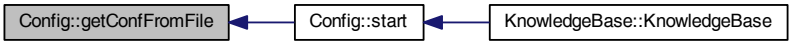
Benutzt All, AXIS_FOUR, AXIS_ONE, AxisDistance, AxisOffset, AxisPosX, AxisShift, BallDiameter, BallHost, BallPort, BallToAxisReachable, BallToAxisShootable, COORDINATE_X, COORDINATE_Y, FieldSize, FIGURE_FIVE, FIGURE_ONE, FigureFieldDistance, FigureFootBreadth, FigureFootHeight, FigureFootThickness, FigurePos, getAxisDistance(), getAxisOffset(), getAxisPosX(), getAxisShift(), getBallAddr(), getBallDiameter(), getBallToAxisReachable(), getBallToAxisShootable(), getFieldSize(), getFigureFieldDistance(), getFigureFootBreadth(), getFigureFootHeight(), getFigureFootThickness(), getFigurePos(), getGoalSize(), getGuiAddr(), getKiBallAddr(), getKiGuiAddr(), getKiSPServerAddr(), getShootAngleInfo(), getSPServerAddr(), getStateInfo(), getVelocityMaxRot(), getVelocityMaxTrans(), GoalSize, GuiHost, GuiPort, KiBallHost, KiBallPort, KiGuiHost, KiGuiPort, KiSPServerHost, KiSPServerPort, MAX_SHOOT_OFFSETS, MAX_STATES_PER_GROUP, NumShootAnglesByLin, NumShootAnglesByRot, pVar, ShootAnglesByLin, ShootAnglesByRot, SPServerHost, SPServerPort, States, STATES_ALL, StatesPerGroup, VelocityMaxRot und VelocityMaxTrans.

Wird benutzt von start().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.13 bool Config::getConnectionInfo (const char * *service*, const char * *attr*, char *& *addr*) [private]

Liest Werte aus der Konfigurationsdatei, die für verschiedene Verbindungen nötig sind.

Parameter

in	*service*	Dienst von dem die Daten zurückgegeben werden soll
in	*attr*	Das Attribut was zurückgegeben werden soll
out	*addr*	Der Wert

Rückgabe

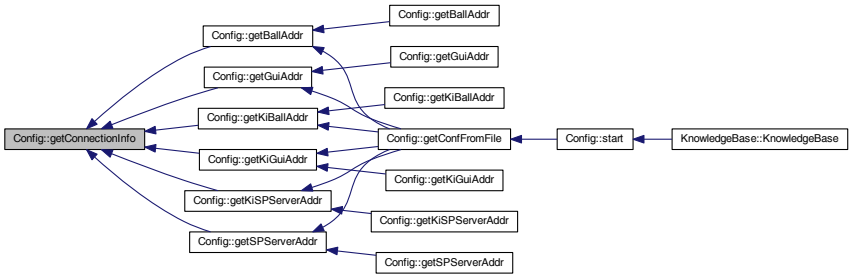
true wenn der Wert gelesen werden konnte

Definiert in Zeile 1045 der Datei Config.cpp.

Benutzt NULL und pProfileHandle.

Wird benutzt von getBallAddr(), getGuiAddr(), getKiBallAddr(), getKiGuiAddr(), getKiSPServerAddr() und getSP-ServerAddr().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.14 bool Config::getConnectionInfo (const char * *service*, unsigned short & *port*) [private]

Liest Port aus der Konfigurationsdatei, über den die Verbindung läuft.

Parameter

in	*service*	Dienst von dem die Daten zurückgegeben werden soll
out	*port*	Der Port

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 1065 der Datei Config.cpp.

Benutzt NULL und pProfileHandle.

5.12.3.15 bool Config::getDimension (const char * *child*, const char * *attr*, float & *val*) [private]

Liest Werte aus der Konfigurationsdatei, die sich auf die Dimensionierung beziehen.

Parameter

in	*child*	Eigenschaft (field, velocity, figure, ball, ...)
in	*attr*	Das Attribut was zurückgegeben werden soll
out	*val*	Der Wert

Rückgabe

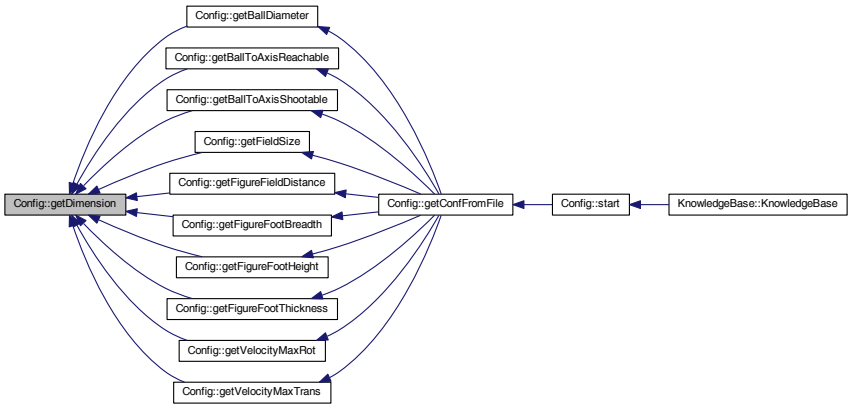
true wenn der Wert gelesen werden konnte

Definiert in Zeile 1088 der Datei Config.cpp.

Benutzt NULL und pProfileHandle.

Wird benutzt von getBallDiameter(), getBallToAxisReachable(), getBallToAxisShootable(), getFieldSize(), getFigureFieldDistance(), getFigureFootBreadth(), getFigureFootHeight(), getFigureFootThickness(), getVelocityMaxRot() und getVelocityMaxTrans().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.16 bool Config::getFieldSize (float & *depth*, float & *breadth*) [private]

Liest die Spielfeldgröße aus der Konfigurationsdatei.

Parameter

out	*depth*	Tiefe
out	*breadth*	Breite

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 831 der Datei Config.cpp.

Benutzt getDimension().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.17 bool Config::getFigureFieldDistance (float & *distance*) [private]

Liest den Abstand des Fußes der Spielfigur zum Spielfeld aus der Konfigurationsdatei.

Parameter

out	*distance*	Abstand der Spielfigur

Rückgabe

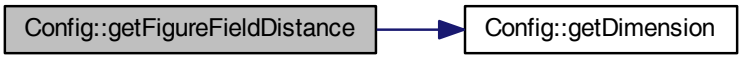
true wenn der Wert gelesen werden konnte

Definiert in Zeile 902 der Datei Config.cpp.

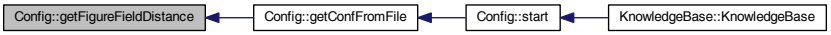
Benutzt getDimension().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.18 bool Config::getFigureFootBreadth (float & *breadth*) [private]

Liest die Breite des Fußes der Spielfigur aus der Konfigurationsdatei.

Parameter

out	*breadth*	Breite des Fußes

Rückgabe

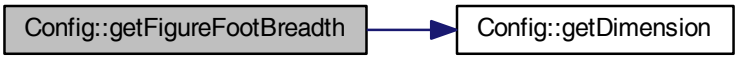
true wenn der Wert gelesen werden konnte

Definiert in Zeile 878 der Datei Config.cpp.

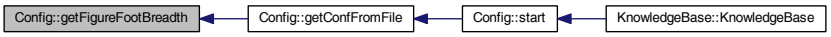
Benutzt getDimension().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.19 bool Config::getFigureFootHeight (float & *height*) [private]

Liest die Höhe des Fußes der Spielfigur aus der Konfigurationsdatei.

Parameter

out	*height*	Höhe des Fußes

Rückgabe

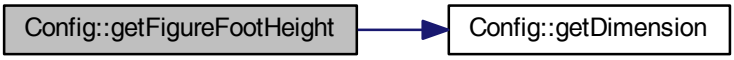
true wenn der Wert gelesen werden konnte

Definiert in Zeile 886 der Datei Config.cpp.

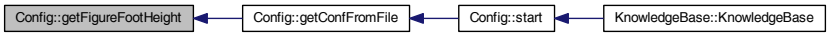
Benutzt getDimension().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.20 bool Config::getFigureFootThickness (float & *thickness*) [private]

Liest die Dicke des Fußes der Spielfigur aus der Konfigurationsdatei.

Parameter

out	*thickness*	Dicke des Fußes

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 894 der Datei Config.cpp.

Benutzt getDimension().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.21 bool Config::getFigurePos (const unsigned int *axis*, const unsigned int *figure*, float & *pos*) [private]

Liest die Position der Spielfigur an der Spielstange aus der Konfigurationsdatei.

Parameter

in	*axis*	Spielstange
in	*figure*	Spielfigur
out	*pos*	Abstand

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 823 der Datei Config.cpp.

Benutzt getFigureVal().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.22 bool Config::getFigureVal (const unsigned int *axis*, const unsigned int *figure*, const char * *attr*, float & *val*) [private]

Liest Werte aus der Konfigurationsdatei, die sich auf eine Spielfigur beziehen.

Parameter

in	*axis*	Spielstange
in	*figure*	Spielfigur
in	*attr*	Das Attribut was zurückgegeben werden soll
out	*val*	Der Wert

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 1139 der Datei Config.cpp.

Benutzt AXIS_FOUR, AXIS_ONE, NULL und pProfileHandle.

Wird benutzt von getFigurePos().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.23 bool Config::getGoalSize (float & *width*) [private]

Liest die Torgröße aus der Konfigurationsdatei.

Parameter

out	*width*	Breite

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 839 der Datei Config.cpp.

Benutzt NULL und pProfileHandle.

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.24 bool Config::getGuiAddr (unsigned short & *port*) [private]

Liest die Adresse der GUI aus der Konfigurationsdatei.

Parameter

out	*port*	Port der GUI

Rückgabe

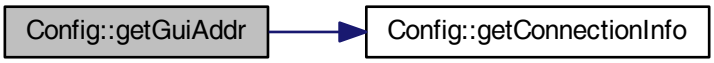
true wenn der Wert gelesen werden konnte

Definiert in Zeile 663 der Datei Config.cpp.

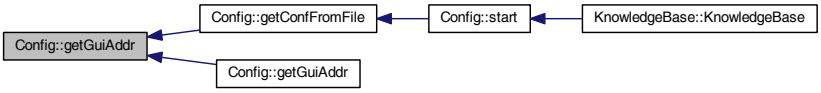
Benutzt getConnectionInfo().

Wird benutzt von getConfFromFile() und getGuiAddr().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.25 bool Config::getGuiAddr (char *& *host*) [private]

Liest die Adresse der GUI aus der Konfigurationsdatei.

Parameter

out	*host*	Host der GUI

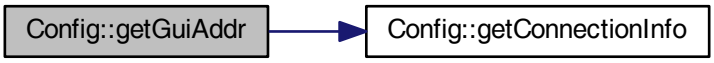
Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 671 der Datei Config.cpp.

Benutzt getConnectionInfo().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.12.3.26 bool Config::getGuiAddr (unsigned short & *port*, char *& *host*) [private]

Liest die Adresse der GUI aus der Konfigurationsdatei.

Parameter

out	*port*	Port der GUI
out	*host*	Host der GUI

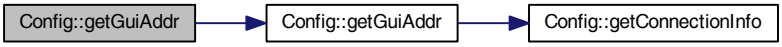
Rückgabe

true wenn die Werte gelesen werden konnten

Definiert in Zeile 679 der Datei Config.cpp.

Benutzt getGuiAddr().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.12.3.27 KnowledgeBase * Config::getKB (void)

Gibt den Zeiger auf die Wissensbasis zurück.

Rückgabe

Zeiger auf die Wissensbasis

Definiert in Zeile 84 der Datei Config.cpp.

Benutzt pKB.

5.12.3.28 bool Config::getKiBallAddr (unsigned short & *port*) [private]

Liest die Adresse der Ki für die Ballerkennung aus der Konfigurationsdatei.

Parameter

out	*port*	Port der Ki für die Ballerkennung

Rückgabe

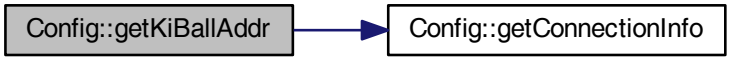
true wenn der Wert gelesen werden konnte

Definiert in Zeile 711 der Datei Config.cpp.

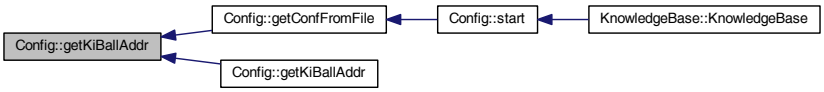
Benutzt getConnectionInfo().

Wird benutzt von getConfFromFile() und getKiBallAddr().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.29 bool Config::getKiBallAddr (char *& *host*) [private]

Liest die Adresse der Ki für die Ballerkennung aus der Konfigurationsdatei.

Parameter

out	*host*	Host der Ki für die Ballerkennung

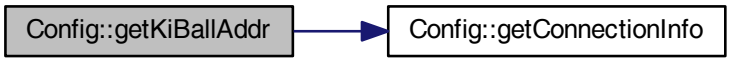
Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 719 der Datei Config.cpp.

Benutzt getConnectionInfo().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.12.3.30 bool Config::getKiBallAddr (unsigned short & *port*, char *& *host*) [private]

Liest die Adresse der für die Ballerkennung Ki aus der Konfigurationsdatei.

Parameter

out	*port*	Port der Ki für die Ballerkennung
out	*host*	Host der Ki für die Ballerkennung

Rückgabe

true wenn die Werte gelesen werden konnten

Definiert in Zeile 727 der Datei Config.cpp.

Benutzt getKiBallAddr().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.12.3.31 bool Config::getKiGuiAddr (unsigned short & *port*) [private]

Liest die Adresse der Ki für die GUI aus der Konfigurationsdatei.

Parameter

out	*port*	Port der Ki für die GUI

Rückgabe

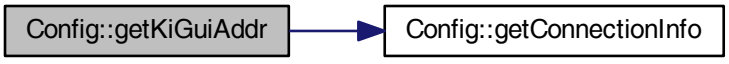
true wenn der Wert gelesen werden konnte

Definiert in Zeile 735 der Datei Config.cpp.

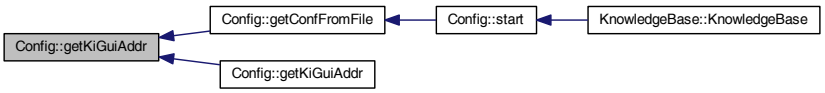
Benutzt getConnectionInfo().

Wird benutzt von getConfFromFile() und getKiGuiAddr().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.32 bool Config::getKiGuiAddr (char *& *host*) [private]

Liest die Adresse der Ki für die GUI aus der Konfigurationsdatei.

Parameter

out	*host*	Host der Ki für die GUI

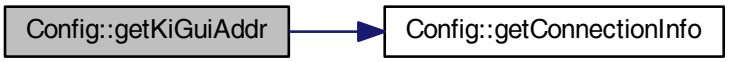
Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 743 der Datei Config.cpp.

Benutzt getConnectionInfo().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.12.3.33 bool Config::getKiGuiAddr (unsigned short & *port*, char *& *host*) [private]

Liest die Adresse der Ki aus der Konfigurationsdatei.

Parameter

out	*port*	Port der Ki für die GUI
out	*host*	Host der Ki für die GUI

Rückgabe

true wenn die Werte gelesen werden konnten

Definiert in Zeile 751 der Datei Config.cpp.

Benutzt getKiGuiAddr().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.12.3.34 bool Config::getKiSPServerAddr (unsigned short & *port*) [private]

Liest die Adresse der Ki für den SPServer aus der Konfigurationsdatei.

Parameter

out	*port*	Port der Ki für den SPServer

Rückgabe

true wenn der Wert gelesen werden konnte

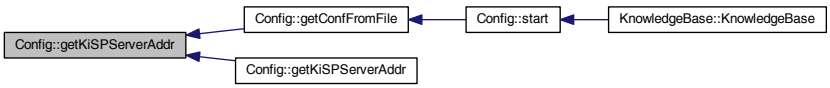
Definiert in Zeile 687 der Datei Config.cpp.

Benutzt getConnectionInfo().

Wird benutzt von getConfFromFile() und getKiSPServerAddr().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.35 bool Config::getKiSPServerAddr (char *& *host*) [private]

Liest die Adresse der Ki für den SPServer aus der Konfigurationsdatei.

Parameter

out	*host*	Host der Ki für den SPServer

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 695 der Datei Config.cpp.

Benutzt getConnectionInfo().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.12.3.36 bool Config::getKiSPServerAddr (unsigned short & *port*, char *& *host*) [private]

Liest die Adresse der für den SPServer Ki aus der Konfigurationsdatei.

Parameter

out	*port*	Port der Ki für den SPServer
out	*host*	Host der Ki für den SPServer

Rückgabe

true wenn die Werte gelesen werden konnten

Definiert in Zeile 703 der Datei Config.cpp.

Benutzt getKiSPServerAddr().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.12.3.37 bool Config::getShootAngleInfo (unsigned int *shoot*, unsigned int *num*, float *& *val*) [private]

Liest Werte aus der Konfigurationsdatei, die sich auf die Schusswinkel beziehen.

Parameter

in	*shoot*	Art der Eigenschaft (BY_ROTATION oder BY_TRANSLATION)
in	*num*	Nummer des Winkels (0 - Anzahl der Abgelegten Winkel - 1)
out	*val*	Zweidimensionales Array mit angle, und offset

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 1190 der Datei Config.cpp.

Benutzt NULL, pProfileHandle, SHOOT_ANGLE und SHOOT_OFFSET.

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.38 bool Config::getSPServerAddr (unsigned short & *port*) [private]

Liest den Port des TCP-Servers aus der Konfigurationsdatei.

Parameter

out	*port*	Port des SPServer

Rückgabe

true wenn der Wert gelesen werden konnte

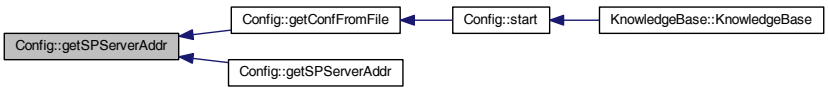
Definiert in Zeile 615 der Datei Config.cpp.

Benutzt getConnectionInfo().

Wird benutzt von getConfFromFile() und getSPServerAddr().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.39 bool Config::getSPServerAddr (char *& *host*) [private]

Liest die Adresse des SPServer aus der Konfigurationsdatei.

Parameter

out	*host*	Host des SPServer

Rückgabe

true wenn der Wert gelesen werden konnte

Definiert in Zeile 623 der Datei Config.cpp.

Benutzt getConnectionInfo().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.12.3.40 bool Config::getSPServerAddr (unsigned short & *port*, char *& *host*) [private]

Liest die Adresse des SPServer aus der Konfigurationsdatei.

Parameter

out	*port*	Port des SPServer
out	*host*	Host des SPServer

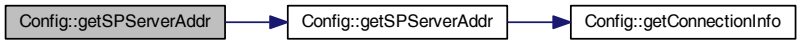
Rückgabe

true wenn die Werte gelesen werden konnten

Definiert in Zeile 631 der Datei Config.cpp.

Benutzt getSPServerAddr().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.12.3.41 bool Config::getState (unsigned int *group*, unsigned int *state*, int & *val*)

Liefert den Wert eines Statusses.

Parameter

in	*group*	Gruppe des Status
in	*state*	Status
out	*val*	Wert des Status

Rückgabe

true wenn der Wert verfügbar ist

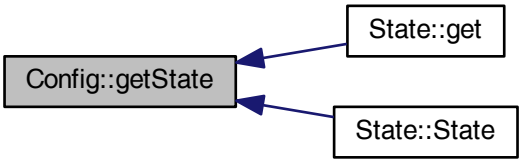
Noch zu erledigen Als getVar() umschreiben

Definiert in Zeile 563 der Datei Config.cpp.

Benutzt MAX_STATES_PER_GROUP, pVar, States und STATES_ALL.

Wird benutzt von State::get() und State::State().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.42 bool Config::getState (unsigned int *group*, unsigned int *state*, int(&) *val[3]*)

Liefert die Werte eines Statusses.

Parameter

in	*group*	Gruppe des Status
in	*state*	Status
out	*val*	Werte des Status (Wert, min und max)

Rückgabe

true wenn die Werte verfügbar sind

Noch zu erledigen Als getVar() umschreiben

Definiert in Zeile 575 der Datei Config.cpp.

Benutzt MAX_STATES_PER_GROUP, pVar, States und STATES_ALL.

5.12.3.43 bool Config::getStateInfo (unsigned int *group*, unsigned int *state*, int *& *vals*) [private]

Liest Werte aus der Konfigurationsdatei, die für die Statusse benötigt werden.

Parameter

in	*group*	Gruppe der Statusse aus der die Daten zurückgegeben werden sollen
in	*state*	Status von dem die Daten zurückgegeben werden sollen
out	*vals*	Dreidimensionales Array mit default, minimal und maximal Wert des Statusses

Rückgabe

true wenn die Werte gelesen werden konnten

Definiert in Zeile 1165 der Datei Config.cpp.

Benutzt NULL und pProfileHandle.

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.44 bool Config::getStatesInGroup (unsigned int *group*, unsigned int & *states*)

Liefert die Anzahl der Statusse der Gruppe zurück.

Parameter

in	*group*	Gruppe der Statusse
out	*states*	Anzahl der Status

Rückgabe

true wenn der Wert verfügbar ist

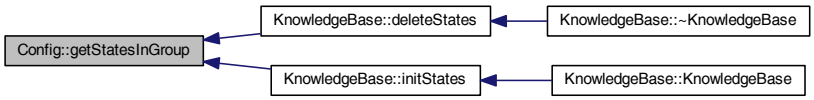
Noch zu erledigen Als getVar() umschreiben

Definiert in Zeile 590 der Datei Config.cpp.

Benutzt pVar, STATES_ALL und StatesPerGroup.

Wird benutzt von KnowledgeBase::deleteStates() und KnowledgeBase::initStates().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.45 bool Config::getVar (ConfigVar *var*, unsigned short & *usVal*)

Gibt den Wert von var zurück.

Parameter

in	*var*	Angabe, welcher Wert zurückgegeben werden soll
out	*usVal*	Referenz auf die Variable, in die der Wert abgelegt wird

Rückgabe

true wenn der Wert gültig ist

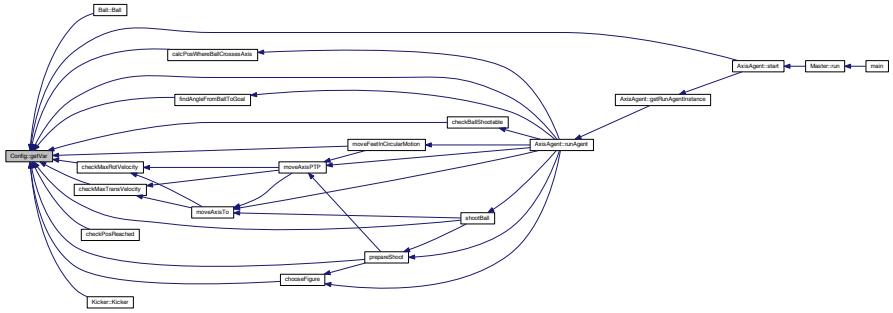
Gibt einen unsigned short Wert zurück.

Definiert in Zeile 427 der Datei Config.cpp.

Benutzt All und pVar.

Wird benutzt von Ball::Ball(), calcPosWhereBallCrossesAxis(), checkBallShootable(), checkMaxRotVelocity(), checkMaxTransVelocity(), checkPosReached(), chooseFigure(), findAngleFromBallToGoal(), Kicker::Kicker(), moveFeetInCircularMotion(), prepareShoot(), AxisAgent::runAgent(), shootBall() und AxisAgent::start().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.46 bool Config::getVar (ConfigVar *var*, unsigned short(&) *ausVal[2]*)

Gibt den Wert von var zurück.

Parameter

in	*var*	Angabe, welcher Wert zurückgegeben werden soll
out	*ausVal*	Referenz auf die Variable, in die der Wert abgelegt wird

Rückgabe

true wenn der Wert gültig ist

Gibt einen unsigned short-Array mit zwei Werten zurück.

Definiert in Zeile 438 der Datei Config.cpp.

Benutzt All, COMPUTER, HUMAN und pVar.

5.12.3.47 bool Config::getVar (ConfigVar *var*, char *& *pcVal*)

Gibt den Wert von var zurück.

Parameter

in	*var*	Angabe, welcher Wert zurückgegeben werden soll
out	*pcVal*	Referenz auf die Variable, in die der Wert abgelegt wird

Rückgabe

true wenn der Wert gültig ist

Gibt einen char∗ Wert zurück.

Definiert in Zeile 450 der Datei Config.cpp.

Benutzt All und pVar.

5.12.3.48 bool Config::getVar (ConfigVar *var*, float & *fVal*)

Gibt den Wert von var zurück.

Parameter

in	*var*	Angabe, welcher Wert zurückgegeben werden soll
out	*fVal*	Referenz auf die Variable, in die der Wert abgelegt wird

Rückgabe

true wenn der Wert gültig ist

Gibt einen float Wert zurück.

Definiert in Zeile 461 der Datei Config.cpp.

Benutzt All und pVar.

5.12.3.49 bool Config::getVar (ConfigVar *var*, float(&) *afVal[2]*)

Gibt den Wert von var zurück.

Parameter

in	*var*	Angabe, welcher Wert zurückgegeben werden soll
out	*afVal*	Referenz auf die Variable, in die der Wert abgelegt wird

Rückgabe

true wenn der Wert gültig ist

Gibt einen float-Array mir zwei Werten zurück.

Definiert in Zeile 472 der Datei Config.cpp.

Benutzt All, COORDINATE_X, COORDINATE_Y und pVar.

5.12.3.50 bool Config::getVar (ConfigVar *var*, float(&) *afVal[4]*)

Gibt den Wert von var zurück.

Parameter

in	*var*	Angabe, welcher Wert zurückgegeben werden soll
out	*afVal*	Referenz auf die Variable, in die der Wert abgelegt wird

Rückgabe

true wenn der Wert gültig ist

Gibt einen float-Array mir vier Werten zurück.

Definiert in Zeile 484 der Datei Config.cpp.

Benutzt All, AXIS_FOUR, AXIS_ONE und pVar.

5.12.3.51 bool Config::getVar (ConfigVar *var*, float(&) *afVal[4][5]*)

Gibt den Wert von var zurück.

Parameter

in	*var*	Angabe, welcher Wert zurückgegeben werden soll
out	*afVal*	Referenz auf die Variable, in die der Wert abgelegt wird

Rückgabe

true wenn der Wert gültig ist

Gibt einen float-Array mir vier mal fünf Werten zurück.

Definiert in Zeile 496 der Datei Config.cpp.

Benutzt All, AXIS_FOUR, AXIS_ONE, FIGURE_FIVE, FIGURE_ONE und pVar.

5.12.3.52 bool Config::getVar (ConfigVar *var*, bool(&) *abVal[4]*)

Gibt den Wert von var zurück.

Parameter

in	*var*	Angabe, welcher Wert zurückgegeben werden soll
out	*abVal*	Referenz auf die Variable, in die der Wert abgelegt wird

Rückgabe

true wenn der Wert gültig ist

Gibt einen bool-Array mit vier Werten zurück.

Definiert in Zeile 509 der Datei Config.cpp.

Benutzt All, AXIS_FOUR, AXIS_ONE und pVar.

5.12.3.53 bool Config::getVar (ConfigVar *var*, int & *iVal*)

Gibt den Wert von var zurück.

Parameter

in	*var*	Angabe, welcher Wert zurückgegeben werden soll
out	*iVal*	Referenz auf die Variable, in die der Wert abgelegt wird

Rückgabe

true wenn der Wert gültig ist

Gibt einen int Wert zurück.

Definiert in Zeile 521 der Datei Config.cpp.

Benutzt All und pVar.

5.12.3.54 bool Config::getVar (ConfigVar *var*, int *iArraySize1*, int *iArraySize2*, float **& *afVal*)

Gibt den Wert von var zurück.

Parameter

in	*var*	Angabe, welcher Wert zurückgegeben werden soll
in	*iArraySize1*	Arraygröße der ersten Dimension
in	*iArraySize2*	Arraygröße der zweiten Dimension
out	*afVal*	Referenz auf die Variable, in die der Wert abgelegt wird

Rückgabe

true wenn der Wert gültig ist

Gibt einen float-Array mir vier mal X Werten zurück.

Definiert in Zeile 532 der Datei Config.cpp.

Benutzt All und pVar.

5.12.3.55 bool Config::getVelocityMaxRot (float & *rotVelo*) [private]

Liest die maxinale Geschwindigkeit der Rotation aus der Konfigurationsdatei.

Parameter

out	*rotVelo*	maxinale Geschwindigkeit

Rückgabe

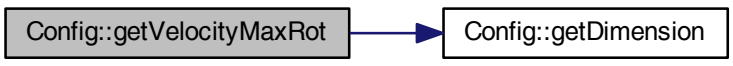
true wenn der Wert gelesen werden konnte

Definiert in Zeile 870 der Datei Config.cpp.

Benutzt getDimension().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.12.3.56 bool Config::getVelocityMaxTrans (float & *transVelo*) [private]

Liest die maxinale Geschwindigkeit der Translation aus der Konfigurationsdatei.

Parameter

out	*transVelo*	maxinale Geschwindigkeit

Rückgabe

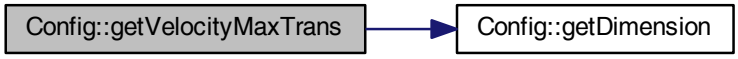
true wenn der Wert gelesen werden konnte

Definiert in Zeile 862 der Datei Config.cpp.

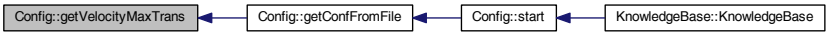
Benutzt getDimension().

Wird benutzt von getConfFromFile().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.57 bool Config::setState (unsigned int *group*, unsigned int *state*, int *val*)

Setzt den Wert eines Statusses.

Parameter

in	*group*	Gruppe des Status
in	*state*	Status
out	*val*	Wert des Status

Rückgabe

true wenn der Wert gesetzt werden konnte

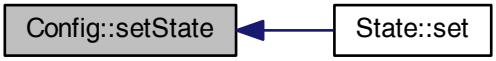
Noch zu erledigen Als setVar() umschreiben

Definiert in Zeile 601 der Datei Config.cpp.

Benutzt MAX_STATES_PER_GROUP, pVar, States und STATES_ALL.

Wird benutzt von State::set().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.58 void Config::start (void)

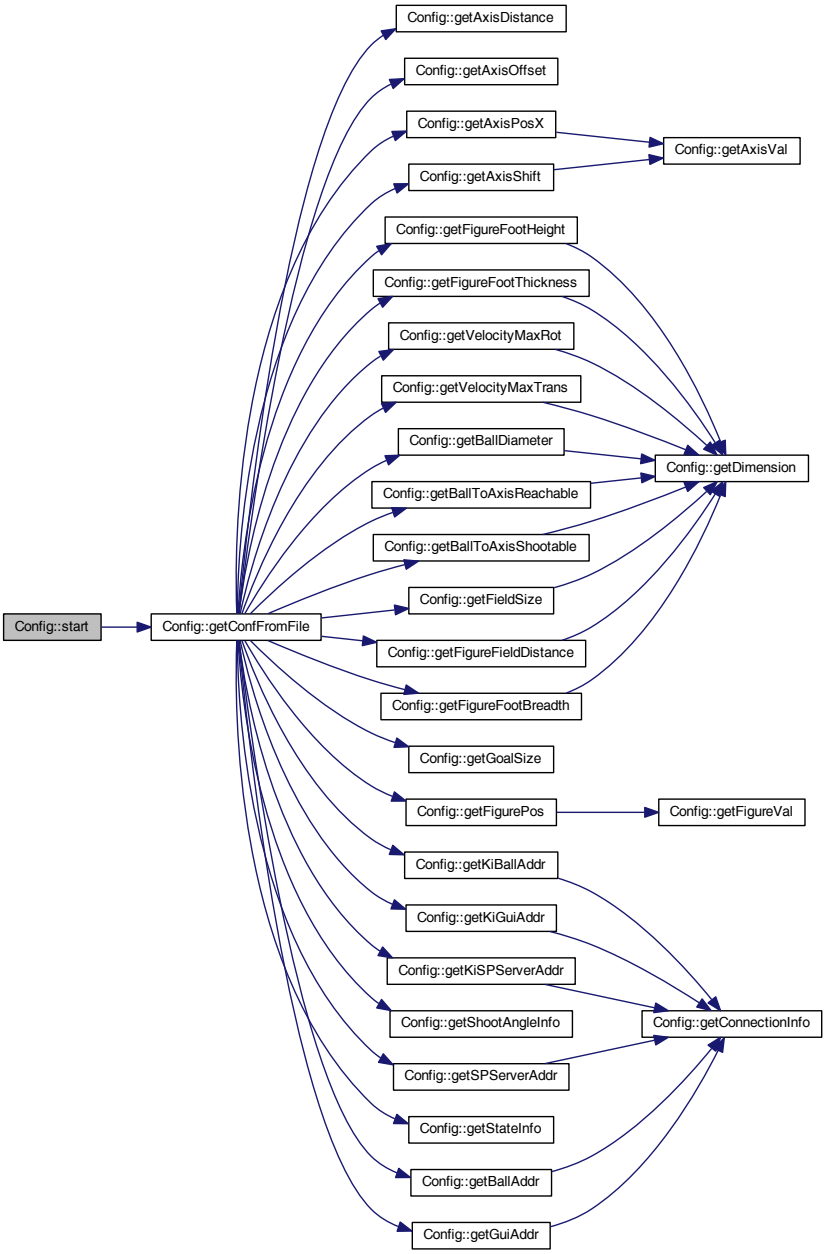
Lädt die Konfiguration aus den entsprechenden Dateien.

Definiert in Zeile 89 der Datei Config.cpp.

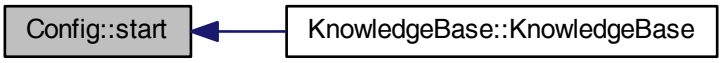
Benutzt ERROR_CONFIG und getConfFromFile().

Wird benutzt von KnowledgeBase::KnowledgeBase().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.12.3.59 bool Config::writeConfToFile (ConfigVar *var* = All)

Schreibt die Konfiguration in die Dateien.

Parameter

in	*var*	Angabe, welcher Wert geschrieben werden soll

Rückgabe

true wenn der Wert geschrieben werden konnte

Bei All werden alle bekannten Werte geschrieben.

Definiert in Zeile 421 der Datei Config.cpp.

5.12.4 Dokumentation der Datenelemente

5.12.4.1 KnowledgeBase* Config::pKB [private]

Zeiger auf die Wissensbasis.

Definiert in Zeile 227 der Datei Config.h.

Wird benutzt von getKB().

5.12.4.2 TiXmlHandle* Config::pProfileHandle [private]

Handle auf die Profil-Kongigurationsdatei.

Definiert in Zeile 237 der Datei Config.h.

Wird benutzt von Config(), getAxisDistance(), getAxisOffset(), getAxisVal(), getConnectionInfo(), getDimension(), getFigureVal(), getGoalSize(), getShootAngleInfo() und getStateInfo().

5.12.4.3 TiXmlDocument Config::profileDoc [private]

Die Datei, die die Grundkonfiguration beinhaltet.

Definiert in Zeile 232 der Datei Config.h.

Wird benutzt von Config().

5.12.4.4 void* Config::pVar[All] [private]

Array mit den Zeigern auf die einzelnen Variablen.

Definiert in Zeile 242 der Datei Config.h.

Wird benutzt von Config(), getConfFromFile(), getState(), getStatesInGroup(), getVar() und setState().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

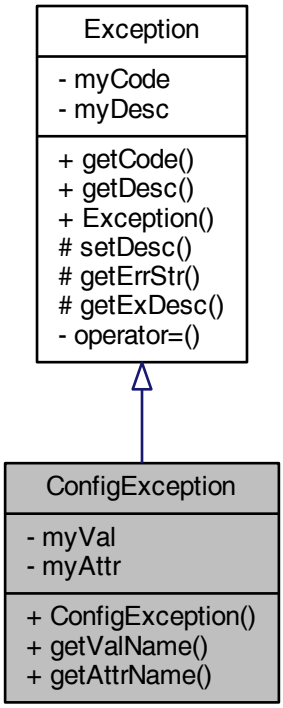
- Config.h
- Config.cpp

5.13 ConfigException Klassenreferenz

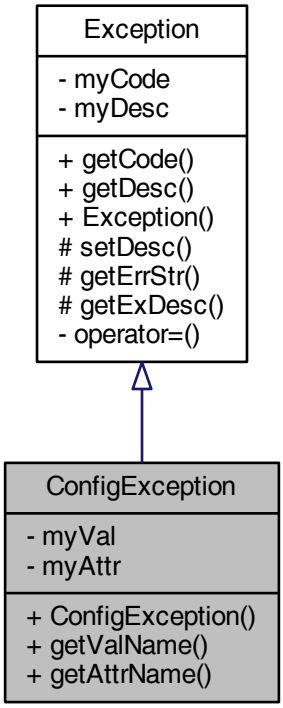
Exception-Klasse für die Konfigurationsdateien.

```
#include <Exception.h>
```

Klassendiagramm für ConfigException:



Zusammengehörigkeiten von ConfigException:



Öffentliche Methoden

- ConfigException (const char *val, const char *attr)

  Konstruktor.
- const char * getValName () const

  gibt die Eigenschaft zurück
- const char * getAttrName () const

  gibt das Attribut zurück

Private Attribute

- const char * myVal

  Eigenschaft.
- const char * myAttr

  Attribut.

5.13.1 Ausführliche Beschreibung

Exception-Klasse für die Konfigurationsdateien.

Definiert in Zeile 172 der Datei Exception.h.

5.13.2 Beschreibung der Konstruktoren und Destruktoren

5.13.2.1 ConfigException::ConfigException (const char * *val*, const char * *attr*) [inline]

Konstruktor.

Parameter

in	*val*	Name der Eigenschaft bei der das Problem aufgetreten ist
in	*attr*	Name des Attributes der Eigenschaft bei der das Problem aufgetreten ist

Setzt den Exception-Code, die Eigenschaft und das Attribut.

Definiert in Zeile 184 der Datei Exception.h.

5.13.3 Dokumentation der Elementfunktionen

5.13.3.1 const char* ConfigException::getAttrName ( ) const [inline]

gibt das Attribut zurück

Rückgabe

Attribut

Definiert in Zeile 199 der Datei Exception.h.

Benutzt myAttr.

Wird benutzt von KnowledgeBase::KnowledgeBase().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.13.3.2 const char* ConfigException::getValName ( ) const [inline]

gibt die Eigenschaft zurück

Rückgabe

Eigenschaft

Definiert in Zeile 193 der Datei Exception.h.

Benutzt myVal.

Wird benutzt von KnowledgeBase::KnowledgeBase().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.13.4 Dokumentation der Datenelemente

5.13.4.1 const char* ConfigException::myAttr [private]

Attribut.

Definiert in Zeile 210 der Datei Exception.h.

Wird benutzt von getAttrName().

5.13.4.2 const char* ConfigException::myVal [private]

Eigenschaft.

Definiert in Zeile 205 der Datei Exception.h.

Wird benutzt von getValName().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

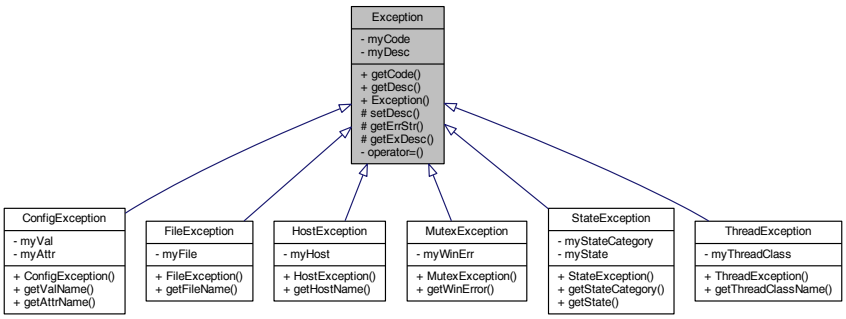
- Exception.h

5.14 Exception Klassenreferenz

Basis-Klasse von der alle Exception-Klassen abgeleitet werden sollten.

`#include <Exception.h>`

Klassendiagramm für Exception:



Öffentliche Methoden

- const unsigned int getCode () const

gibt den Exception-Code zurück

- const char ∗ getDesc () const

  gibt die Exception-Beschreibung zurück
- Exception (const unsigned int code)

  Konstruktor.

Geschützte Methoden

- void setDesc (const char ∗desc)

  Setzt die Exception Beschriebung.
- const char ∗ getErrStr (const unsigned int code)

  Hilfsfunktion, erzeugt den Tag-Namen der XML-Datei.
- const char ∗ getExDesc (const unsigned int exCode)

  gibt eine Beschreibung der Exception zurück

Private Methoden

- Exception & operator= (const Exception &)

  Zuweisungsoperator und Copy-Konstruktor.

Private Attribute

- unsigned int myCode

  Exception-Code.
- const char ∗ myDesc

  Exception Beschreibung.

5.14.1 Ausführliche Beschreibung

Basis-Klasse von der alle Exception-Klassen abgeleitet werden sollten.

Definiert in Zeile 22 der Datei Exception.h.

5.14.2 Beschreibung der Konstruktoren und Destruktoren

5.14.2.1 Exception::Exception (const unsigned int *code*) [inline]

Konstruktor.

Parameter

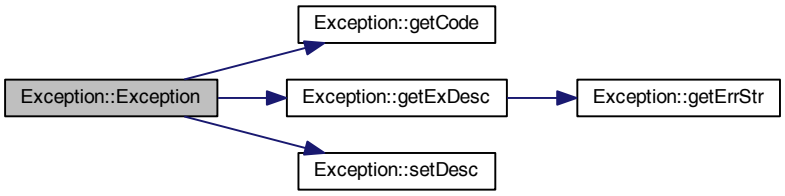
in	*code*	Exception-Code

Setzt den Exception-Code.

Definiert in Zeile 43 der Datei Exception.h.

Benutzt getCode(), getExDesc(), myCode und setDesc().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.14.3 Dokumentation der Elementfunktionen

5.14.3.1 const unsigned int Exception::getCode ( ) const [inline]

gibt den Exception-Code zurück

Rückgabe

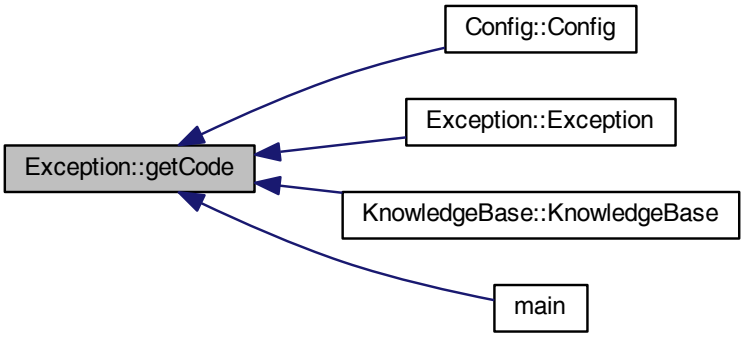
den Exception-Code

Definiert in Zeile 29 der Datei Exception.h.

Benutzt myCode.

Wird benutzt von Config::Config(), Exception(), KnowledgeBase::KnowledgeBase() und main().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.14.3.2 const char* Exception::getDesc ( ) const [inline]

gibt die Exception-Beschreibung zurück

Rückgabe

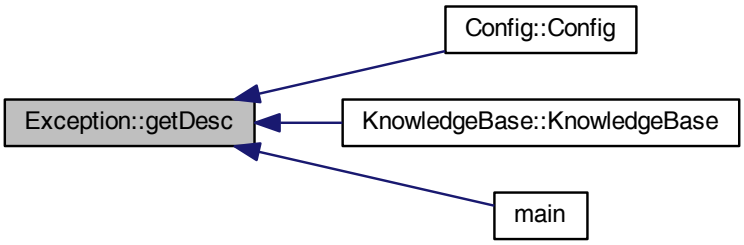
die Exception Beschreibung

Definiert in Zeile 35 der Datei Exception.h.

Benutzt myDesc.

Wird benutzt von Config::Config(), KnowledgeBase::KnowledgeBase() und main().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.14.3.3 const char* Exception::getErrStr (const unsigned int *code*) [inline, protected]

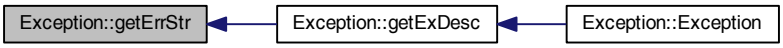
Hilfsfunktion, erzeugt den Tag-Namen der XML-Datei.

Definiert in Zeile 67 der Datei Exception.h.

Benutzt ERROR_UNKNOWN und myCode.

Wird benutzt von getExDesc().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.14.3.4 const char* Exception::getExDesc (const unsigned int *exCode*) [inline, protected]

gibt eine Beschreibung der Exception zurück

Definiert in Zeile 90 der Datei Exception.h.

Benutzt getErrStr().

Wird benutzt von Exception().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.14.3.5 Exception& Exception::operator= (const Exception &) [private]

Zuweisungsoperator und Copy-Konstruktor.

Wird als private Member-Funktion gesetzt um ein Kopieren der Exception zu vermeiden.

5.14.3.6 void Exception::setDesc (const char * *desc*) [inline, protected]

Setzt die Exception Beschriebung.

Definiert in Zeile 55 der Datei Exception.h.

Benutzt myDesc.

Wird benutzt von Exception().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.14.4 Dokumentation der Datenelemente

5.14.4.1 unsigned int Exception::myCode [private]

Exception-Code.

Definiert in Zeile 121 der Datei Exception.h.

Wird benutzt von Exception(), getCode() und getErrStr().

5.14.4.2 const char* Exception::myDesc [private]

Exception Beschreibung.

Definiert in Zeile 126 der Datei Exception.h.

Wird benutzt von getDesc() und setDesc().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

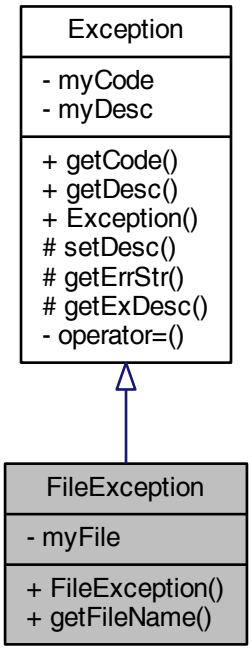
- Exception.h

5.15 FileException Klassenreferenz

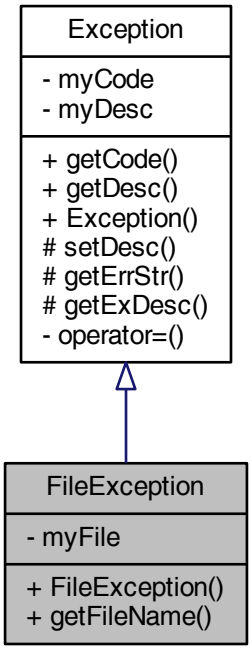
Exception-Klasse für Dateizugriff.

`#include <Exception.h>`

Klassendiagramm für FileException:



Zusammengehörigkeiten von FileException:



Öffentliche Methoden

- FileException (const char *file)

  Konstruktor.
- const char * getFileName () const

  gibt den Dateinamen zurück

Private Attribute

- const char * myFile

  Dateiname.

5.15.1 Ausführliche Beschreibung

Exception-Klasse für Dateizugriff.

Definiert in Zeile 141 der Datei Exception.h.

5.15.2 Beschreibung der Konstruktoren und Destruktoren

5.15.2.1 FileException::FileException (const char * *file*) [inline]

Konstruktor.

Parameter

in	*file*	Name der Datei bei der das Problem aufgetreten ist

Setzt den Exception-Code und den Dateinamen.

Definiert in Zeile 150 der Datei Exception.h.

5.15.3 Dokumentation der Elementfunktionen

5.15.3.1 const char* FileException::getFileName ( ) const [inline]

gibt den Dateinamen zurück

Rückgabe

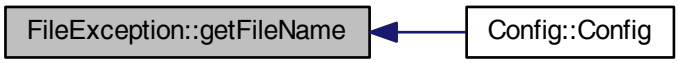
Dateiname

Definiert in Zeile 159 der Datei Exception.h.

Benutzt myFile.

Wird benutzt von Config::Config().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.15.4 Dokumentation der Datenelemente

5.15.4.1 const char* FileException::myFile [private]

Dateiname.

Definiert in Zeile 165 der Datei Exception.h.

Wird benutzt von getFileName().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Exception.h

5.16 float_1 Strukturreferenz

Anfrage-Paket für bestimmte Daten.

`#include <UdpDefines.h>`

Öffentliche Attribute

- unsigned int id: 16
- float flt

5.16.1 Ausführliche Beschreibung

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein float-Wert.

Definiert in Zeile 172 der Datei UdpDefines.h.

5.16.2 Dokumentation der Datenelemente

5.16.2.1 float float_1::flt

Definiert in Zeile 175 der Datei UdpDefines.h.

5.16.2.2 unsigned int float_1::id

Definiert in Zeile 174 der Datei UdpDefines.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- UdpDefines.h

5.17 float_2_2_4 Strukturreferenz

Anfrage-Paket für bestimmte Daten.

```
#include <UdpDefines.h>
```

Öffentliche Attribute

- unsigned int id: 16
- float flt [2][2][4]

5.17.1 Ausführliche Beschreibung

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein dreidimensionales Array aus zwei mal zwei mal vier float-Werten.

Definiert in Zeile 205 der Datei UdpDefines.h.

5.17.2 Dokumentation der Datenelemente

5.17.2.1 float float_2_2_4::flt[2][2][4]

Definiert in Zeile 208 der Datei UdpDefines.h.

Wird benutzt von SpsClient::sendAxisPos() und SpsClient::update().

5.17.2.2 unsigned int float_2_2_4::id

Definiert in Zeile 207 der Datei UdpDefines.h.

Wird benutzt von SpsClient::sendAxisPos().

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- UdpDefines.h

5.18 float_2_4 Strukturreferenz

Anfrage-Paket für bestimmte Daten.

```
#include <UdpDefines.h>
```

Öffentliche Attribute

- unsigned int id: 16
- float flt [2][4]

5.18.1 Ausführliche Beschreibung

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein zweidimensionales Array aus zwei mal vier float-Werten.

Definiert in Zeile 194 der Datei UdpDefines.h.

5.18.2 Dokumentation der Datenelemente

5.18.2.1 float float_2_4::flt[2][4]

Definiert in Zeile 197 der Datei UdpDefines.h.

5.18.2.2 unsigned int float_2_4::id

Definiert in Zeile 196 der Datei UdpDefines.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- UdpDefines.h

5.19 float_4 Strukturreferenz

Anfrage-Paket für bestimmte Daten.

```
#include <UdpDefines.h>
```

Öffentliche Attribute

- unsigned int id: 16
- float flt [4]

5.19.1 Ausführliche Beschreibung

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein Array aus vier float-Werten.

Definiert in Zeile 183 der Datei UdpDefines.h.

5.19.2 Dokumentation der Datenelemente

5.19.2.1 float float_4::flt[4]

Definiert in Zeile 186 der Datei UdpDefines.h.

5.19.2.2 unsigned int float_4::id

Definiert in Zeile 185 der Datei UdpDefines.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

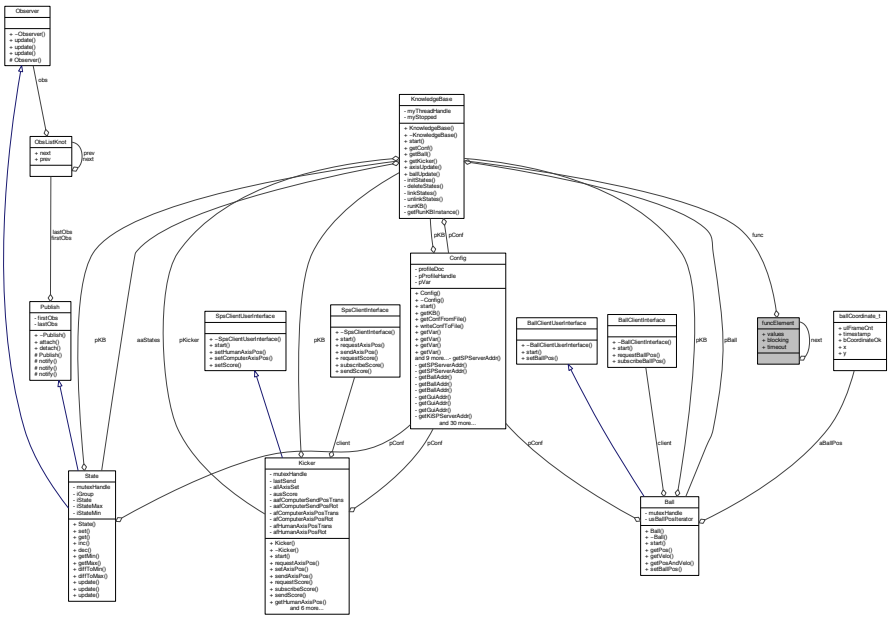
- UdpDefines.h

5.20 funcElement Strukturreferenz

Struktur für ein Listenelement.

`#include <AxisAgent.h>`

Zusammengehörigkeiten von funcElement:



Öffentliche Attribute

- moveFunc func

  Pointer auf die Funktion die auszuführen ist.
- float * values

  Parameter-Array.
- bool blocking

  Angabe ob die Funktion blockierend ist oder nicht.
- int timeout

  Timeout der Funktion.
- funcElement * next

  Pointer auf das nächste Listenelement.

5.20.1 Ausführliche Beschreibung

Struktur für ein Listenelement.

Struktur für ein Listenelement einer einfach verketteten Liste, welche Funktionspointer enthält.

Definiert in Zeile 29 der Datei AxisAgent.h.

5.20.2 Dokumentation der Datenelemente

5.20.2.1 bool funcElement::blocking

Angabe ob die Funktion blockierend ist oder nicht.

Definiert in Zeile 36 der Datei AxisAgent.h.

Wird benutzt von FuncList::attach() und FuncList::execFunc().

5.20.2.2 moveFunc funcElement::func

Pointer auf die Funktion die auszuführen ist.

Definiert in Zeile 32 der Datei AxisAgent.h.

Wird benutzt von FuncList::attach() und FuncList::execFunc().

5.20.2.3 funcElement * funcElement::next

Pointer auf das nächste Listenelement.

Definiert in Zeile 40 der Datei AxisAgent.h.

Wird benutzt von FuncList::attach(), AxisAgent::runAgent() und FuncList::~FuncList().

5.20.2.4 int funcElement::timeout

Timeout der Funktion.

Definiert in Zeile 38 der Datei AxisAgent.h.

Wird benutzt von FuncList::attach() und FuncList::execFunc().

5.20.2.5 float * funcElement::values

Parameter-Array.

Definiert in Zeile 34 der Datei AxisAgent.h.

Wird benutzt von FuncList::attach(), FuncList::execFunc() und FuncList::~FuncList().

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Dateien:

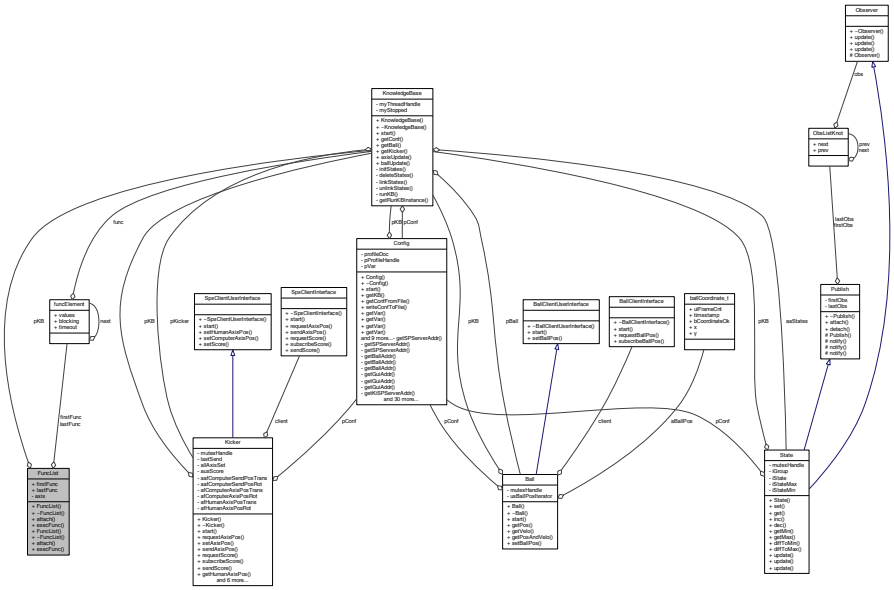
- AxisAgent.h
- AxisAgent_wobbeln.h

5.21 FuncList Klassenreferenz

Eine einfach verkettete Liste für Funktionspointer.

```
#include <AxisAgent.h>
```

Zusammengehörigkeiten von FuncList:



Öffentliche Methoden

- FuncList (Axis ax, KnowledgeBase ∗kB)

  Konstruktor.
- ∼FuncList ()

  Destruktor.
- void attach (moveFunc func, float values[], int arraysize, bool blocking=true, int timeout=1000)

  Fügt einen Funktionspointer in die Liste hinzu.
- bool execFunc (const funcElement ∗funcEl)

  Führt die in der Strukrur übergebene Funktion mir den entsprechenden Parametern aus.
- FuncList (Axis ax, KnowledgeBase ∗kB)

  Konstruktor.
- ∼FuncList ()

  Destruktor.
- void attach (moveFunc func, float values[], int arraysize, bool blocking=true, int timeout=1000)

  Fügt einen Funktionspointer in die Liste hinzu.
- bool execFunc (const funcElement ∗funcEl)

  Führt die in der Strukrur übergebene Funktion mir den entsprechenden Parametern aus.

Öffentliche Attribute

- funcElement ∗ firstFunc

  Pointer auf das erste Listenelement.
- funcElement ∗ lastFunc

  Pointer auf das letzte Listenelement.

Private Attribute

- Axis axis

  Spielstange, welche der Agent ansteuern soll.
- KnowledgeBase ∗ pKB

  Zeiger auf die Wissensbasis, zur Kommunikation.

5.21.1 Ausführliche Beschreibung

Eine einfach verkettete Liste für Funktionspointer.

Definiert in Zeile 47 der Datei AxisAgent.h.

5.21.2 Beschreibung der Konstruktoren und Destruktoren

5.21.2.1 FuncList::FuncList (Axis *ax*, KnowledgeBase ∗ *kB*) [inline]

Konstruktor.

Definiert in Zeile 53 der Datei AxisAgent.h.

Benutzt firstFunc, lastFunc und NULL.

5.21.2.2 FuncList::∼FuncList ( ) [inline]

Destruktor.

Definiert in Zeile 61 der Datei AxisAgent.h.

Benutzt firstFunc, lastFunc, funcElement::next, NULL und funcElement::values.

5.21.2.3 FuncList::FuncList (Axis *ax*, KnowledgeBase ∗ *kB*) [inline]

Konstruktor.

Definiert in Zeile 53 der Datei AxisAgent_wobbeln.h.

Benutzt firstFunc, lastFunc und NULL.

5.21.2.4 FuncList::∼FuncList ( ) [inline]

Destruktor.

Definiert in Zeile 61 der Datei AxisAgent_wobbeln.h.

Benutzt firstFunc, lastFunc, funcElement::next, NULL und funcElement::values.

5.21.3 Dokumentation der Elementfunktionen

5.21.3.1 void FuncList::attach (moveFunc *func*, float *values[]*, int *arraysize*, bool *blocking* = true, int *timeout* = 1000) [inline]

Fügt einen Funktionspointer in die Liste hinzu.

Parameter

in	*func*	Funktionspointer
in	*values*	Parameter für die Funktion
in	*arraysize*	Anzahl der Parameter für die Funktion
in	*blocking*	bei true wird die Funktion ausgeführt bis die Zielposition erreicht wurde, ansonsten kehrt die Funktion gleich nach Senden des Befehls zurück
in	*timeout*	wenn blocking true ist, wird die Funktion nach timeout Millisekunden abgebrochen

Definiert in Zeile 83 der Datei AxisAgent.h.

Benutzt funcElement::blocking, firstFunc, funcElement::func, lastFunc, funcElement::next, NULL, funcElement::timeout und funcElement::values.

Wird benutzt von AxisAgent::runAgent().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.21.3.2 void FuncList::attach (moveFunc *func*, float *values[]*, int *arraysize*, bool *blocking* = `true`, int *timeout* = `1000`) `[inline]`

Fügt einen Funktionspointer in die Liste hinzu.

Parameter

in	*func*	Funktionspointer
in	*values*	Parameter für die Funktion
in	*arraysize*	Anzahl der Parameter für die Funktion
in	*blocking*	bei true wird die Funktion ausgeführt bis die Zielposition erreicht wurde, ansonsten kehrt die Funktion gleich nach Senden des Befehls zurück
in	*timeout*	wenn blocking true ist, wird die Funktion nach timeout Millisekunden abgebrochen

Definiert in Zeile 83 der Datei AxisAgent_wobbeln.h.

Benutzt funcElement::blocking, firstFunc, funcElement::func, lastFunc, funcElement::next, NULL, funcElement::timeout und funcElement::values.

5.21.3.3 bool FuncList::execFunc (const funcElement * *funcEl*) `[inline]`

Führt die in der Strukrur übergebene Funktion mir den entsprechenden Parametern aus.

Definiert in Zeile 110 der Datei AxisAgent_wobbeln.h.

Benutzt axis, funcElement::blocking, funcElement::func, pKB, funcElement::timeout und funcElement::values.

5.21.3.4 bool FuncList::execFunc (const funcElement * *funcEl*) `[inline]`

Führt die in der Strukrur übergebene Funktion mir den entsprechenden Parametern aus.

Definiert in Zeile 110 der Datei AxisAgent.h.

Benutzt axis, funcElement::blocking, funcElement::func, pKB, funcElement::timeout und funcElement::values.

Wird benutzt von AxisAgent::runAgent().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.21.4 Dokumentation der Datenelemente

5.21.4.1 Axis FuncList::axis [private]

Spielstange, welche der Agent ansteuern soll.

Definiert in Zeile 127 der Datei AxisAgent.h.

Wird benutzt von execFunc().

5.21.4.2 funcElement * FuncList::firstFunc

Pointer auf das erste Listenelement.

Definiert in Zeile 117 der Datei AxisAgent.h.

Wird benutzt von attach(), FuncList(), AxisAgent::runAgent() und ~FuncList().

5.21.4.3 funcElement * FuncList::lastFunc

Pointer auf das letzte Listenelement.

Definiert in Zeile 121 der Datei AxisAgent.h.

Wird benutzt von attach(), FuncList() und ~FuncList().

5.21.4.4 KnowledgeBase * FuncList::pKB [private]

Zeiger auf die Wissensbasis, zur Kommunikation.

Definiert in Zeile 132 der Datei AxisAgent.h.

Wird benutzt von execFunc().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- AxisAgent.h
- AxisAgent_wobbeln.h

5.22 Gui Klassenreferenz

```
#include <Gui.h>
```

Öffentliche Methoden

- Gui (void)

  Konstruktor.
- ~Gui (void)

Destruktor.

5.22.1 Ausführliche Beschreibung

Definiert in Zeile 12 der Datei Gui.h.

5.22.2 Beschreibung der Konstruktoren und Destruktoren

5.22.2.1 Gui::Gui (void)

Konstruktor.

Definiert in Zeile 12 der Datei Gui.cpp.

5.22.2.2 Gui::~Gui (void)

Destruktor.

Definiert in Zeile 16 der Datei Gui.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

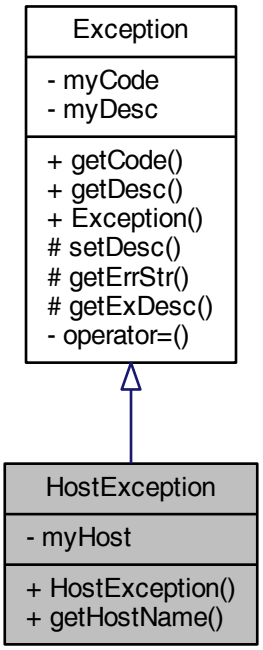
- Gui.h
- Gui.cpp

5.23 HostException Klassenreferenz

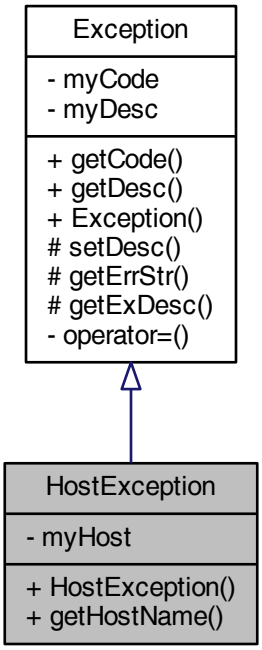
Exception-Klasse für die Host-Auflösung.

```
#include <Exception.h>
```

Klassendiagramm für HostException:



Zusammengehörigkeiten von HostException:



Öffentliche Methoden

- HostException (const char *host)

  Konstruktor.
- const char * getHostName () const

  gibt den Host zurück

Private Attribute

- const char * myHost

  Host.

5.23.1 Ausführliche Beschreibung

Exception-Klasse für die Host-Auflösung.

Definiert in Zeile 248 der Datei Exception.h.

5.23.2 Beschreibung der Konstruktoren und Destruktoren

5.23.2.1 HostException::HostException (const char * *host*) [inline]

Konstruktor.

Parameter

in	*host*	Host-Adresse bei der das Problem aufgetreten ist

Setzt den Exception-Code, die Eigenschaft und das Attribut.

Definiert in Zeile 257 der Datei Exception.h.

5.23.3 Dokumentation der Elementfunktionen

5.23.3.1 const char* HostException::getHostName ( ) const [inline]

gibt den Host zurück

Rückgabe

Host

Definiert in Zeile 266 der Datei Exception.h.

Benutzt myHost.

Wird benutzt von KnowledgeBase::KnowledgeBase().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.23.4 Dokumentation der Datenelemente

5.23.4.1 const char* HostException::myHost [private]

Host.

Definiert in Zeile 272 der Datei Exception.h.

Wird benutzt von getHostName().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

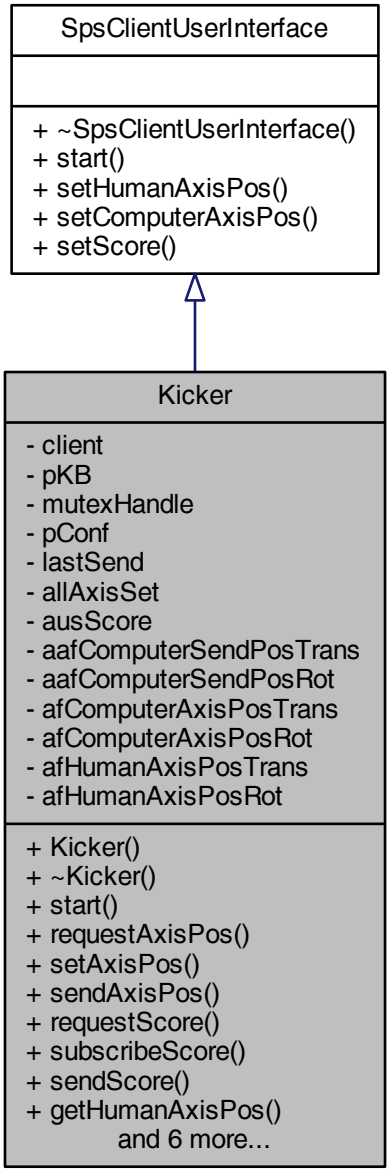
- Exception.h

5.24 Kicker Klassenreferenz

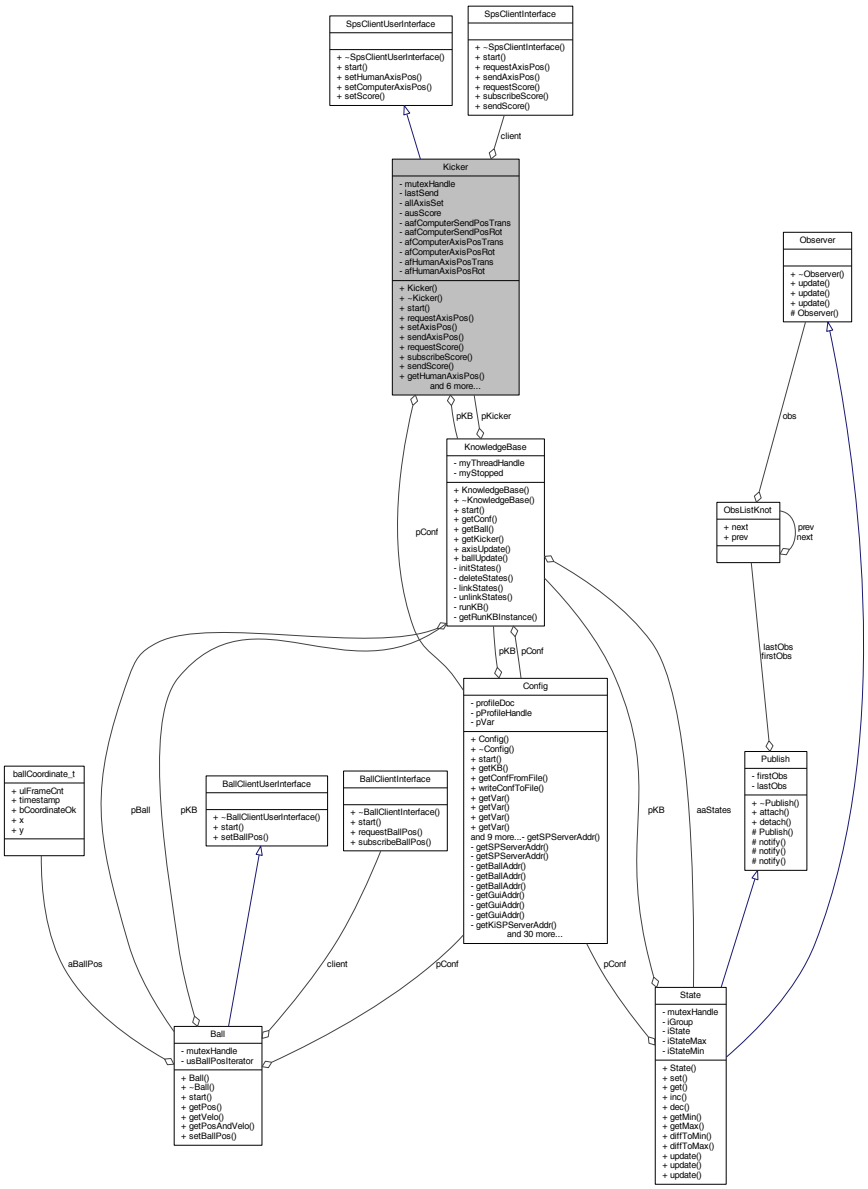
Eine Klasse, die den UDP-Client anlegt und benutzt.

```
#include <Kicker.h>
```

Klassendiagramm für Kicker:



Zusammengehörigkeiten von Kicker:



Öffentliche Methoden

- Kicker (Config *pConfig)

  Konstruktor.
- ~Kicker (void)

  Destruktor.
- void start (void)

  Startet den die Benutzung des Clients.
- bool requestAxisPos (void)

  Fragt die Spielstangenpositionen beim Server an.
- bool setAxisPos (unsigned int axis, float axisPos[2], float axisAngle[2])

  Setzt die Zielposition der Spielstangen.
- bool sendAxisPos (void)

Sendet die Spielstangenpositionen an den Server.

- bool requestScore (void)

  Fragt die Spielstangenpositionen beim Server an.
- bool subscribeScore (bool subscribe=true)

  Fragt die Spielstangenpositionen beim Server an.
- bool sendScore (unsigned short computer, unsigned short human)

  Fragt die Spielstangenpositionen beim Server an.
- bool getHumanAxisPos (unsigned int axis, float &axisPos, float &axisAngle)

  gibt eine Spielstangenposition des Menschen zurück
- bool getHumanAxisPos (float(&axisPos)[4], float(&axisAngle)[4])

  gibt die Spielstangenpositionen des Menschen zurück
- bool getComputerAxisPos (unsigned int axis, float &axisPos, float &axisAngle)

  gibt eine Spielstangenposition des Computers zurück
- bool getComputerAxisPos (float(&axisPos)[4], float(&axisAngle)[4])

  gibt die Spielstangenpositionen des Computers zurück
- bool getScore (unsigned short &computer, unsigned short &human)

  gibt den Spielstand zurück
- void setHumanAxisPos (float axisPos[4], float axisAngle[4])

  setzt die Spielstangenposition des Menschen
- void setComputerAxisPos (float axisPos[4], float axisAngle[4])

  setzt die Spielstangenposition des Computers
- void setScore (unsigned short computer, unsigned short human)

  setzt den Spielstand

Private Attribute

- SpsClientInterface * client

  Zeiger auf den Client.
- KnowledgeBase * pKB

  Zeiger auf die Wissensbasis, zur Kommunikation.
- HANDLE mutexHandle

  Mutex-Handle.
- Config * pConf

  Zugriff auf die Konfigurationen.
- clock_t lastSend

  Zeitpunkt zudem die Spielstangenpositionen das letzte mal gesendet wurden.
- bool allAxisSet [4]

  Signalisiert ob für jede Spielstange ein gültiger Wert gesetzt wurde.
- unsigned short ausScore [2]

  Spielstand, Computer und Mensch.
- float aafComputerSendPosTrans [2][4]

  Position und Geschwindigkeit der Translation des Computers, die an den Server gesendet wird, Spielstange 1-4.
- float aafComputerSendPosRot [2][4]

  Position und Geschwindigkeit der Rotation des Computers, die an den Server gesendet wird, Spielstange 1-4.
- float afComputerAxisPosTrans [4]

  Position der Translation des Computers, Spielstange 1-4.
- float afComputerAxisPosRot [4]

  Position der Rotation des Computers, Spielstange 1-4.
- float afHumanAxisPosTrans [4]

  Position der Translation des Menschen, Spielstange 1-4.
- float afHumanAxisPosRot [4]

  Position der Rotation des Menschen, Spielstange 1-4.

5.24.1 Ausführliche Beschreibung

Eine Klasse, die den UDP-Client anlegt und benutzt.

Beispielimplementierung wie der UDP-Client zu verwenden ist.

Definiert in Zeile 27 der Datei Kicker.h.

5.24.2 Beschreibung der Konstruktoren und Destruktoren

5.24.2.1 Kicker::Kicker (Config * *pConfig*)

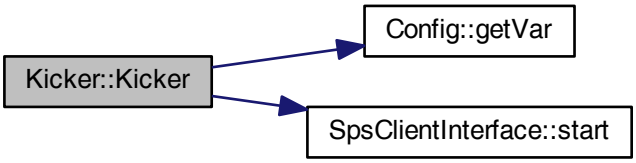
Konstruktor.

Legt die Klasse Client an.

Definiert in Zeile 21 der Datei Kicker.cpp.

Benutzt allAxisSet, AXIS_FOUR, AXIS_ONE, client, ERROR_CONFIG, ERROR_KICKER, Config::getVar(), KiSPServerPort, lastSend, mutexHandle, NULL, pConf, SPServerHost, SPServerPort und SpsClientInterface::start().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.24.2.2 Kicker::~Kicker (void)

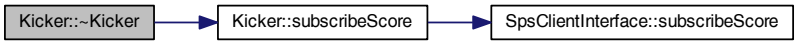
Destruktor.

Löscht die Klasse Client.

Definiert in Zeile 53 der Datei Kicker.cpp.

Benutzt client, mutexHandle und subscribeScore().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.24.3 Dokumentation der Elementfunktionen

5.24.3.1 bool Kicker::getComputerAxisPos (unsigned int *axis,* float & *axisPos,* float & *axisAngle*)

gibt eine Spielstangenposition des Computers zurück

Parameter

in	*axis*	Spielstange
out	*axisPos*	Translation
out	*axisAngle*	Rotation

Rückgabe

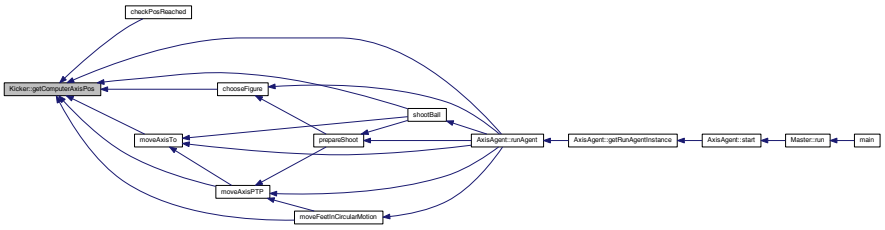
true, wenn gültige Werte zurückgegeben wurden

Definiert in Zeile 250 der Datei Kicker.cpp.

Benutzt afComputerAxisPosRot, afComputerAxisPosTrans, AXIS_FOUR, AXIS_ONE und mutexHandle.

Wird benutzt von checkPosReached(), chooseFigure(), moveAxisPTP(), moveAxisTo(), moveFeetInCircularMotion(), AxisAgent::runAgent() und shootBall().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.24.3.2 bool Kicker::getComputerAxisPos (float(&) *axisPos[4],* float(&) *axisAngle[4]*)

gibt die Spielstangenpositionen des Computers zurück

Parameter

out	*axisPos*	Translation
out	*axisAngle*	Rotation

Rückgabe

true, wenn gültige Werte zurückgegeben wurden

Definiert in Zeile 229 der Datei Kicker.cpp.

Benutzt afComputerAxisPosRot, afComputerAxisPosTrans, AXIS_FOUR und mutexHandle.

5.24.3.3 bool Kicker::getHumanAxisPos (unsigned int *axis,* float & *axisPos,* float & *axisAngle*)

gibt eine Spielstangenposition des Menschen zurück

Parameter

in	*axis*	Spielstange
out	*axisPos*	Translation
out	*axisAngle*	Rotation

Rückgabe

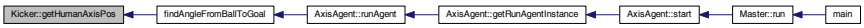
true, wenn gültige Werte zurückgegeben wurden

Definiert in Zeile 187 der Datei Kicker.cpp.

Benutzt afHumanAxisPosRot, afHumanAxisPosTrans, AXIS_FOUR, AXIS_ONE und mutexHandle.

Wird benutzt von findAngleFromBallToGoal().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.24.3.4 bool Kicker::getHumanAxisPos (float(&) *axisPos[4]*, float(&) *axisAngle[4]*)

gibt die Spielstangenpositionen des Menschen zurück

Parameter

out	*axisPos*	Translation
out	*axisAngle*	Rotation

Rückgabe

true, wenn gültige Werte zurückgegeben wurden

Definiert in Zeile 208 der Datei Kicker.cpp.

Benutzt afHumanAxisPosRot, afHumanAxisPosTrans, AXIS_FOUR und mutexHandle.

5.24.3.5 bool Kicker::getScore (unsigned short & *computer*, unsigned short & *human*)

gibt den Spielstand zurück

Parameter

out	*computer*	Spielstand des Computers
out	*human*	Spielstand des Menschen

Rückgabe

true, wenn gültige Werte zurückgegeben wurden

Definiert in Zeile 271 der Datei Kicker.cpp.

Benutzt ausScore, COMPUTER, HUMAN und mutexHandle.

5.24.3.6 bool Kicker::requestAxisPos (void)

Fragt die Spielstangenpositionen beim Server an.

Rückgabe

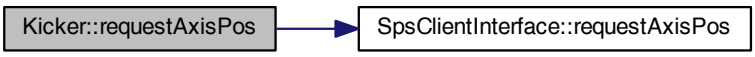
true wenn die Anfrage gesendet wurde

Definiert in Zeile 89 der Datei Kicker.cpp.

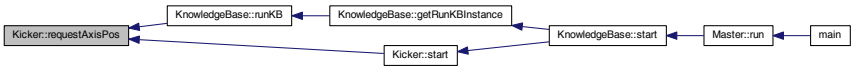
Benutzt client und SpsClientInterface::requestAxisPos().

Wird benutzt von KnowledgeBase::runKB() und start().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.24.3.7 bool Kicker::requestScore (void)

Fragt die Spielstangenpositionen beim Server an.

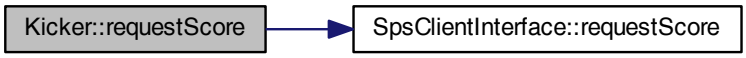
Rückgabe

true wenn die Anfrage gesendet wurde

Definiert in Zeile 155 der Datei Kicker.cpp.

Benutzt client und SpsClientInterface::requestScore().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.24.3.8 bool Kicker::sendAxisPos (void)

Sendet die Spielstangenpositionen an den Server.

Rückgabe

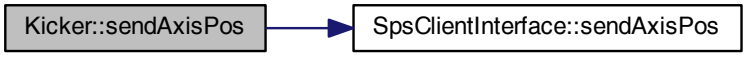
true wenn die Spielstangenpositionen gesendet wurde

Definiert in Zeile 137 der Datei Kicker.cpp.

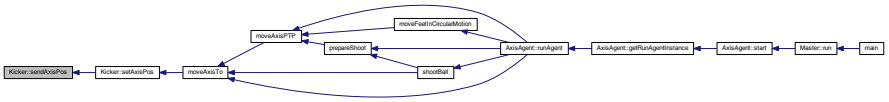
Benutzt aafComputerSendPosRot, aafComputerSendPosTrans, client, lastSend, mutexHandle und SpsClientInterface::sendAxisPos().

Wird benutzt von setAxisPos().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.24.3.9 bool Kicker::sendScore (unsigned short *computer,* unsigned short *human*)

Fragt die Spielstangenpositionen beim Server an.

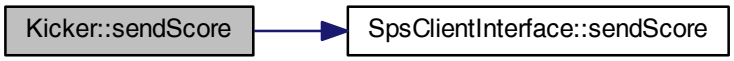
Rückgabe

true wenn die Anfrage gesendet wurde

Definiert in Zeile 169 der Datei Kicker.cpp.

Benutzt ausScore, client, COMPUTER, HUMAN, mutexHandle und SpsClientInterface::sendScore().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.24.3.10 bool Kicker::setAxisPos (unsigned int *axis,* float *axisPos[2],* float *axisAngle[2]*)

Setzt die Zielposition der Spielstangen.

Parameter

in	*axis*	Spielstanfe für die die Zielposition gilt
in	*axisPos*	Position und Geschwindigkeit der Translation des Computers
in	*axisAngle*	Position und Geschwindigkeit der Rotation des Computers

Rückgabe

true wenn die Position übernommen wurde

Noch zu erledigen Bedingung mit allAxisSet[] überdenken!!! Führt manchmal zu seltsamen Verhalten.

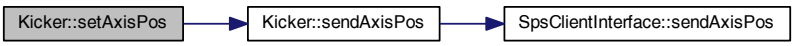
Setzt die Zielposition der Spielstangen und sendet diese an den Server wenn sich die Werte genügend geändert haben und eine gewisse Zeit seit dem letzten Senden der Positionen vergangen ist.

Definiert in Zeile 96 der Datei Kicker.cpp.

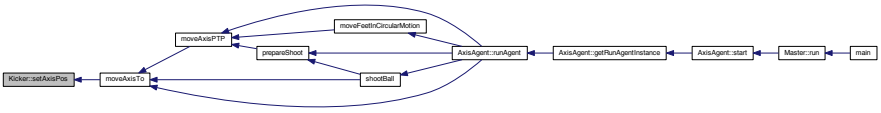
Benutzt aafComputerSendPosRot, aafComputerSendPosTrans, allAxisSet, AXIS_FOUR, AXIS_ONE, lastSend, mutexHandle, sendAxisPos(), VALUE und VELOCITY.

Wird benutzt von moveAxisTo().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.24.3.11 void Kicker::setComputerAxisPos (float *axisPos[4]*, float *axisAngle[4]*) [virtual]

setzt die Spielstangenposition des Computers

Parameter

in	*axisPos*	Translation die vom Client empfangen wurde
in	*axisAngle*	Rotation die vom Client empfangen wurde

Wird vom Client aufgerufen wenn eine Spielstangenposition vom Server eingetroffen ist.

Implementiert SpsClientUserInterface.

Definiert in Zeile 326 der Datei Kicker.cpp.

Benutzt afComputerAxisPosRot, afComputerAxisPosTrans, AXIS_FOUR, KnowledgeBase::axisUpdate(), COMPUTER, mutexHandle und pKB.

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.24.3.12 void Kicker::setHumanAxisPos (float *axisPos[4]*, float *axisAngle[4]*) [virtual]

setzt die Spielstangenposition des Menschen

Parameter

in	*axisPos*	Translation die vom Client empfangen wurde
in	*axisAngle*	Rotation die vom Client empfangen wurde

Wird vom Client aufgerufen wenn eine Spielstangenposition vom Server eingetroffen ist.

Implementiert SpsClientUserInterface.

Definiert in Zeile 289 der Datei Kicker.cpp.

Benutzt afHumanAxisPosRot, afHumanAxisPosTrans, AXIS_FOUR, KnowledgeBase::axisUpdate(), HUMAN, mutexHandle und pKB.

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.24.3.13 void Kicker::setScore (unsigned short *computer*, unsigned short *human*) [virtual]

setzt den Spielstand

Parameter

in	*computer*	Spielstand des Computers
in	*human*	Spielstand des Menschen

Wird vom Client aufgerufen wenn ein Spielstand vom Server eingetroffen ist.

Implementiert SpsClientUserInterface.

Definiert in Zeile 363 der Datei Kicker.cpp.

Benutzt ausScore, COMPUTER, HUMAN und mutexHandle.

5.24.3.14 void Kicker::start (void) [virtual]

Startet den die Benutzung des Clients.

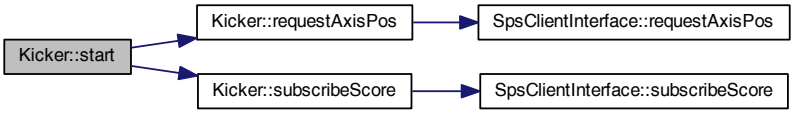
Implementiert SpsClientUserInterface.

Definiert in Zeile 73 der Datei Kicker.cpp.

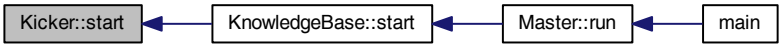
Benutzt requestAxisPos() und subscribeScore().

Wird benutzt von KnowledgeBase::start().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.24.3.15 bool Kicker::subscribeScore (bool *subscribe* = true)

Fragt die Spielstangenpositionen beim Server an.

Rückgabe

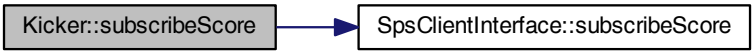
true wenn die Anfrage gesendet wurde

Definiert in Zeile 162 der Datei Kicker.cpp.

Benutzt client und SpsClientInterface::subscribeScore().

Wird benutzt von start() und ~Kicker().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.24.4 Dokumentation der Datenelemente

5.24.4.1 float Kicker::aafComputerSendPosRot[2][4] [private]

Position und Geschwindigkeit der Rotation des Computers, die an den Server gesendet wird, Spielstange 1-4.

Definiert in Zeile 218 der Datei Kicker.h.

Wird benutzt von sendAxisPos() und setAxisPos().

5.24.4.2 float Kicker::aafComputerSendPosTrans[2][4] [private]

Position und Geschwindigkeit der Translation des Computers, die an den Server gesendet wird, Spielstange 1-4.

Definiert in Zeile 213 der Datei Kicker.h.

Wird benutzt von sendAxisPos() und setAxisPos().

5.24.4.3 float Kicker::afComputerAxisPosRot[4] [private]

Position der Rotation des Computers, Spielstange 1-4.

Definiert in Zeile 228 der Datei Kicker.h.

Wird benutzt von getComputerAxisPos() und setComputerAxisPos().

5.24.4.4 float Kicker::afComputerAxisPosTrans[4] [private]

Position der Translation des Computers, Spielstange 1-4.

Definiert in Zeile 223 der Datei Kicker.h.

Wird benutzt von getComputerAxisPos() und setComputerAxisPos().

5.24.4.5 float Kicker::afHumanAxisPosRot[4] [private]

Position der Rotation des Menschen, Spielstange 1-4.

Definiert in Zeile 238 der Datei Kicker.h.

Wird benutzt von getHumanAxisPos() und setHumanAxisPos().

5.24.4.6 float Kicker::afHumanAxisPosTrans[4] [private]

Position der Translation des Menschen, Spielstange 1-4.

Definiert in Zeile 233 der Datei Kicker.h.

Wird benutzt von getHumanAxisPos() und setHumanAxisPos().

5.24.4.7 bool Kicker::allAxisSet[4] [private]

Signalisiert ob für jede Spielstange ein gültiger Wert gesetzt wurde.

Dies ist nötig da 4 Threads asynchron diese Klasse benutzen.

Definiert in Zeile 203 der Datei Kicker.h.

Wird benutzt von Kicker() und setAxisPos().

5.24.4.8 unsigned short Kicker::ausScore[2] [private]

Spielstand, Computer und Mensch.

Definiert in Zeile 208 der Datei Kicker.h.

Wird benutzt von getScore(), sendScore() und setScore().

5.24.4.9 SpsClientInterface* Kicker::client [private]

Zeiger auf den Client.

Definiert in Zeile 171 der Datei Kicker.h.

Wird benutzt von Kicker(), requestAxisPos(), requestScore(), sendAxisPos(), sendScore(), subscribeScore() und ~Kicker().

5.24.4.10 clock_t Kicker::lastSend [private]

Zeitpunkt zudem die Spielstangenpositionen das letzte mal gesendet wurden.

Definiert in Zeile 196 der Datei Kicker.h.

Wird benutzt von Kicker(), sendAxisPos() und setAxisPos().

5.24.4.11 HANDLE Kicker::mutexHandle [private]

Mutex-Handle.

Noch zu erledigen Benutzung vom Mutex beschreiben

Mutex-Handle zum sicheren Datenzugriff da die Funktionen aus ClientUserInterface (ausser start()) von einem anderen Thread aufgerugen werden.

Definiert in Zeile 185 der Datei Kicker.h.

Wird benutzt von getComputerAxisPos(), getHumanAxisPos(), getScore(), Kicker(), sendAxisPos(), sendScore(), setAxisPos(), setComputerAxisPos(), setHumanAxisPos(), setScore() und ~Kicker().

5.24.4.12 Config* Kicker::pConf [private]

Zugriff auf die Konfigurationen.

Siehe auch

Config

Definiert in Zeile 191 der Datei Kicker.h.

Wird benutzt von Kicker().

5.24.4.13 KnowledgeBase* Kicker::pKB [private]

Zeiger auf die Wissensbasis, zur Kommunikation.

Definiert in Zeile 176 der Datei Kicker.h.

Wird benutzt von setComputerAxisPos() und setHumanAxisPos().

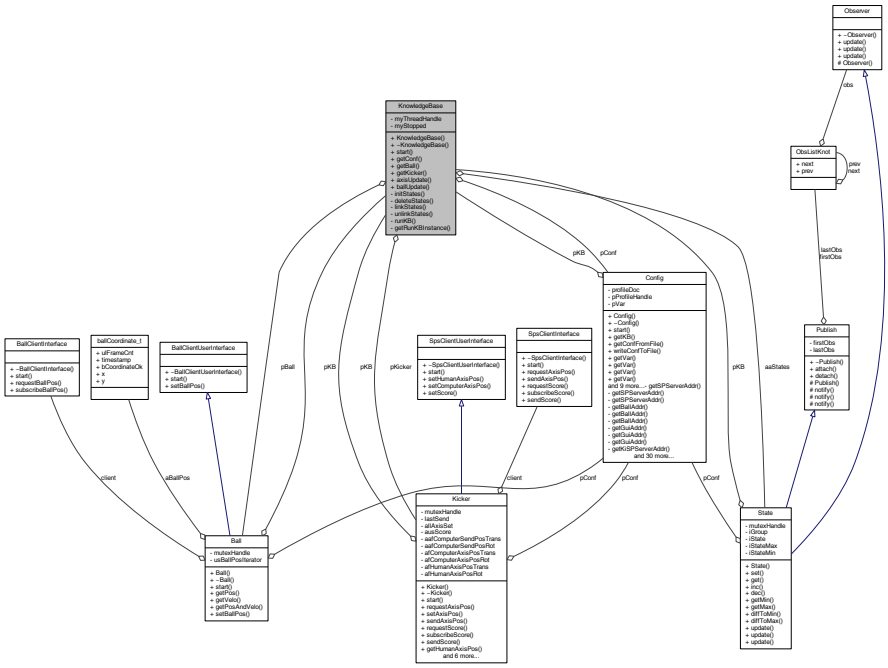
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Kicker.h
- Kicker.cpp

5.25 KnowledgeBase Klassenreferenz

`#include <KnowledgeBase.h>`

Zusammengehörigkeiten von KnowledgeBase:



Öffentliche Methoden

- KnowledgeBase (void)

  Konstruktor.
- ~KnowledgeBase (void)

  Destruktor.
- void start (void)

  Startet das System.
- Config * getConf (void)

  Gibt den Zeiger auf die Konfiguration zurück.
- Ball * getBall (void)

  Gibt den Zeiger auf die Ballerkennung zurück.
- Kicker * getKicker (void)

Gibt den Zeiger auf die SPS zurück.

- void axisUpdate (Gamer gamer, Axis axis)

  Wird aufgerufen wenn eine neue Spielstangenposition vorliegt.
- void ballUpdate (void)

  Wird aufgerufen wenn eine neue Ballposition vorliegt.

Öffentliche Attribute

- State ** aaStates [STATES_ALL]

  Array mit Zeigern auf State-Arrays.

Private Methoden

- void initStates (void)

  Setzt die Initialwerte der Statusse.
- void deleteStates (void)

  Löscht die Statusse.
- void linkStates (void)

  Verbindet die Statusse miteinander, so dass sie sich gegenseitig aktualisieren.
- void unlinkStates (void)

  Kappt die Verbindungen der Statusse zueinander.
- DWORD runKB (void)

  Eigener Thread, welcher die Daten verwaltet.

Private, statische Methoden

- static DWORD WINAPI getRunKBInstance (void *instance)

  Gibt den Funktionspointer auf die Funktion runKB() zurück.

Private Attribute

- Config * pConf

  Zeiger auf die Konfiguration des Systems.
- Ball * pBall

  Zeiger auf die Kommunikationsschnittstelle zur Ballerkennung.
- Kicker * pKicker

  Zeiger auf die Kommunikationsschnittstelle zur SPS.
- HANDLE myThreadHandle

  Handle vom Thread.
- bool myStopped

  Wird auf true gesetzt wenn der Agent beendet werden soll.

5.25.1 Ausführliche Beschreibung

Definiert in Zeile 29 der Datei KnowledgeBase.h.

5.25.2 Beschreibung der Konstruktoren und Destruktoren

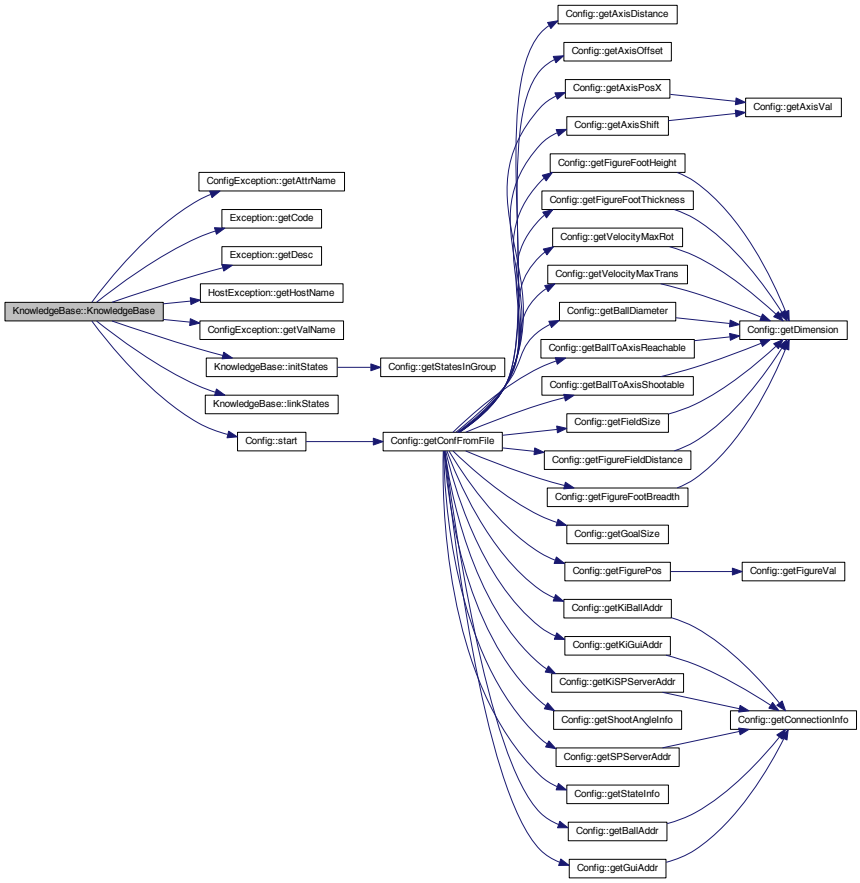
5.25.2.1 KnowledgeBase::KnowledgeBase (void)

Konstruktor.

Definiert in Zeile 20 der Datei KnowledgeBase.cpp.

Benutzt ConfigException::getAttrName(), Exception::getCode(), Exception::getDesc(), HostException::getHostName(), ConfigException::getValName(), initStates(), linkStates(), myStopped, pBall, pConf, pKicker und Config::start().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



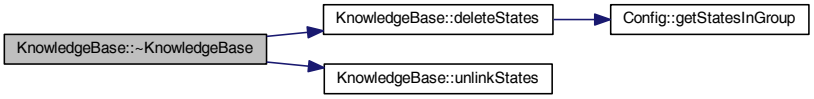
5.25.2.2 KnowledgeBase::~KnowledgeBase (void)

Destruktor.

Definiert in Zeile 54 der Datei KnowledgeBase.cpp.

Benutzt deleteStates(), myStopped, myThreadHandle, NULL, pBall, pConf, pKicker und unlinkStates().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.25.3 Dokumentation der Elementfunktionen

5.25.3.1 void KnowledgeBase::axisUpdate (Gamer *gamer*, Axis *axis*)

Wird aufgerufen wenn eine neue Spielstangenposition vorliegt.

Parameter

in	*gamer*	Spieler für den die neue Spielstangenposition gilt (Computer oder Mensch)
in	*axis*	Spielstange für die die neue Spielstangenposition gilt (1-4)

Siehe auch

Kicker

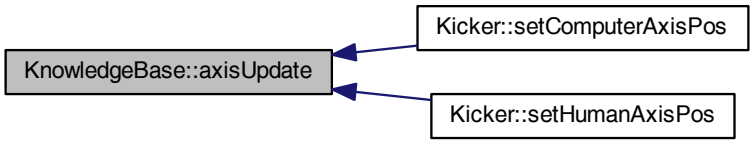
Noch zu erledigen Funktion umsetzen

Achtung, diese Funktion wird von einem anderen Thread aufgerufen!

Definiert in Zeile 144 der Datei KnowledgeBase.cpp.

Wird benutzt von Kicker::setComputerAxisPos() und Kicker::setHumanAxisPos().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.25.3.2 void KnowledgeBase::ballUpdate (void)

Wird aufgerufen wenn eine neue Ballposition vorliegt.

Siehe auch

Ball

Noch zu erledigen Funktion umsetzen

Achtung, diese Funktion wird von einem anderen Thread aufgerufen!

Definiert in Zeile 152 der Datei KnowledgeBase.cpp.

Wird benutzt von Ball::setBallPos().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.25.3.3 void KnowledgeBase::deleteStates (void) [private]

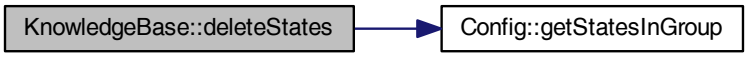
Löscht die Statusse.

Definiert in Zeile 177 der Datei KnowledgeBase.cpp.

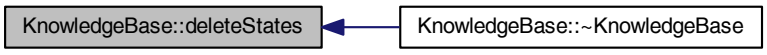
Benutzt aaStates, Config::getStatesInGroup(), NULL, pConf und STATES_ALL.

Wird benutzt von ~KnowledgeBase().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.25.3.4 Ball* KnowledgeBase::getBall (void) [inline]

Gibt den Zeiger auf die Ballerkennung zurück.

Rückgabe

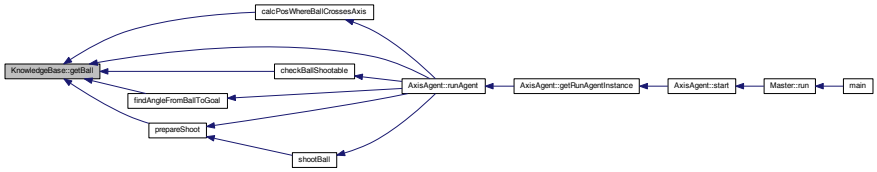
Zeiger auf die Ballerkennung

Definiert in Zeile 57 der Datei KnowledgeBase.h.

Benutzt pBall.

Wird benutzt von calcPosWhereBallCrossesAxis(), checkBallShootable(), findAngleFromBallToGoal(), prepareShoot() und AxisAgent::runAgent().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.25.3.5 Config* KnowledgeBase::getConf (void) [inline]

Gibt den Zeiger auf die Konfiguration zurück.

Rückgabe

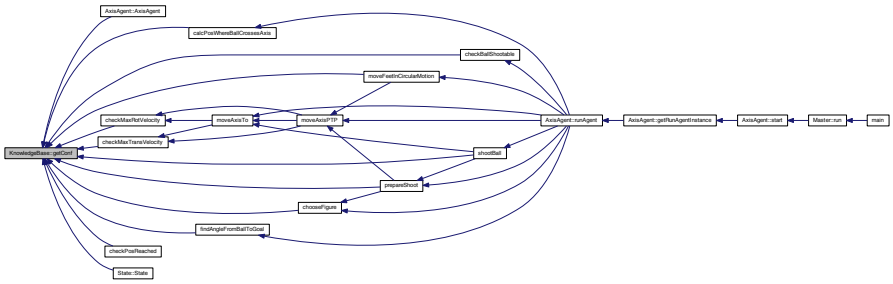
Zeiger auf die Konfiguration

Definiert in Zeile 51 der Datei KnowledgeBase.h.

Benutzt pConf.

Wird benutzt von AxisAgent::AxisAgent(), calcPosWhereBallCrossesAxis(), checkBallShootable(), checkMaxRotVelocity(), checkMaxTransVelocity(), checkPosReached(), chooseFigure(), findAngleFromBallToGoal(), moveFeetInCircularMotion(), prepareShoot(), shootBall() und State::State().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.25.3.6 Kicker* KnowledgeBase::getKicker (void) [inline]

Gibt den Zeiger auf die SPS zurück.

Rückgabe

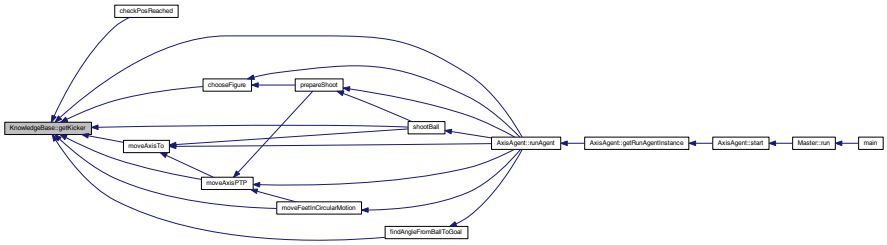
Zeiger auf die SPS

Definiert in Zeile 63 der Datei KnowledgeBase.h.

Benutzt pKicker.

Wird benutzt von checkPosReached(), chooseFigure(), findAngleFromBallToGoal(), moveAxisPTP(), moveAxisTo(), moveFeetInCircularMotion(), AxisAgent::runAgent() und shootBall().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.25.3.7 DWORD WINAPI KnowledgeBase::getRunKBInstance (void * *instance*) [static, private]

Gibt den Funktionspointer auf die Funktion runKB() zurück.

Parameter

in	*instance*	Pointer auf die Instanz dieser Klasse

Rückgabe

Funktionspointer auf die Funktion runKB() der aktuellen Instanz

Trick um einen neuen Thread mit einer Memberfunktion zu erzeugen.

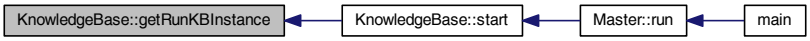
Definiert in Zeile 246 der Datei KnowledgeBase.cpp.

Benutzt NULL und runKB().

Wird benutzt von start().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.25.3.8 void KnowledgeBase::initStates (void) [private]

Setzt die Initialwerte der Statusse.

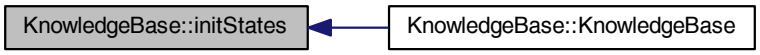
Definiert in Zeile 159 der Datei KnowledgeBase.cpp.

Benutzt aaStates, Config::getStatesInGroup(), NULL, pConf und STATES_ALL.

Wird benutzt von KnowledgeBase().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



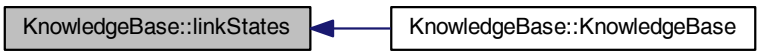
5.25.3.9 void KnowledgeBase::linkStates (void) [private]

Verbindet die Statusse miteinander, so dass sie sich gegenseitig aktualisieren.

Definiert in Zeile 198 der Datei KnowledgeBase.cpp.

Wird benutzt von KnowledgeBase().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.25.3.10 DWORD KnowledgeBase::runKB (void) [private]

Eigener Thread, welcher die Daten verwaltet.

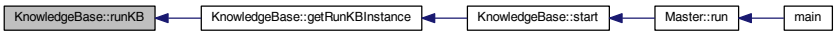
Definiert in Zeile 218 der Datei KnowledgeBase.cpp.

Benutzt myStopped, pKicker und Kicker::requestAxisPos().

Wird benutzt von getRunKBInstance().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.25.3.11 void KnowledgeBase::start (void)

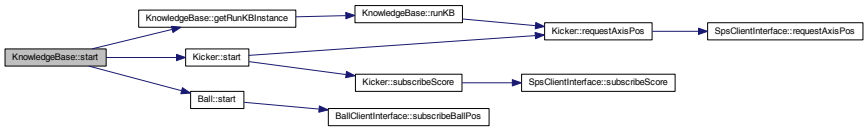
Startet das System.

Definiert in Zeile 118 der Datei KnowledgeBase.cpp.

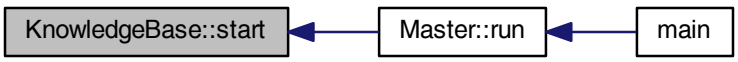
Benutzt getRunKBInstance(), myThreadHandle, NULL, pBall, pKicker, Kicker::start() und Ball::start().

Wird benutzt von Master::run().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



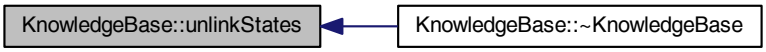
5.25.3.12 void KnowledgeBase::unlinkStates (void) [private]

Kappt die Verbindungen der Statusse zueinander.

Definiert in Zeile 214 der Datei KnowledgeBase.cpp.

Wird benutzt von ~KnowledgeBase().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.25.4 Dokumentation der Datenelemente

5.25.4.1 State** KnowledgeBase::aaStates[STATES_ALL]

Array mit Zeigern auf State-Arrays.

Definiert in Zeile 88 der Datei KnowledgeBase.h.

Wird benutzt von deleteStates() und initStates().

5.25.4.2 bool KnowledgeBase::myStopped [private]

Wird auf true gesetzt wenn der Agent beendet werden soll.

Siehe auch

KnowledgeBase::~KnowledgeBase(void)
KnowledgeBase::runKB(void)

Definiert in Zeile 120 der Datei KnowledgeBase.h.

Wird benutzt von KnowledgeBase(), runKB() und ~KnowledgeBase().

5.25.4.3 HANDLE KnowledgeBase::myThreadHandle [private]

Handle vom Thread.

Handle vom Thread welcher für die Verwaltung der Daten zuständig ist. Wird in KnowledgeBase::start() beim Erzeugen des Threads gesetzt. Wird benutzt um im Destruktor auf das Ende des Threads zu warten.

Definiert in Zeile 113 der Datei KnowledgeBase.h.

Wird benutzt von start() und ~KnowledgeBase().

5.25.4.4 Ball* KnowledgeBase::pBall [private]

Zeiger auf die Kommunikationsschnittstelle zur Ballerkennung.

Definiert in Zeile 99 der Datei KnowledgeBase.h.

Wird benutzt von getBall(), KnowledgeBase(), start() und ~KnowledgeBase().

5.25.4.5 Config* KnowledgeBase::pConf [private]

Zeiger auf die Konfiguration des Systems.

Definiert in Zeile 94 der Datei KnowledgeBase.h.

Wird benutzt von deleteStates(), getConf(), initStates(), KnowledgeBase() und ~KnowledgeBase().

5.25.4.6 Kicker* KnowledgeBase::pKicker [private]

Zeiger auf die Kommunikationsschnittstelle zur SPS.

Definiert in Zeile 104 der Datei KnowledgeBase.h.

Wird benutzt von getKicker(), KnowledgeBase(), runKB(), start() und ~KnowledgeBase().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- KnowledgeBase.h
- KnowledgeBase.cpp

5.26 KnowlegeBase Klassenreferenz

Diese Klasse stellt die Wissensbasis dar.

```
#include <KnowledgeBase.h>
```

5.26.1 Ausführliche Beschreibung

Diese Klasse stellt die Wissensbasis dar.

Diese Klasse stellt die Wissensbasis dar, welche alle Informationen verwaltet.

Definiert in Zeile 21 der Datei KnowledgeBase.h.

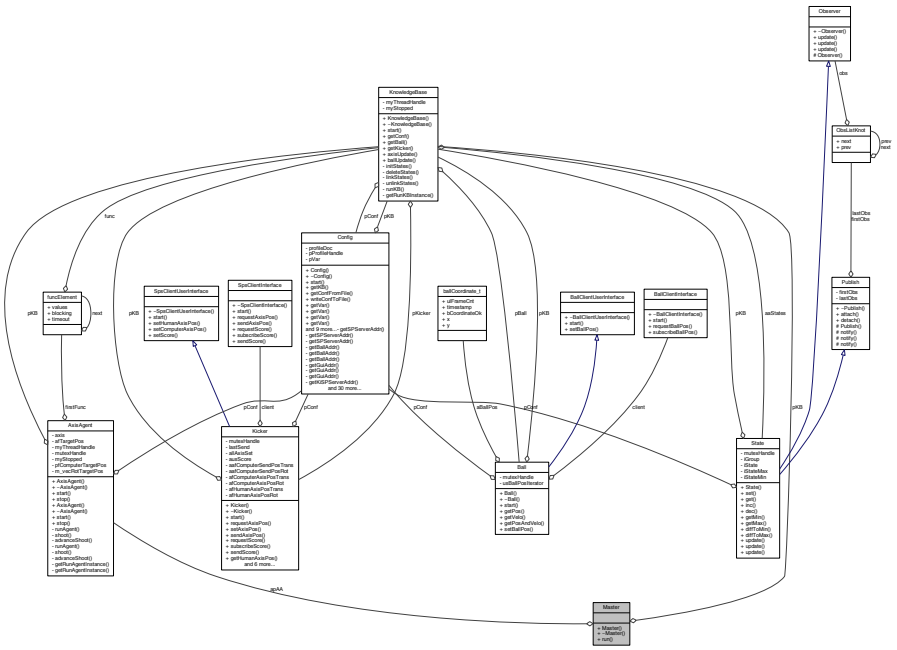
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- KnowledgeBase.h

5.27 Master Klassenreferenz

```
#include <Master.h>
```

Zusammengehörigkeiten von Master:



Öffentliche Methoden

- Master (void)

  Konstruktor.
- ~Master (void)

  Destruktor.
- void run (void)

  Startet das System.

Private Attribute

- KnowledgeBase * pKB

  Zeiger auf die Wissensbasis.
- AxisAgent * apAA [4]

  Zeiger auf die vier Agenten der jeweiligen Spielstange.

5.27.1 Ausführliche Beschreibung

Definiert in Zeile 18 der Datei Master.h.

5.27.2 Beschreibung der Konstruktoren und Destruktoren

5.27.2.1 Master::Master (void)

Konstruktor.

Definiert in Zeile 21 der Datei Master.cpp.

Benutzt apAA, AXIS_FOUR, AXIS_ONE und pKB.

5.27.2.2 Master::~Master (void)

Destruktor.

Definiert in Zeile 38 der Datei Master.cpp.

Benutzt apAA, AXIS_FOUR, AXIS_ONE und pKB.

5.27.3 Dokumentation der Elementfunktionen

5.27.3.1 void Master::run (void)

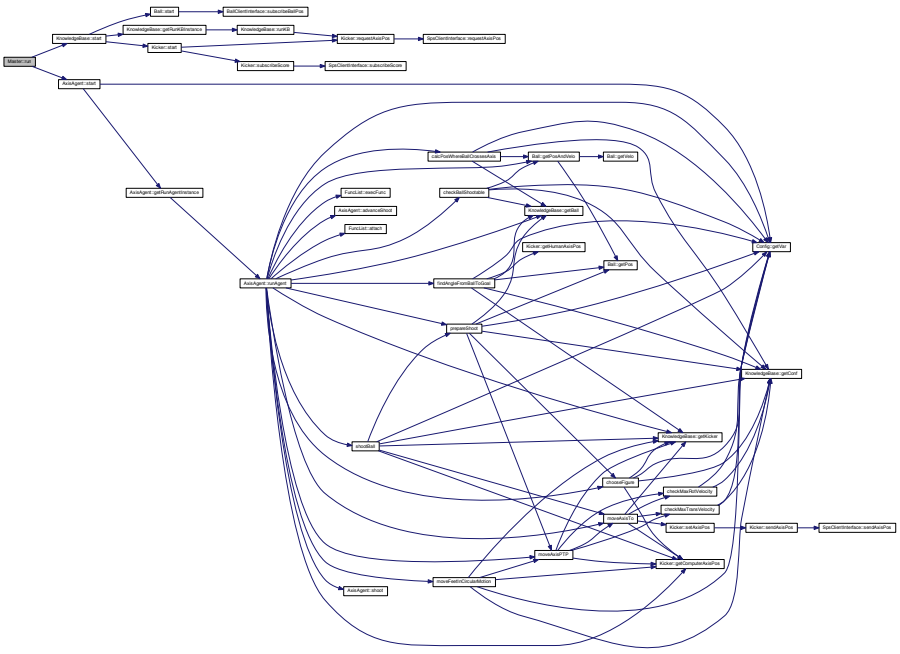
Startet das System.

Definiert in Zeile 57 der Datei Master.cpp.

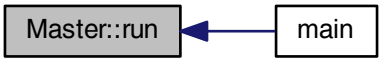
Benutzt apAA, AXIS_FOUR, AXIS_ONE, pKB, KnowledgeBase::start() und AxisAgent::start().

Wird benutzt von main().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.27.4 Dokumentation der Datenelemente

5.27.4.1 AxisAgent* Master::apAA[4] [private]

Zeiger auf die vier Agenten der jeweiligen Spielstange.

Definiert in Zeile 45 der Datei Master.h.

Wird benutzt von Master(), run() und ~Master().

5.27.4.2 KnowledgeBase* Master::pKB [private]

Zeiger auf die Wissensbasis.

Definiert in Zeile 40 der Datei Master.h.

Wird benutzt von Master(), run() und ~Master().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

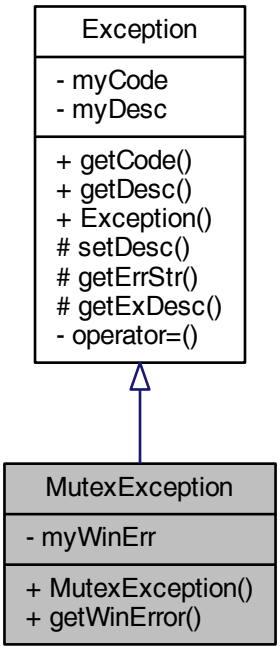
- Master.h
- Master.cpp

5.28 MutexException Klassenreferenz

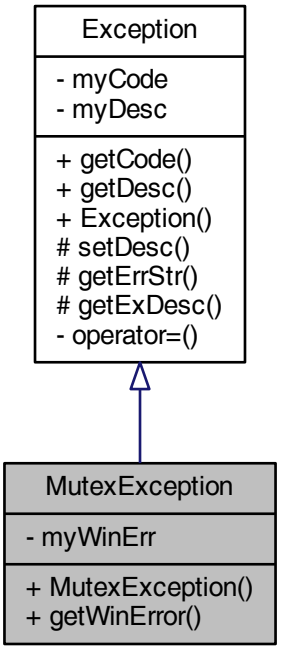
Exception-Klasse für die Mutexzugriffe.

`#include <Exception.h>`

Klassendiagramm für MutexException:



Zusammengehörigkeiten von MutexException:



Öffentliche Methoden

- MutexException (DWORD winErr)

  Konstruktor.
- DWORD getWinError ()

  gibt die Windows-Fehlermeldung zurück

Private Attribute

- DWORD myWinErr

  Eigenschaft.

5.28.1 Ausführliche Beschreibung

Exception-Klasse für die Mutexzugriffe.

Definiert in Zeile 217 der Datei Exception.h.

5.28.2 Beschreibung der Konstruktoren und Destruktoren

5.28.2.1 MutexException::MutexException (DWORD *winErr*) [inline]

Konstruktor.

Parameter

in	*winErr*	Windows-Fehlercode

Setzt den Exception-Code, die Eigenschaft und das Attribut.

Definiert in Zeile 226 der Datei Exception.h.

5.28.3 Dokumentation der Elementfunktionen

5.28.3.1 DWORD MutexException::getWinError ( ) [inline]

gibt die Windows-Fehlermeldung zurück

Rückgabe

Eigenschaft

Definiert in Zeile 235 der Datei Exception.h.

Benutzt myWinErr.

5.28.4 Dokumentation der Datenelemente

5.28.4.1 DWORD MutexException::myWinErr [private]

Eigenschaft.

Definiert in Zeile 241 der Datei Exception.h.

Wird benutzt von getWinError().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

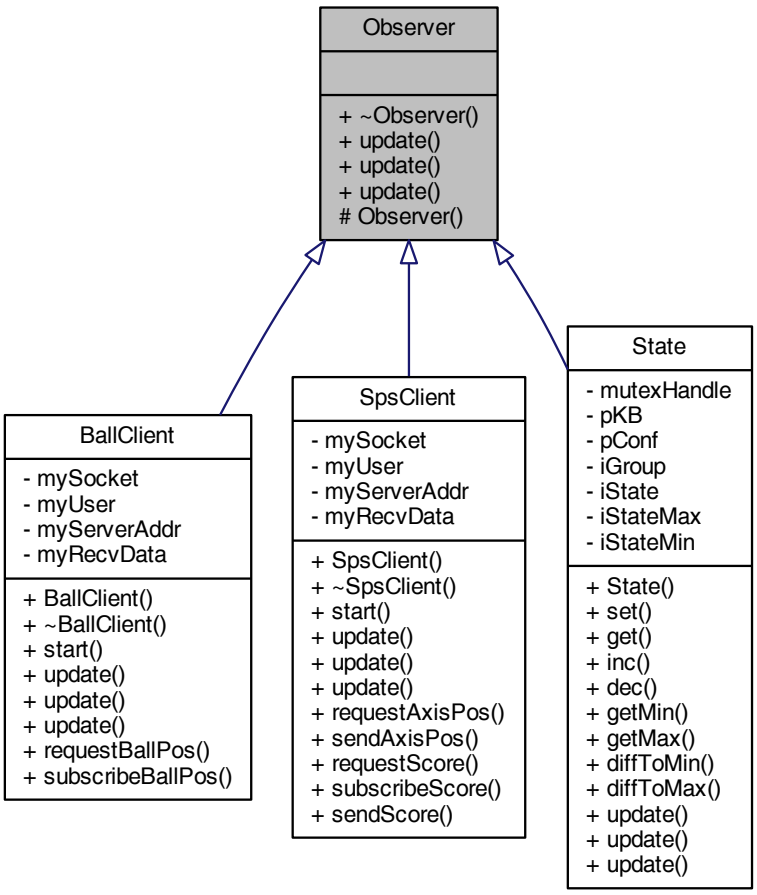
- Exception.h

5.29 Observer Klassenreferenz

Observer Klasse für das Observer Entwurfsmodell.

```
#include <observer.h>
```

Klassendiagramm für Observer:



Öffentliche Methoden

- virtual ~Observer ()

  Destruktor.
- virtual void update (Publish *pub)=0

  Wird von Publish aufgerufen wenn ein Ereignis eingetreten ist.
- virtual void update (Publish *pub, int *iValues, unsigned int iValuesSize)=0

  Wird von Publish aufgerufen wenn ein Ereignis eingetreten ist.
- virtual void update (Publish *pub, float *fValues, unsigned int fValuesSize, int *iValues=NULL, unsigned int iValuesSize=0)=0

  Wird von Publish aufgerufen wenn ein Ereignis eingetreten ist.

Geschützte Methoden

- Observer ()

  Konstruktor.

5.29.1 Ausführliche Beschreibung

Observer Klasse für das Observer Entwurfsmodell.

Siehe auch

Publish

Definiert in Zeile 20 der Datei observer.h.

5.29.2 Beschreibung der Konstruktoren und Destruktoren

5.29.2.1 virtual Observer::~Observer ( ) [inline, virtual]

Destruktor.

Definiert in Zeile 26 der Datei observer.h.

5.29.2.2 Observer::Observer ( ) [inline, protected]

Konstruktor.

Definiert in Zeile 56 der Datei observer.h.

5.29.3 Dokumentation der Elementfunktionen

5.29.3.1 virtual void Observer::update (Publish * *pub*) [pure virtual]

Wird von Publish aufgerufen wenn ein Ereignis eingetreten ist.

Parameter

in	*pub*	Publisher Objekt, welches über ein Ereignis benachichtigt

Implementiert in State, BallClient und SpsClient.

Wird benutzt von Publish::notify().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.29.3.2 virtual void Observer::update (Publish * *pub*, int * *iValues*, unsigned int *iValuesSize*) [pure virtual]

Wird von Publish aufgerufen wenn ein Ereignis eingetreten ist.

Parameter

in	*pub*	Publisher Objekt, welches über ein Ereignis benachichtigt
in	*iValues*	int-Array zum übergeben von Variabeln
in	*iValuesSize*	Größe des Array values

Implementiert in State, BallClient und SpsClient.

5.29.3.3 virtual void Observer::update (Publish * *pub*, float * *fValues*, unsigned int *fValuesSize*, int * *iValues* = NULL, unsigned int *iValuesSize* = 0) [pure virtual]

Wird von Publish aufgerufen wenn ein Ereignis eingetreten ist.

Parameter

in	*pub*	Publisher Objekt, welches über ein Ereignis benachichtigt
in	*fValues*	float Array zum übergeben von Variabeln
in	*fValuesSize*	Größe des Array values
in	*iValues*	int-Array zum übergeben von Variabeln
in	*iValuesSize*	Größe des Array values

Implementiert in State, BallClient und SpsClient.

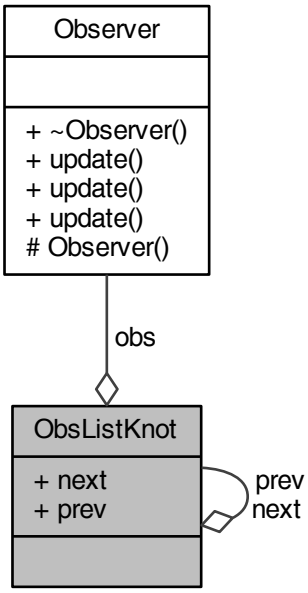
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- observer.h

5.30 ObsListKnot Strukturreferenz

`#include <publish.h>`

Zusammengehörigkeiten von ObsListKnot:



Öffentliche Attribute

- Observer * obs

Pointer auf das Observer Objekt.

- ObsListKnot * next

  Pointer auf das nächste Listenelement.

- ObsListKnot * prev

  Pointer auf das vorherige Listenelement.

5.30.1 Ausführliche Beschreibung

Definiert in Zeile 24 der Datei publish.h.

5.30.2 Dokumentation der Datenelemente

5.30.2.1 ObsListKnot* ObsListKnot::next

Pointer auf das nächste Listenelement.

Definiert in Zeile 29 der Datei publish.h.

Wird benutzt von Publish::attach(), Publish::detach() und Publish::notify().

5.30.2.2 Observer* ObsListKnot::obs

Pointer auf das Observer Objekt.

Definiert in Zeile 27 der Datei publish.h.

Wird benutzt von Publish::attach(), Publish::detach() und Publish::notify().

5.30.2.3 ObsListKnot* ObsListKnot::prev

Pointer auf das vorherige Listenelement.

Definiert in Zeile 31 der Datei publish.h.

Wird benutzt von Publish::attach() und Publish::detach().

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

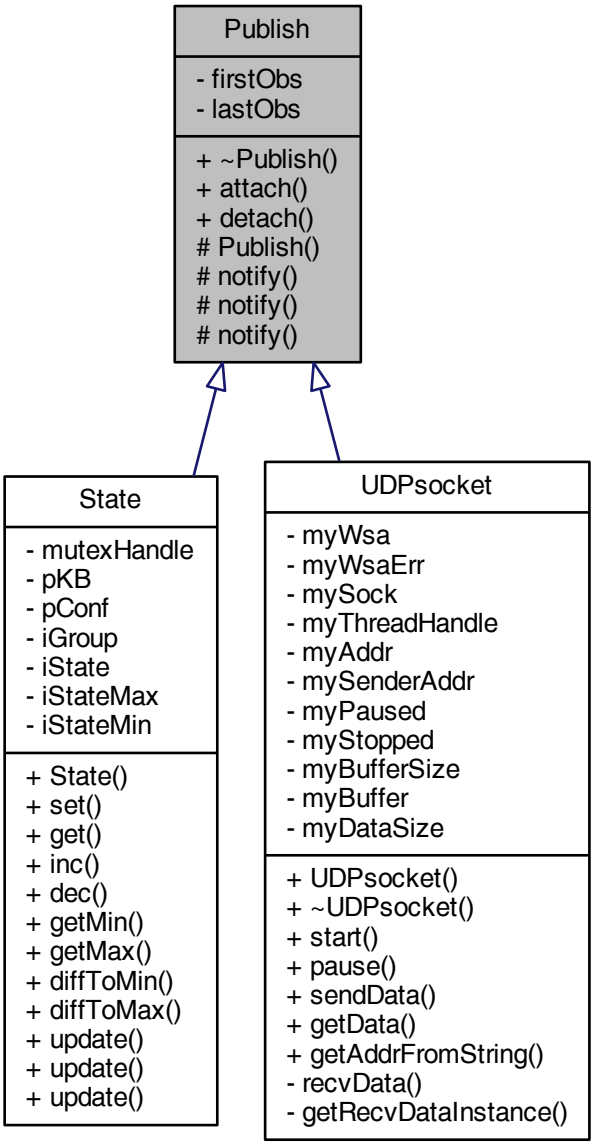
- publish.h

5.31 Publish Klassenreferenz

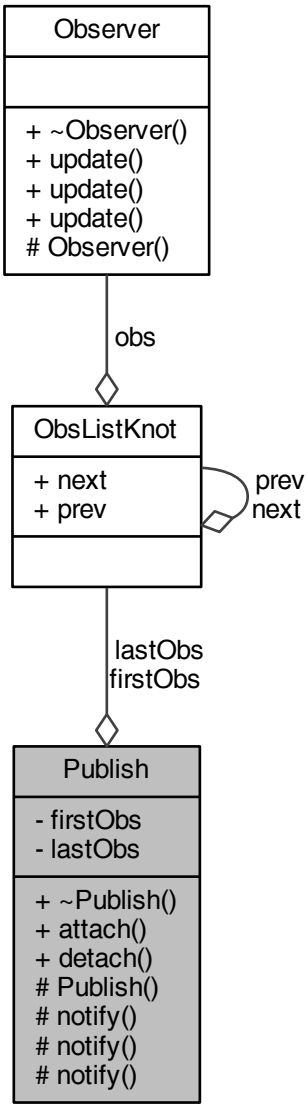
Publisher Klasse für das Observer Entwurfsmodell.

```
#include <publish.h>
```

Klassendiagramm für Publish:



Zusammengehörigkeiten von Publish:



Öffentliche Methoden

- virtual ~Publish ()

  Destruktor.

- virtual void attach (Observer *obs)

  Fügt ein Observer Objekt in die Liste der zu benachrichtigenden Objekte hinzu.

- virtual bool detach (Observer *obs)

  Entfernt das Observer Objekt aus der Liste der zu benachichtigenden Objekte.

Geschützte Methoden

- Publish ()

  Konstruktor.
- virtual void notify (void)

  Benachichtigt alle eingetragenen Observer Objekte.
- virtual void notify (int ∗iValues, unsigned int iValuesSize)

  Benachichtigt alle eingetragenen Observer Objekte.
- virtual void notify (float ∗fValues, unsigned int fValuesSize, int ∗iValues=NULL, unsigned int iValuesSize=0)

  Benachichtigt alle eingetragenen Observer Objekte.

Private Attribute

- ObsListKnot_t ∗ firstObs

  Pointer auf das erste Listenelement.
- ObsListKnot_t ∗ lastObs

  Pointer auf das letzte Listenelement.

5.31.1 Ausführliche Beschreibung

Publisher Klasse für das Observer Entwurfsmodell.

Siehe auch

Observer

Definiert in Zeile 39 der Datei publish.h.

5.31.2 Beschreibung der Konstruktoren und Destruktoren

5.31.2.1 virtual Publish::∼Publish ( ) [inline, virtual]

Destruktor.

Definiert in Zeile 45 der Datei publish.h.

5.31.2.2 Publish::Publish ( ) [inline, protected]

Konstruktor.

Da die Liste noch leer ist wird das erste und letzte auf NULL gesetzt.

Definiert in Zeile 133 der Datei publish.h.

Benutzt firstObs, lastObs und NULL.

5.31.3 Dokumentation der Elementfunktionen

5.31.3.1 virtual void Publish::attach (Observer ∗ *obs*) [inline, virtual]

Fügt ein Observer Objekt in die Liste der zu benachrichtigenden Objekte hinzu.

Parameter

in	*obs*	Pointer auf das in die Liste einzutragende Observer Element

Definiert in Zeile 53 der Datei publish.h.

Benutzt firstObs, lastObs, ObsListKnot::next, NULL, ObsListKnot::obs und ObsListKnot::prev.

Wird benutzt von BallClient::start().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.31.3.2 virtual bool Publish::detach (Observer * *obs*) [inline, virtual]

Entfernt das Observer Objekt aus der Liste der zu benachichtigenden Objekte.

Parameter

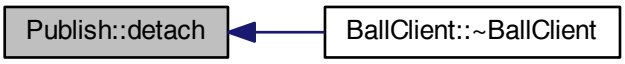
in	*obs*	Pointer auf das aus der Liste zu entfernende Observer Element

Definiert in Zeile 80 der Datei publish.h.

Benutzt firstObs, lastObs, ObsListKnot::next, NULL, ObsListKnot::obs und ObsListKnot::prev.

Wird benutzt von BallClient::~BallClient().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.31.3.3 virtual void Publish::notify (void) [inline, protected, virtual]

Benachichtigt alle eingetragenen Observer Objekte.

Siehe auch

Observer::update()

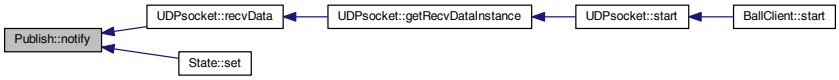
Definiert in Zeile 139 der Datei publish.h.

Benutzt firstObs, ObsListKnot::next, ObsListKnot::obs und Observer::update().

Wird benutzt von UDPsocket::recvData() und State::set().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.31.3.4 virtual void Publish::notify (int * *iValues*, unsigned int *iValuesSize*) [inline, protected, virtual]

Benachichtigt alle eingetragenen Observer Objekte.

Siehe auch

Observer::update()

Definiert in Zeile 156 der Datei publish.h.

Benutzt firstObs, ObsListKnot::next, ObsListKnot::obs und Observer::update().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.31.3.5 virtual void Publish::notify (float * *fValues*, unsigned int *fValuesSize*, int * *iValues* = NULL, unsigned int *iValuesSize* = 0) [inline, protected, virtual]

Benachichtigt alle eingetragenen Observer Objekte.

Siehe auch

Observer::update()

Definiert in Zeile 173 der Datei publish.h.

Benutzt firstObs, ObsListKnot::next, ObsListKnot::obs und Observer::update().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.31.4 Dokumentation der Datenelemente

5.31.4.1 ObsListKnot_t* Publish::firstObs [private]

Pointer auf das erste Listenelement.

Definiert in Zeile 193 der Datei publish.h.

Wird benutzt von attach(), detach(), notify() und Publish().

5.31.4.2 ObsListKnot_t* Publish::lastObs [private]

Pointer auf das letzte Listenelement.

Definiert in Zeile 198 der Datei publish.h.

Wird benutzt von attach(), detach() und Publish().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

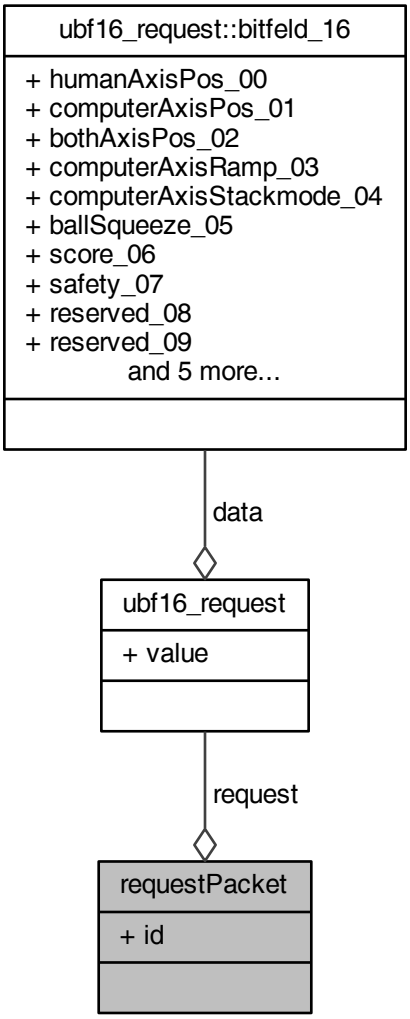
- publish.h

5.32 requestPacket Strukturreferenz

Anfrage-Paket für bestimmte Daten.

`#include <UdpDefines.h>`

Zusammengehörigkeiten von requestPacket:



Öffentliche Attribute

- unsigned int id: 16
- ubf16_request request

5.32.1 Ausführliche Beschreibung

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein Ánfrage-Bitfeld ubf8_request.

Definiert in Zeile 128 der Datei UdpDefines.h.

5.32.2 Dokumentation der Datenelemente

5.32.2.1 unsigned int requestPacket::id

Definiert in Zeile 130 der Datei UdpDefines.h.

Wird benutzt von SpsClient::requestAxisPos() und SpsClient::requestScore().

5.32.2.2 ubf16_request requestPacket::request

Definiert in Zeile 131 der Datei UdpDefines.h.

Wird benutzt von SpsClient::requestAxisPos() und SpsClient::requestScore().

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

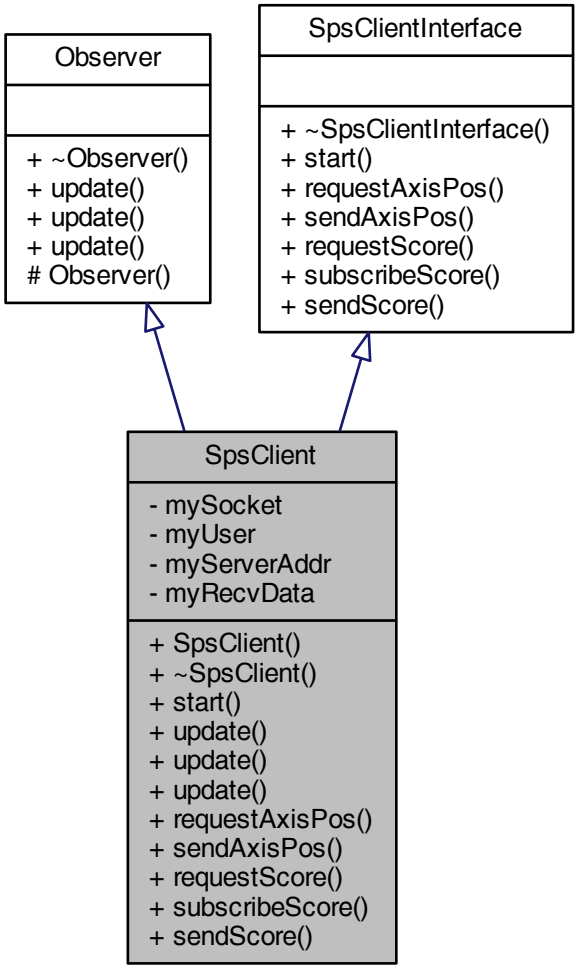
- UdpDefines.h

5.33 SpsClient Klassenreferenz

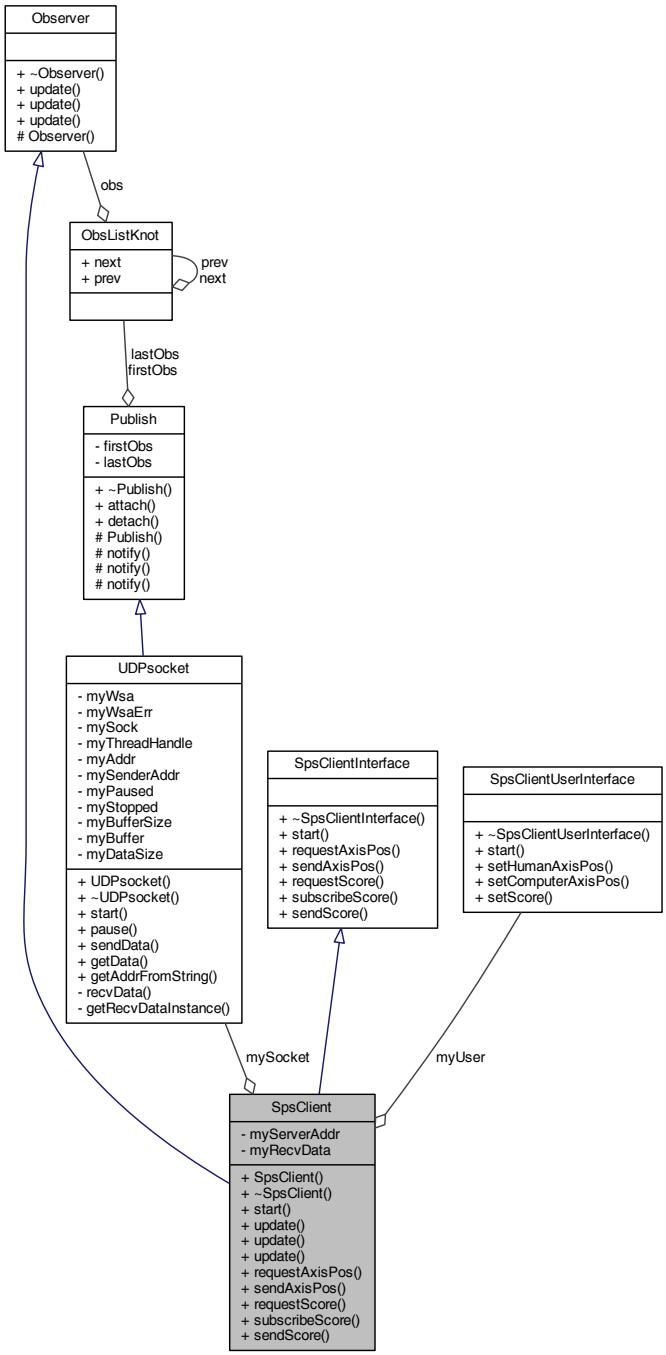
ein UDP-Client

```
#include <SpsClient.h>
```

Klassendiagramm für SpsClient:



Zusammengehörigkeiten von SpsClient:



Öffentliche Methoden

- SpsClient (unsigned short localPort, const char ∗serverHost, unsigned short serverPort)

  Konstruktor.
- ∼SpsClient ()

  Destruktor.
- bool start (SpsClientUserInterface ∗userInterface)

Startet den Client.

- void update (Publish ∗pub)

  update-Funktion der Observer Klasse
- void update (Publish ∗pub, int ∗iValues, unsigned int iValuesSize)

  update-Funktion der Observer Klasse
- void update (Publish ∗pub, float ∗fValues, unsigned int fValuesSize, int ∗iValues=NULL, unsigned int iValues-Size=0)

  update-Funktion der Observer Klasse
- bool requestAxisPos ()

  Fragt die Spielstangenpositionen beim Server an.
- bool sendAxisPos (float axisPos[2][4], float axisAngle[2][4])

  Sendet die Spielstangenpositionen an den Server.
- bool requestScore ()

  Fragt die Spielstangenpositionen beim Server an.
- bool subscribeScore (bool subs=true)

  Fragt die Spielstangenpositionen beim Server an.
- bool sendScore (unsigned short computer, unsigned short human)

  Fragt die Spielstangenpositionen beim Server an.

Private Attribute

- UDPsocket ∗ mySocket

  der UDP-Socket
- SpsClientUserInterface ∗ myUser

  Zeiger auf den Benutzer dieser Klasse.
- sockaddr_in myServerAddr

  Adresse des Servers.
- char ∗ myRecvData

  Speicher für die empfangenen Daten.

5.33.1 Ausführliche Beschreibung

ein UDP-Client

Beispielimplementierung eines UDP-Client unter Verwendung der Klasse UDPsocket.

Definiert in Zeile 26 der Datei SpsClient.h.

5.33.2 Beschreibung der Konstruktoren und Destruktoren

5.33.2.1 SpsClient::SpsClient (unsigned short *localPort*, const char ∗ *serverHost*, unsigned short *serverPort*)

Konstruktor.

Parameter

in	*localPort*	eigenen IP-Port
in	*serverHost*	Rechnername oder IP-Adresse des Servers
in	*serverPort*	Port des Servers

Ermittelt die Server-Adresse und legt den Empfangsspeicher und die Klasse UDPsocket an.

Definiert in Zeile 19 der Datei SpsClient.cpp.

Benutzt BUFFER_SIZE und NULL.

5.33.2.2 SpsClient::~SpsClient ( )

Destruktor.

Schließt den UDP-Socket und gibt den Empfangsspeicher wieder frei.

Definiert in Zeile 34 der Datei SpsClient.cpp.

Benutzt NULL.

5.33.3 Dokumentation der Elementfunktionen

5.33.3.1 bool SpsClient::requestAxisPos (void) [virtual]

Fragt die Spielstangenpositionen beim Server an.

Rückgabe

true wenn die Anfrage gesendet wurde

Implementiert SpsClientInterface.

Definiert in Zeile 130 der Datei SpsClient.cpp.

Benutzt ubf16_request::bitfeld_16::bothAxisPos_02, ubf16_request::data, requestPacket::id, REQUEST, requestPacket::request und ubf16_request::value.

5.33.3.2 bool SpsClient::requestScore (void) [virtual]

Fragt die Spielstangenpositionen beim Server an.

Rückgabe

true wenn die Anfrage gesendet wurde

Implementiert SpsClientInterface.

Definiert in Zeile 175 der Datei SpsClient.cpp.

Benutzt ubf16_request::data, requestPacket::id, REQUEST, requestPacket::request, ubf16_request::bitfeld_16::score_06 und ubf16_request::value.

5.33.3.3 bool SpsClient::sendAxisPos (float *axisPos[2][4]*, float *axisAngle[2][4]*) [virtual]

Sendet die Spielstangenpositionen an den Server.

Parameter

in	*axisPos*	Array mit der Translation, Position und Geschwindigkeit
in	*axisAngle*	Array mit der Rotation, Winkel und Geschwindigkeit

Rückgabe

true wenn die Spielstangenpositionen gesendet wurde

Implementiert SpsClientInterface.

Definiert in Zeile 150 der Datei SpsClient.cpp.

Benutzt ANGLE, AXIS_FOUR, AXIS_ONE, float_2_2_4::flt, float_2_2_4::id, POSITION, SET_AXISPOS, VALUE und VELOCITY.

5.33.3.4 bool SpsClient::sendScore (unsigned short *computer*, unsigned short *human*) [virtual]

Fragt die Spielstangenpositionen beim Server an.

Parameter

in	*computer*	Spielstand des Computers
in	*human*	Spielstand des menschlichen Spielers

Rückgabe

true wenn die Anfrage gesendet wurde

Implementiert SpsClientInterface.

Definiert in Zeile 220 der Datei SpsClient.cpp.

Benutzt COMPUTER, HUMAN, unsignedShort_2::id, SET_SCORE und unsignedShort_2::us.

5.33.3.5 bool SpsClient::start (SpsClientUserInterface * *userInterface*) [virtual]

Startet den Client.

Parameter

in	*userInterface*	Zeiger auf den Benutzer des Clients

Rückgabe

true wenn alles beim Starten des Clients geklappt hat

Fügt den Client als Abonnent zum Socket hinzu und startet den Socket.

Implementiert SpsClientInterface.

Definiert in Zeile 45 der Datei SpsClient.cpp.

5.33.3.6 bool SpsClient::subscribeScore (bool *subs* = true) [virtual]

Fragt die Spielstangenpositionen beim Server an.

Parameter

in	*subs*	muss auf true gesetzt Werden um den Spielstand zu abonnieren, auf false um das Abonnement zu kündigen

Rückgabe

true wenn die Anfrage gesendet wurde

Implementiert SpsClientInterface.

Definiert in Zeile 195 der Datei SpsClient.cpp.

Benutzt bool_1::bl, bool_1::id und SUBSCRIBE_SCORE.

5.33.3.7 void SpsClient::update (Publish * *pub*) [virtual]

update-Funktion der Observer Klasse

Parameter

in	*pub*	Publisher Objekt, welches über ein Ereignis benachichtigt (hier UDPsocket)

Wird von UDPsocket aufgerufen wenn Daten eingetroffen sind.

Implementiert Observer.

Definiert in Zeile 62 der Datei SpsClient.cpp.

Benutzt ANGLE, COMPUTER, float_2_2_4::flt, GET_BOTH_AXISPOS, GET_SCORE, HUMAN, POSITION und unsignedShort_2::us.

5.33.3.8 void SpsClient::update (Publish * *pub*, int * *iValues*, unsigned int *iValuesSize*) [virtual]

update-Funktion der Observer Klasse

Parameter

in	*pub*	Publisher Objekt, welches über ein Ereignis benachichtigt (hier UDPsocket)
in	*iValues*	wird ignoriert
in	*iValuesSize*	wird ignoriert

Wird von UDPsocket aufgerufen wenn Daten eingetroffen sind.

Implementiert Observer.

Definiert in Zeile 114 der Datei SpsClient.cpp.

5.33.3.9 void SpsClient::update (Publish * *pub*, float * *fValues*, unsigned int *fValuesSize*, int * *iValues* = NULL, unsigned int *iValuesSize* = 0) [virtual]

update-Funktion der Observer Klasse

Parameter

in	*pub*	Publisher Objekt, welches über ein Ereignis benachichtigt (hier UDPsocket)
in	*fValues*	wird ignoriert
in	*fValuesSize*	wird ignoriert
in	*iValues*	wird ignoriert
in	*iValuesSize*	wird ignoriert

Wird von UDPsocket aufgerufen wenn Daten eingetroffen sind.

Implementiert Observer.

Definiert in Zeile 119 der Datei SpsClient.cpp.

5.33.4 Dokumentation der Datenelemente

5.33.4.1 char* SpsClient::myRecvData [private]

Speicher für die empfangenen Daten.

Definiert in Zeile 150 der Datei SpsClient.h.

5.33.4.2 sockaddr_in SpsClient::myServerAddr [private]

Adresse des Servers.

Definiert in Zeile 145 der Datei SpsClient.h.

5.33.4.3 UDPsocket* SpsClient::mySocket [private]

der UDP-Socket

Definiert in Zeile 135 der Datei SpsClient.h.

5.33.4.4 SpsClientUserInterface* SpsClient::myUser [private]

Zeiger auf den Benutzer dieser Klasse.

Definiert in Zeile 140 der Datei SpsClient.h.

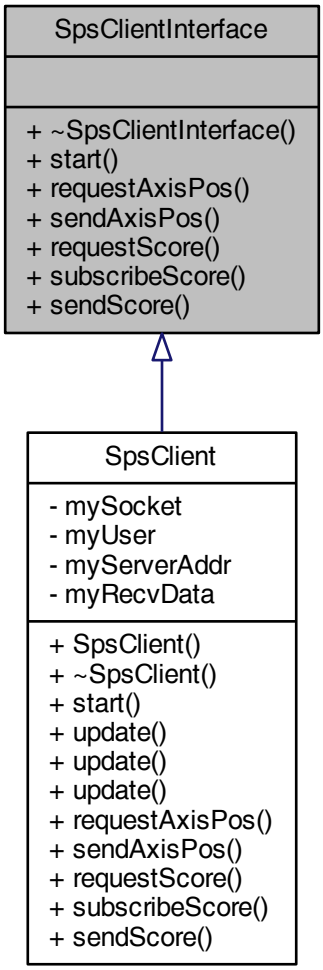
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- SpsClient.h
- SpsClient.cpp

5.34 SpsClientInterface Klassenreferenz

```
#include <SpsClientInterface.h>
```

Klassendiagramm für SpsClientInterface:



Öffentliche Methoden

- virtual ~SpsClientInterface (void)

  Virtueller Destruktor.
- virtual bool start (SpsClientUserInterface *userInterface)=0

  Startet den Client.
- virtual bool requestAxisPos ()=0

  Fragt die Spielstangenpositionen beim Server an.
- virtual bool sendAxisPos (float axisPos[2][4], float axisAngle[2][4])=0

  Sendet die Spielstangenpositionen an den Server.
- virtual bool requestScore ()=0

  Fragt die Spielstangenpositionen beim Server an.
- virtual bool subscribeScore (bool subs=true)=0

  Fragt die Spielstangenpositionen beim Server an.
- virtual bool sendScore (unsigned short computer, unsigned short human)=0

Fragt die Spielstangenpositionen beim Server an.

5.34.1 Ausführliche Beschreibung

Definiert in Zeile 18 der Datei SpsClientInterface.h.

5.34.2 Beschreibung der Konstruktoren und Destruktoren

5.34.2.1 virtual SpsClientInterface::~SpsClientInterface (void) [inline, virtual]

Virtueller Destruktor.

Definiert in Zeile 24 der Datei SpsClientInterface.h.

5.34.3 Dokumentation der Elementfunktionen

5.34.3.1 virtual bool SpsClientInterface::requestAxisPos ( ) [pure virtual]

Fragt die Spielstangenpositionen beim Server an.

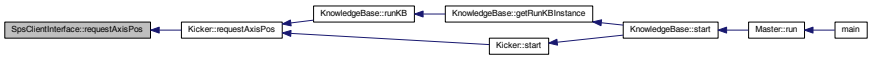
Rückgabe

true wenn die Anfrage gesendet wurde

Implementiert in SpsClient.

Wird benutzt von Kicker::requestAxisPos().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.34.3.2 virtual bool SpsClientInterface::requestScore ( ) [pure virtual]

Fragt die Spielstangenpositionen beim Server an.

Rückgabe

true wenn die Anfrage gesendet wurde

Implementiert in SpsClient.

Wird benutzt von Kicker::requestScore().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.34.3.3 virtual bool SpsClientInterface::sendAxisPos (float *axisPos[2][4]*, float *axisAngle[2][4]*) [pure virtual]

Sendet die Spielstangenpositionen an den Server.

Parameter

in	*axisPos*	Array mit der Translation
in	*axisAngle*	Array mit der Rotation

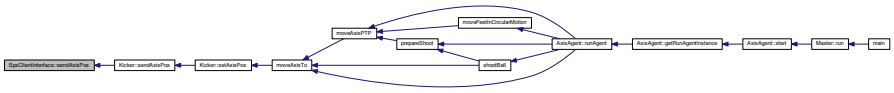
Rückgabe

true wenn die Spielstangenpositionen gesendet wurde

Implementiert in SpsClient.

Wird benutzt von Kicker::sendAxisPos().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.34.3.4 virtual bool SpsClientInterface::sendScore (unsigned short *computer*, unsigned short *human*) [pure virtual]

Fragt die Spielstangenpositionen beim Server an.

Parameter

in	*computer*	Spielstand des Computers
in	*human*	Spielstand des menschlichen Spielers

Rückgabe

true wenn die Anfrage gesendet wurde

Implementiert in SpsClient.

Wird benutzt von Kicker::sendScore().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.34.3.5 virtual bool SpsClientInterface::start (SpsClientUserInterface * *userInterface*) [pure virtual]

Startet den Client.

Parameter

in	*userInterface*	Zeiger auf den Benutzer des Clients

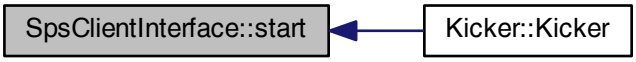
Rückgabe

true wenn alles beim Starten des Clients geklappt hat

Implementiert in SpsClient.

Wird benutzt von Kicker::Kicker().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.34.3.6 virtual bool SpsClientInterface::subscribeScore (bool *subs* = true) [pure virtual]

Fragt die Spielstangenpositionen beim Server an.

Parameter

in	*subs*	muss auf true gesetzt Werden um den Spielstand zu abonnieren, auf false um das Abonnement zu kündigen

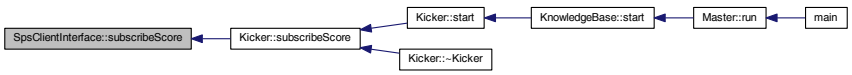
Rückgabe

true wenn die Anfrage gesendet wurde

Implementiert in SpsClient.

Wird benutzt von Kicker::subscribeScore().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

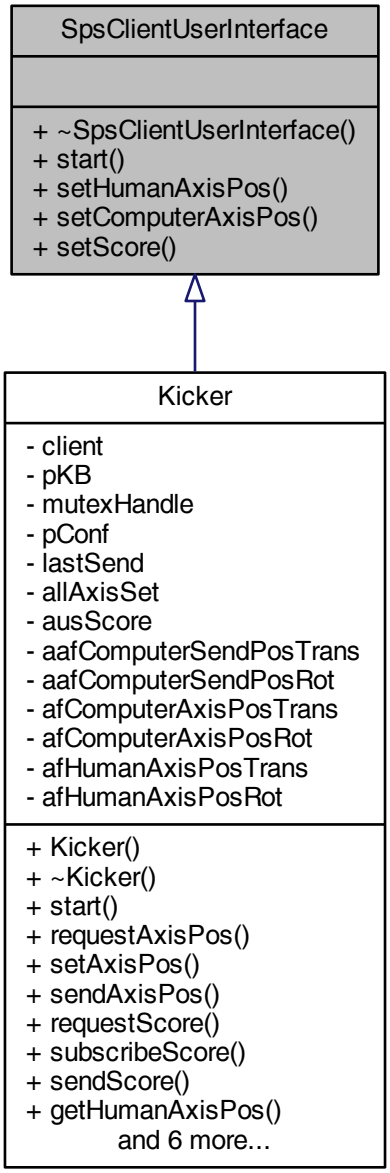
- SpsClientInterface.h

5.35 SpsClientUserInterface Schnittstellenreferenz

Interface zur Kommunikation mit dem Benutzer des UDP-Clients.

`#include <SpsClientUserInterface.h>`

Klassendiagramm für SpsClientUserInterface:



Öffentliche Methoden

- virtual ~SpsClientUserInterface (void)

  Virtueller Destruktor.
- virtual void start ()=0

Startet den die Benutzung des Clients.

- virtual void setHumanAxisPos (float axisPos[4], float axisAngle[4])=0

  setzt die Spielstangenposition des Menschen

- virtual void setComputerAxisPos (float axisPos[4], float axisAngle[4])=0

  setzt die Spielstangenposition des Computers

- virtual void setScore (unsigned short computer, unsigned short human)=0

  setzt den Spielstand

5.35.1 Ausführliche Beschreibung

Interface zur Kommunikation mit dem Benutzer des UDP-Clients.

Definiert in Zeile 16 der Datei SpsClientUserInterface.h.

5.35.2 Beschreibung der Konstruktoren und Destruktoren

5.35.2.1 virtual SpsClientUserInterface::~SpsClientUserInterface (void) [inline, virtual]

Virtueller Destruktor.

Definiert in Zeile 22 der Datei SpsClientUserInterface.h.

5.35.3 Dokumentation der Elementfunktionen

5.35.3.1 virtual void SpsClientUserInterface::setComputerAxisPos (float *axisPos[4]*, float *axisAngle[4]*) [pure virtual]

setzt die Spielstangenposition des Computers

Parameter

in	*axisPos*	Translation die vom Client empfangen wurde
in	*axisAngle*	Rotation die vom Client empfangen wurde

Wird vom Client aufgerufen wenn eine Spielstangenposition vom Server eingetroffen ist.

Implementiert in Kicker.

5.35.3.2 virtual void SpsClientUserInterface::setHumanAxisPos (float *axisPos[4]*, float *axisAngle[4]*) [pure virtual]

setzt die Spielstangenposition des Menschen

Parameter

in	*axisPos*	Translation die vom Client empfangen wurde
in	*axisAngle*	Rotation die vom Client empfangen wurde

Wird vom Client aufgerufen wenn eine Spielstangenposition vom Server eingetroffen ist.

Implementiert in Kicker.

5.35.3.3 virtual void SpsClientUserInterface::setScore (unsigned short *computer*, unsigned short *human*) [pure virtual]

setzt den Spielstand

Parameter

in	*computer*	Spielstand des Computers
in	*human*	Spielstand des Menschen

Wird vom Client aufgerufen wenn ein Spielstand vom Server eingetroffen ist.

Implementiert in Kicker.

5.35.3.4 virtual void SpsClientUserInterface::start ( ) [pure virtual]

Startet den die Benutzung des Clients.

Rückgabe

true wenn alles beim Starten des Clients geklappt hat

Implementiert in Kicker.

Die Dokumentation für diese Schnittstelle wurde erzeugt aufgrund der Datei:

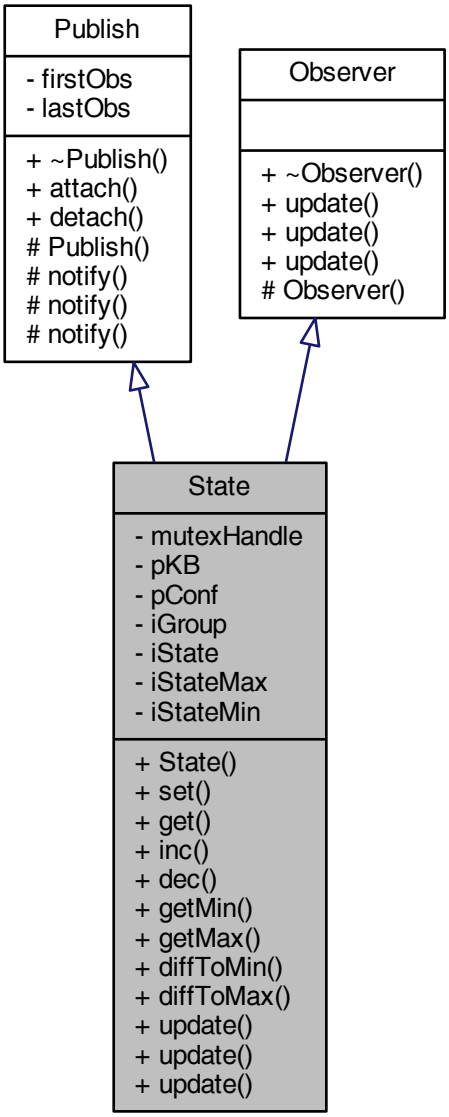
- SpsClientUserInterface.h

5.36 State Klassenreferenz

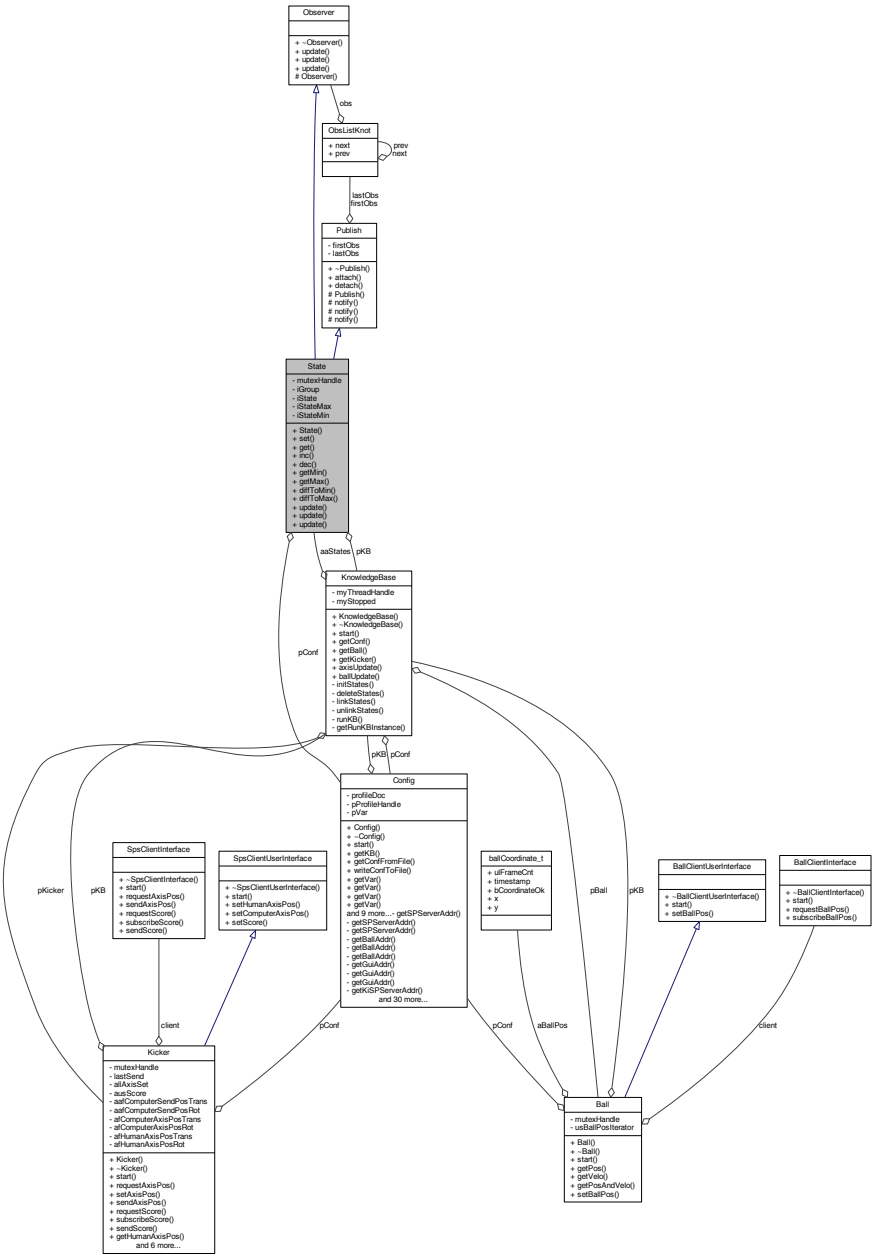
Status einer Eigenschaft.

`#include <States.h>`

Klassendiagramm für State:



Zusammengehörigkeiten von State:



Öffentliche Methoden

- State (unsigned int group, unsigned int state, KnowledgeBase *kB)

  Konstruktor.
- void set (int state=0)

  setzt den Status
- int get (void)

  liefert den Status
- void inc (void)

  Inkrementiert den Status.
- void dec (void)

Dekrementiert den Status.

- int getMin (void)

  liefert den minimalen Wert des Status
- int getMax (void)

  liefert den maximalen Wert des Status
- int diffToMin (void)

  liefert die Differenz zum minimalen Wert
- int diffToMax (void)

  liefert die Differenz zum maximalen Wert
- void update (Publish *pub)

  update-Funktion der Basisklasse Observer
- void update (Publish *pub, int *iValues, unsigned int iValuesSize)

  update-Funktion der Basisklasse Observer
- void update (Publish *pub, float *fValues, unsigned int fValuesSize, int *iValues=NULL, unsigned int iValues-Size=0)

  update-Funktion der Basisklasse Observer

Private Attribute

- HANDLE mutexHandle

  Mutex-Handle.
- KnowledgeBase * pKB

  Zeiger auf die Wissensbasis.
- Config * pConf

  Zeiger auf die Konfiguration.
- unsigned int iGroup

  Gruppe des Status.
- unsigned int iState

  Status in der Gruppe.
- int iStateMax

  maximaler und minimaler Wert des Status
- int iStateMin

5.36.1 Ausführliche Beschreibung

Status einer Eigenschaft.

Jede Instanz dieser Klasse stellt einen Status dar.

Definiert in Zeile 26 der Datei States.h.

5.36.2 Beschreibung der Konstruktoren und Destruktoren

5.36.2.1 State::State (unsigned int *group*, unsigned int *state*, KnowledgeBase * *kB*) [inline]

Konstruktor.

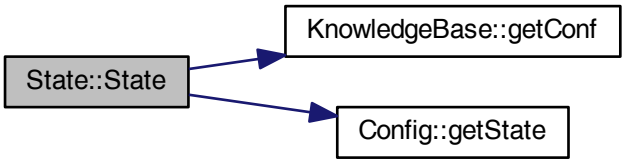
Parameter

in	*group*	Gruppe der der Status angehört
in	*state*	Welcher Status in der Gruppe es ist
in	*kB*	Zeiger auf die Wissensbasis

Definiert in Zeile 145 der Datei States.h.

Benutzt KnowledgeBase::getConf(), Config::getState(), iGroup, iState, iStateMax, iStateMin, mutexHandle, NULL, pConf, pKB, STATE_MAX und STATE_MIN.

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.36.3 Dokumentation der Elementfunktionen

5.36.3.1 void State::dec (void) [inline]

Dekrementiert den Status.

Definiert in Zeile 60 der Datei States.h.

5.36.3.2 int State::diffToMax (void) [inline]

liefert die Differenz zum maximalen Wert

return Differenz zwischen aktuellem Wert und Maximum

Definiert in Zeile 84 der Datei States.h.

Benutzt iStateMax.

5.36.3.3 int State::diffToMin (void) [inline]

liefert die Differenz zum minimalen Wert

return Differenz zwischen aktuellem Wert und Minimum

Definiert in Zeile 78 der Datei States.h.

Benutzt iStateMin.

5.36.3.4 int State::get (void) [inline]

liefert den Status

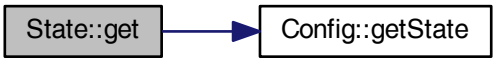
Rückgabe

Wert des Status

Definiert in Zeile 198 der Datei States.h.

Benutzt ERROR_MUTEX_WAIT, Config::getState(), iGroup, iState, mutexHandle und pConf.

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.36.3.5 int State::getMax (void) [inline]

liefert den maximalen Wert des Status

Rückgabe

maximaler Wert des Status

Definiert in Zeile 72 der Datei States.h.

Benutzt iStateMax.

5.36.3.6 int State::getMin (void) [inline]

liefert den minimalen Wert des Status

Rückgabe

minimaler Wert des Status

Definiert in Zeile 66 der Datei States.h.

Benutzt iStateMin.

5.36.3.7 void State::inc (void) [inline]

Inkrementiert den Status.

Definiert in Zeile 55 der Datei States.h.

5.36.3.8 void State::set (int *state* = 0) [inline]

setzt den Status

Parameter

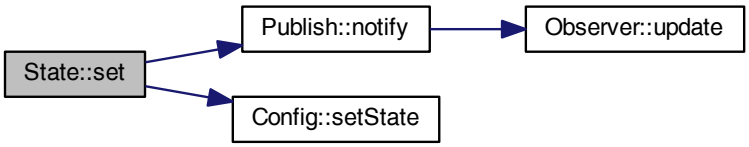
in	*state*	neuer Status

Setzt den Status und überprüft ihn dabei auf Maximum/Minimum. Wenn sich der Status verändert hat, werden alle Abonnenten benachichtigt.

Definiert in Zeile 169 der Datei States.h.

Benutzt ERROR_MUTEX_WAIT, iGroup, iState, iStateMax, iStateMin, mutexHandle, Publish::notify(), pConf und Config::setState().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



5.36.3.9 void State::update (Publish * *pub*) [inline, virtual]

update-Funktion der Basisklasse Observer

Implementiert Observer.

Definiert in Zeile 89 der Datei States.h.

Wird benutzt von update().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.36.3.10 void State::update (Publish * *pub*, int * *iValues*, unsigned int *iValuesSize*) [inline, virtual]

update-Funktion der Basisklasse Observer

Parameter

in	*pub*	Zeiger auf den Publisher
in	*iValues*	Array mit int-Werten (Gruppe, Status und Wert) vom Publisher
in	*iValuesSize*	Anzahl der Werte in iValues (sollte 3 sein)

Implementiert Observer.

Definiert in Zeile 214 der Datei States.h.

5.36.3.11 void State::update (Publish * *pub*, float * *fValues*, unsigned int *fValuesSize*, int * *iValues* = NULL, unsigned int *iValuesSize* = 0) [inline, virtual]

update-Funktion der Basisklasse Observer

Parameter

in	*pub*	Zeiger auf den Publisher
in	*fValues*	Array mit float-Werten (wird ignoriert)
in	*fValuesSize*	Anzahl der Werte in fValues (wird ignoriert)
in	*iValues*	Array mit int-Werten (Gruppe, Status und Wert) vom Publisher
in	*iValuesSize*	Anzahl der Werte in iValues (sollte 3 sein)

Implementiert Observer.

Definiert in Zeile 107 der Datei States.h.

Benutzt update().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

5.36.4 Dokumentation der Datenelemente

5.36.4.1 unsigned int State::iGroup [private]

Gruppe des Status.

Definiert in Zeile 132 der Datei States.h.

Wird benutzt von get(), set() und State().

5.36.4.2 unsigned int State::iState [private]

Status in der Gruppe.

Definiert in Zeile 137 der Datei States.h.

Wird benutzt von get(), set() und State().

5.36.4.3 int State::iStateMax [private]

maximaler und minimaler Wert des Status

Definiert in Zeile 142 der Datei States.h.

Wird benutzt von diffToMax(), getMax(), set() und State().

5.36.4.4 int State::iStateMin [private]

Definiert in Zeile 142 der Datei States.h.

Wird benutzt von diffToMin(), getMin(), set() und State().

5.36.4.5 HANDLE State::mutexHandle [private]

Mutex-Handle.

Definiert in Zeile 117 der Datei States.h.

Wird benutzt von get(), set() und State().

5.36.4.6 Config* State::pConf [private]

Zeiger auf die Konfiguration.

Definiert in Zeile 127 der Datei States.h.

Wird benutzt von get(), set() und State().

5.36.4.7 KnowledgeBase* State::pKB [private]

Zeiger auf die Wissensbasis.

Definiert in Zeile 122 der Datei States.h.

Wird benutzt von State().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

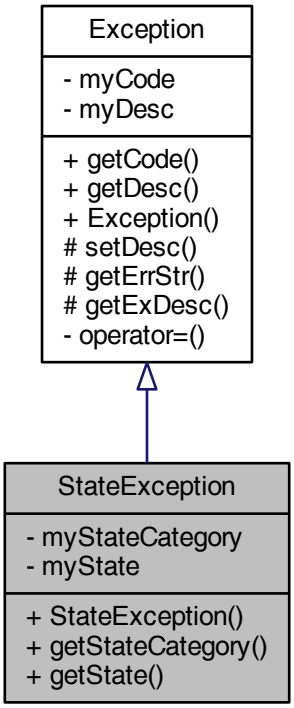
- States.h

5.37 StateException Klassenreferenz

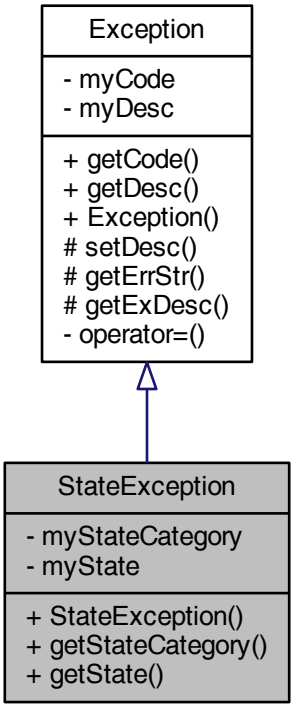
Exception-Klasse für Zustände.

```
#include <Exception.h>
```

Klassendiagramm für StateException:



Zusammengehörigkeiten von StateException:



Öffentliche Methoden

- StateException (const char ∗category, unsigned int state)

  Konstruktor.
- const char ∗ getStateCategory () const

  gibt den Namen der Zustandskategorie zurück
- unsigned int getState () const

  gibt die ID des Zustandes zurück

Private Attribute

- const char ∗ myStateCategory

  Zustandskategorie.
- unsigned int myState

  ID des Zustandes.

5.37.1 Ausführliche Beschreibung

Exception-Klasse für Zustände.

Definiert in Zeile 310 der Datei Exception.h.

5.37.2 Beschreibung der Konstruktoren und Destruktoren

5.37.2.1 StateException::StateException (const char * *category*, unsigned int *state*) [inline]

Konstruktor.

Parameter

in	*category*	Name der Zustandskategorie
in	*state*	ID des Zustandes

Setzt den Exception-Code, den Namen der Zustandskategorie und die ID des Zustandes.

Definiert in Zeile 320 der Datei Exception.h.

5.37.3 Dokumentation der Elementfunktionen

5.37.3.1 unsigned int StateException::getState ( ) const [inline]

gibt die ID des Zustandes zurück

Rückgabe

ID des Zustandes

Definiert in Zeile 335 der Datei Exception.h.

Benutzt myState.

5.37.3.2 const char* StateException::getStateCategory ( ) const [inline]

gibt den Namen der Zustandskategorie zurück

Rückgabe

Zustandskategorie

Definiert in Zeile 329 der Datei Exception.h.

Benutzt myStateCategory.

5.37.4 Dokumentation der Datenelemente

5.37.4.1 unsigned int StateException::myState [private]

ID des Zustandes.

Definiert in Zeile 346 der Datei Exception.h.

Wird benutzt von getState().

5.37.4.2 const char* StateException::myStateCategory [private]

Zustandskategorie.

Definiert in Zeile 341 der Datei Exception.h.

Wird benutzt von getStateCategory().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

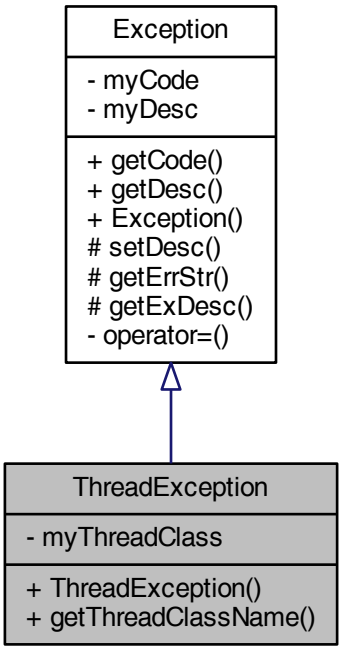
- Exception.h

5.38 ThreadException Klassenreferenz

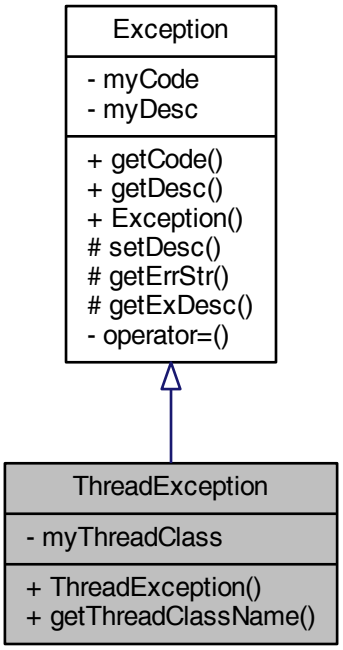
Exception-Klasse für Threads.

```
#include <Exception.h>
```

Klassendiagramm für ThreadException:



Zusammengehörigkeiten von ThreadException:



Öffentliche Methoden

- ThreadException (const char *threadClass)

  Konstruktor.
- const char * getThreadClassName () const

  gibt den Klassennamen zurück

Private Attribute

- const char * myThreadClass

  Klassenname.

5.38.1 Ausführliche Beschreibung

Exception-Klasse für Threads.

Definiert in Zeile 279 der Datei Exception.h.

5.38.2 Beschreibung der Konstruktoren und Destruktoren

5.38.2.1 ThreadException::ThreadException (const char * *threadClass*) [inline]

Konstruktor.

Parameter

in	*threadClass*	Name der Klasse in der das Problem aufgetreten ist

Setzt den Exception-Code und den Klassennamen.

Definiert in Zeile 288 der Datei Exception.h.

5.38.3 Dokumentation der Elementfunktionen

5.38.3.1 const char* ThreadException::getThreadClassName ( ) const [inline]

gibt den Klassennamen zurück

Rückgabe

Klassennamen

Definiert in Zeile 297 der Datei Exception.h.

Benutzt myThreadClass.

5.38.4 Dokumentation der Datenelemente

5.38.4.1 const char* ThreadException::myThreadClass [private]

Klassenname.

Definiert in Zeile 303 der Datei Exception.h.

Wird benutzt von getThreadClassName().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Exception.h

5.39 tObsListKnot Strukturreferenz

Struktur für ein Listenelement.

`#include <publish.h>`

5.39.1 Ausführliche Beschreibung

Struktur für ein Listenelement.

Struktur für ein Listenelement einer doppelt verketteten Liste, welche alle Observer Elemente enthält, die sich angemeldet haben.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

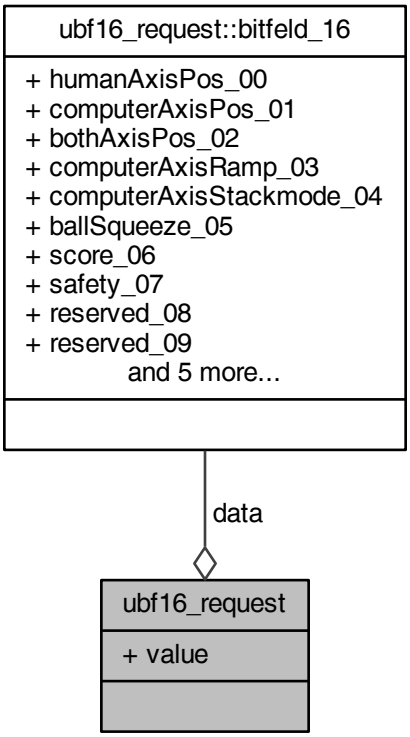
- publish.h

5.40 ubf16_request Variantenreferenz

Bitfeld, bei dem jedes Bit bestimmte Daten anfordert.

```
#include <UdpDefines.h>
```

Zusammengehörigkeiten von ubf16_request:



Klassen

- struct bitfeld_16

Öffentliche Attribute

- unsigned int value: 16
- struct ubf16_request::bitfeld_16 data

5.40.1 Ausführliche Beschreibung

Bitfeld, bei dem jedes Bit bestimmte Daten anfordert.

Definiert in Zeile 99 der Datei UdpDefines.h.

5.40.2 Dokumentation der Datenelemente

5.40.2.1 struct ubf16_request::bitfeld_16 ubf16_request::data

Wird benutzt von SpsClient::requestAxisPos() und SpsClient::requestScore().

5.40.2.2 unsigned int ubf16_request::value

Definiert in Zeile 101 der Datei UdpDefines.h.

Wird benutzt von SpsClient::requestAxisPos() und SpsClient::requestScore().

Die Dokumentation für diese Variante wurde erzeugt aufgrund der Datei:

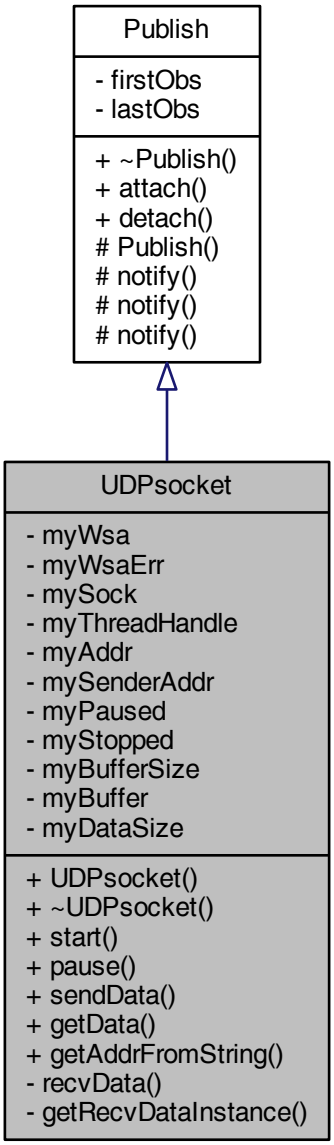
- UdpDefines.h

5.41 UDPsocket Klassenreferenz

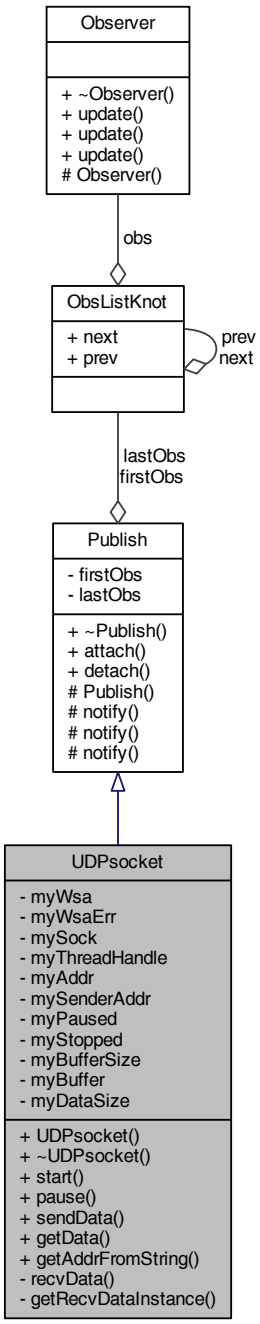
Stellt ein UDP-Socket dar.

```
#include <UdpSocket.h>
```

Klassendiagramm für UDPsocket:



Zusammengehörigkeiten von UDPsocket:



Öffentliche Methoden

- UDPsocket (unsigned int bufferSize, unsigned short localPort)

  Konstruktor.
- ~UDPsocket ()

  Destruktor.
- bool start ()

Startet die Socketfunktion.

- void pause (bool pause=true)

  Pausiert den Empfang von Daten.
- int sendData (const char ∗data, int len, const sockaddr_in host)

  Sendet Daten an einen anderen Socket.
- int getData (char ∗&data, sockaddr_in &host)

  Liefert die empfangenen Daten.
- bool getAddrFromString (const char ∗host, sockaddr_in ∗addr)

  Ermittelt die IP-Adresse aus einer Zeichenkette.

Private Methoden

- DWORD recvData ()

  Eigener Thread, welcher Pakete empfängt.

Private, statische Methoden

- static DWORD WINAPI getRecvDataInstance (void ∗instance)

  Gibt den Funktionspointer auf die Funktion recvData() zurück.

Private Attribute

- WSADATA myWsa

  Struktur, welche Informationen über Winsock enthält.
- int myWsaErr

  Wird beim Initialisieren von Winsock gesetzt.
- SOCKET mySock

  Socket Deskriptor.
- HANDLE myThreadHandle

  Handle vom Thread.
- sockaddr_in myAddr

  eigene Adresse
- sockaddr_in mySenderAddr

  Sender-Adresse.
- bool myPaused

  Wird auf true gesetzt wenn der Daten-Empfang pausiert werden soll.
- bool myStopped

  Wird auf true gesetzt wenn der Daten-Empfang entgültig beendet wird.
- unsigned int myBufferSize

  Größe des Empfangsspeichers.
- char ∗ myBuffer

  Empfanfsspeicher.
- int myDataSize

  Anzahl der empfangenen Bytes.

5.41.1 Ausführliche Beschreibung

Stellt ein UDP-Socket dar.

Diese Klasse stellt ein UDP-Socket dar, sie kann Pakete an einen anderen Host senden und von diesem empfangen. Ob die Klasse als Server oder Client verwendet wird, hängt davon ab wie sie aufgerufen wird.

Definiert in Zeile 28 der Datei UdpSocket.h.

5.41.2 Beschreibung der Konstruktoren und Destruktoren

5.41.2.1 UDPsocket::UDPsocket (unsigned int *bufferSize,* unsigned short *localPort*)

Konstruktor.

Parameter

in	*bufferSize*	Größe des Empfangsspeichers (maximal zu erwartende Datenmenge)
in	*localPort*	Port des eigenen Sockets

Bereitet Winsock vor und wandelt die lokale Host-Adresse um.

Definiert in Zeile 16 der Datei UdpSocket.cpp.

Benutzt myAddr, myPaused, myStopped, myWsa und myWsaErr.

5.41.2.2 UDPsocket::~UDPsocket ( )

Destruktor.

Beendet den Daten-Empfang, räumt Winsock auf und gibt den Empfangsspeicher wieder frei.

Definiert in Zeile 30 der Datei UdpSocket.cpp.

Benutzt myBuffer, myPaused, mySock, myStopped, myThreadHandle und NULL.

5.41.3 Dokumentation der Elementfunktionen

5.41.3.1 bool UDPsocket::getAddrFromString (const char * *host,* sockaddr_in * *addr*)

Ermittelt die IP-Adresse aus einer Zeichenkette.

Parameter

in	*host*	Zeichenkette mit der IP-Adresse oder dem Rechnernamen
out	*addr*	Struktur in der die ermittelte Adresse abgelegt werden soll

Rückgabe

true wenn die Adresse ermittelt werden konnte, ansonsten false

Definiert in Zeile 199 der Datei UdpSocket.cpp.

Benutzt NULL.

Wird benutzt von BallClient::BallClient().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.41.3.2 int UDPsocket::getData (char *& *data*, sockaddr_in & *host*)

Liefert die empfangenen Daten.

Parameter

out	*data*	Referenz auf den Zeiger auf den Speicherbereich, in dem die Daten liegen
out	*host*	Referenz auf die Variable, in die die Adresse des Senders gespeichert wird

Rückgabe

Anzahl der empfangenen Bytes

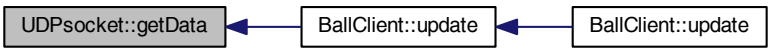
Wenn die empfangen Daten ausserhalb von der update()-Funktion des Observers verwendet werden sollen, müssen die Daten kopiert werden da der Speicherbereich beim erneuten eintreffen von Daten überschrieben wird.

Definiert in Zeile 142 der Datei UdpSocket.cpp.

Benutzt myBuffer, myDataSize und mySenderAddr.

Wird benutzt von BallClient::update().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.41.3.3 DWORD WINAPI UDPsocket::getRecvDataInstance (void * *instance*) [static, private]

Gibt den Funktionspointer auf die Funktion recvData() zurück.

Parameter

in	*instance*	Pointer auf die Instanz dieser Klasse

Rückgabe

Funktionspointer auf die Funktion recvData() der aktuellen Instanz

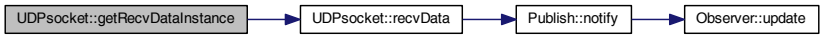
Trick um einen neuen Thread mit einer Memberfunktion zu erzeugen.

Definiert in Zeile 190 der Datei UdpSocket.cpp.

Benutzt NULL und recvData().

Wird benutzt von start().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.41.3.4 void UDPsocket::pause (bool *pause* = `true`)

Pausiert den Empfang von Daten.

Parameter

in	*pause*	bei true wird der Empfang von Daten pausiert, bei false wird der Empfang fortgesetzt

Wenn pausiert werden weiterhin Pakete empfangen, allerdings wird der Observer darüber nur nicht mehr informiert.

Definiert in Zeile 126 der Datei UdpSocket.cpp.

Benutzt myPaused.

5.41.3.5 DWORD UDPsocket::recvData ( ) `[private]`

Eigener Thread, welcher Pakete empfängt.

Eigener Thread, welcher Pakete empfängt und alle angemeldeten Observer Objekte über das Eintreffen neuer Pakete informiert.

Siehe auch

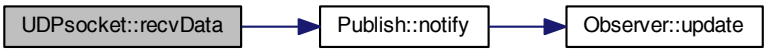
Publish::notify()

Definiert in Zeile 150 der Datei UdpSocket.cpp.

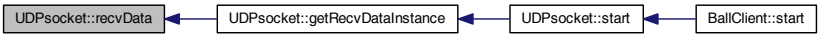
Benutzt myBuffer, myBufferSize, myDataSize, myPaused, mySenderAddr, mySock, myStopped und Publish::notify().

Wird benutzt von getRecvDataInstance().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.41.3.6 int UDPsocket::sendData (const char * *data*, int *len*, const sockaddr_in *host*)

Sendet Daten an einen anderen Socket.

Parameter

in	*data*	Pointer auf die zu sendenden Daten
in	*len*	Anzahl der zu sendenden Bytes
in	*host*	Adresse des entfernten Rechners

Rückgabe

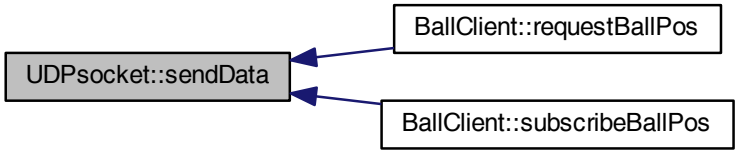
Anzahl der tatsächlich gesendeten Bytes

Definiert in Zeile 132 der Datei UdpSocket.cpp.

Benutzt mySock.

Wird benutzt von BallClient::requestBallPos() und BallClient::subscribeBallPos().

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



5.41.3.7 bool UDPsocket::start (void)

Startet die Socketfunktion.

Rückgabe

true bei Erfolg, ansonsten false

Überprüft ob Winsock richtig vorbereitet wurde, legt den Socket an und bindet diesen, legt den Empfangsspeicher an und startet den Thread, der die Daten empfängt.

Definiert in Zeile 76 der Datei UdpSocket.cpp.

Benutzt getRecvDataInstance(), myAddr, myBuffer, myBufferSize, myDataSize, myPaused, mySock, myStopped, myThreadHandle, myWsa, myWsaErr und NULL.

Wird benutzt von BallClient::start().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

5.41.4 Dokumentation der Datenelemente

5.41.4.1 sockaddr_in UDPsocket::myAddr [private]

eigene Adresse

Siehe auch

UDPsocket::UDPsocket()

Definiert in Zeile 135 der Datei UdpSocket.h.

Wird benutzt von start() und UDPsocket().

5.41.4.2 char* UDPsocket::myBuffer [private]

Empfanfsspeicher.

Siehe auch

UDPsocket::recvData()
UDPsocket::getData()

Empfanfsspeicher in dem die empfangenen Daten abgelegt werden.

Definiert in Zeile 170 der Datei UdpSocket.h.

Wird benutzt von getData(), recvData(), start() und ~UDPsocket().

5.41.4.3 unsigned int UDPsocket::myBufferSize [private]

Größe des Empfangsspeichers.

Siehe auch

UDPsocket::UDPsocket()

Definiert in Zeile 161 der Datei UdpSocket.h.

Wird benutzt von recvData() und start().

5.41.4.4 int UDPsocket::myDataSize [private]

Anzahl der empfangenen Bytes.

Siehe auch

UDPsocket::recvData()
UDPsocket::getData()
UDPsocket::myBuffer

Anzahl der Bytes die empfangen wurden und im Empfangsspeicher abgelegt sind.

Definiert in Zeile 180 der Datei UdpSocket.h.

Wird benutzt von getData(), recvData() und start().

5.41.4.5 bool UDPsocket::myPaused [private]

Wird auf true gesetzt wenn der Daten-Empfang pausiert werden soll.

Siehe auch

UDPclient::pause()
UDPclient::recvData()

Definiert in Zeile 148 der Datei UdpSocket.h.

Wird benutzt von pause(), recvData(), start(), UDPsocket() und ~UDPsocket().

5.41.4.6 sockaddr_in UDPsocket::mySenderAddr [private]

Sender-Adresse.

Siehe auch

UDPsocket::recvData()

Definiert in Zeile 141 der Datei UdpSocket.h.

Wird benutzt von getData() und recvData().

5.41.4.7 SOCKET UDPsocket::mySock [private]

Socket Deskriptor.

Wird in UDPsocket::start() beim Anlegen des Sockets gesetzt.

Definiert in Zeile 120 der Datei UdpSocket.h.

Wird benutzt von recvData(), sendData(), start() und ~UDPsocket().

5.41.4.8 bool UDPsocket::myStopped [private]

Wird auf true gesetzt wenn der Daten-Empfang entgültig beendet wird.

Siehe auch

UDPclient::~UDPclient()
UDPclient::recvData()

Definiert in Zeile 155 der Datei UdpSocket.h.

Wird benutzt von recvData(), start(), UDPsocket() und ~UDPsocket().

5.41.4.9 HANDLE UDPsocket::myThreadHandle [private]

Handle vom Thread.

Handle vom Thread welcher zum Empfangen von Paketen zuständig ist. Wird in UDPsocket::start() beim Erzeugen des Threads gesetzt. Wird benutzt um im Destruktor auf das Ende des Threads zu warten.

Definiert in Zeile 129 der Datei UdpSocket.h.

Wird benutzt von start() und ~UDPsocket().

5.41.4.10 WSADATA UDPsocket::myWsa [private]

Struktur, welche Informationen über Winsock enthält.

Wird in UDPsocket::UDPsocket() gesetzt und in UDPsocket::start() überprüft

Definiert in Zeile 104 der Datei UdpSocket.h.

Wird benutzt von start() und UDPsocket().

5.41.4.11 int UDPsocket::myWsaErr [private]

Wird beim Initialisieren von Winsock gesetzt.

Wird beim erfolgreichen Initialisieren von Winsock in UDPsocket::UDPsocket() auf Null (0) gesetzt. Enthält im Fehlerfall einen Fehlercode.

Definiert in Zeile 113 der Datei UdpSocket.h.

Wird benutzt von start() und UDPsocket().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- UdpSocket.h
- UdpSocket.cpp

5.42 unsignedShort_2 Strukturreferenz

Anfrage-Paket für bestimmte Daten.

```
#include <UdpDefines.h>
```

Öffentliche Attribute

- unsigned int id: 16
- unsigned short us [2]

5.42.1 Ausführliche Beschreibung

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein Array aus zwei unsigned short-Werten.

Definiert in Zeile 161 der Datei UdpDefines.h.

5.42.2 Dokumentation der Datenelemente

5.42.2.1 unsigned int unsignedShort_2::id

Definiert in Zeile 163 der Datei UdpDefines.h.

Wird benutzt von SpsClient::sendScore().

5.42.2.2 unsigned short unsignedShort_2::us[2]

Definiert in Zeile 164 der Datei UdpDefines.h.

Wird benutzt von SpsClient::sendScore() und SpsClient::update().

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- UdpDefines.h

Kapitel 6

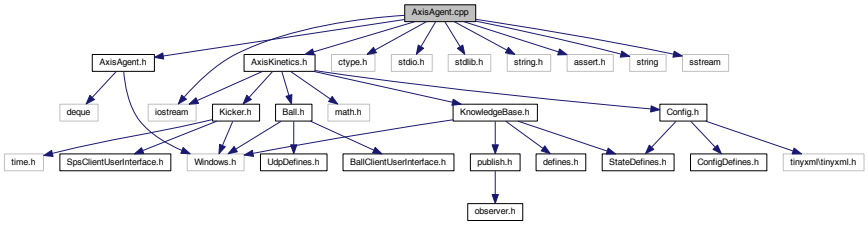
Datei-Dokumentation

6.1 AxisAgent.cpp-Dateireferenz

Enthält die Implementierung der Klasse AxisAgent.

```
#include <iostream>
#include "AxisAgent.h"
#include "AxisKinetics.h"
```

Include-Abhängigkeitsdiagramm für AxisAgent.cpp:



6.1.1 Ausführliche Beschreibung

Enthält die Implementierung der Klasse AxisAgent.

Autor

Scharel Clemens

Datum

04.04.2012

Version

0.1 Entwicklungsphase

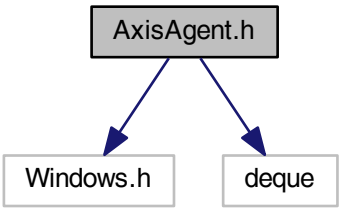
Definiert in Datei AxisAgent.cpp.

6.2 AxisAgent.h-Dateireferenz

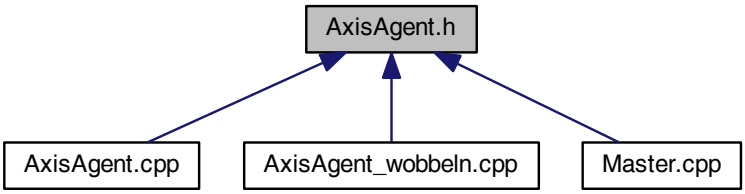
Enthält die Definition der Klasse AxisAgent.

```
#include <Windows.h>
#include <deque>
```

Include-Abhängigkeitsdiagramm für AxisAgent.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- struct funcElement

  Struktur für ein Listenelement.
- class FuncList

  Eine einfach verkettete Liste für Funktionspointer.
- class AxisAgent

  Ein Agent zur Ansteuerung einer Spielstange.

Typdefinitionen

- typedef bool(∗ moveFunc)(unsigned int axis, float values[], KnowledgeBase ∗pKB, bool blocking, int timeout)
- typedef struct funcElement funcElement

6.2.1 Ausführliche Beschreibung

Enthält die Definition der Klasse AxisAgent.

Autor

Scharel Clemens

Datum

04.04.2012

Version

0.1 Entwicklungsphase

Definiert in Datei AxisAgent.h.

6.2.2 Dokumentation der benutzerdefinierten Typen

6.2.2.1 typedef struct funcElement funcElement

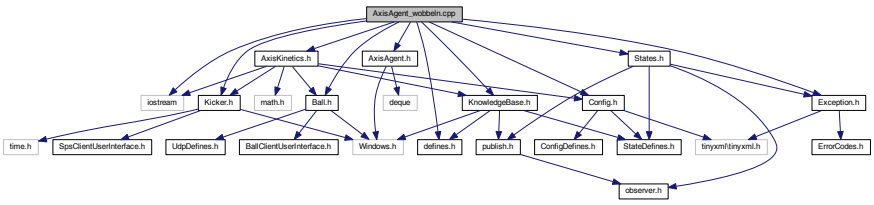
6.2.2.2 typedef bool(∗ moveFunc)(unsigned int axis, float values[], KnowledgeBase ∗pKB, bool blocking, int timeout)

Definiert in Zeile 20 der Datei AxisAgent.h.

6.3 AxisAgent_wobbeln.cpp-Dateireferenz

```
#include <iostream>
#include "AxisAgent.h"
#include "AxisKinetics.h"
#include "KnowledgeBase.h"
#include "States.h"
#include "Config.h"
#include "Kicker.h"
#include "Ball.h"
#include "Exception.h"
#include "defines.h"
```

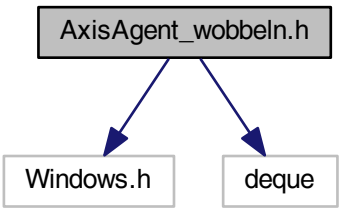
Include-Abhängigkeitsdiagramm für AxisAgent_wobbeln.cpp:



6.4 AxisAgent_wobbeln.h-Dateireferenz

```
#include <Windows.h>
#include <deque>
```

Include-Abhängigkeitsdiagramm für AxisAgent_wobbeln.h:



Klassen

- struct funcElement

  Struktur für ein Listenelement.
- class FuncList

  Eine einfach verkettete Liste für Funktionspointer.
- class AxisAgent

  Ein Agent zur Ansteuerung einer Spielstange.

Typdefinitionen

- typedef bool(∗ moveFunc)(unsigned int axis, float values[], KnowledgeBase ∗pKB, bool blocking, int timeout)
- typedef struct funcElement funcElement

6.4.1 Dokumentation der benutzerdefinierten Typen

6.4.1.1 typedef struct funcElement funcElement

6.4.1.2 typedef bool(∗ moveFunc)(unsigned int axis, float values[], KnowledgeBase ∗pKB, bool blocking, int timeout)

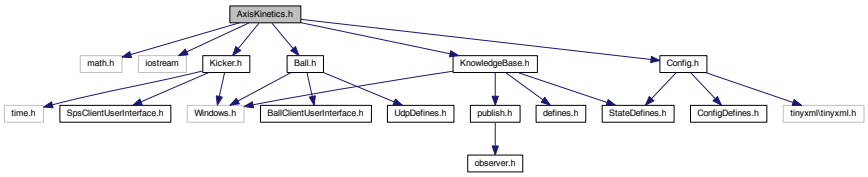
Definiert in Zeile 20 der Datei AxisAgent_wobbeln.h.

6.5 AxisKinetics.h-Dateireferenz

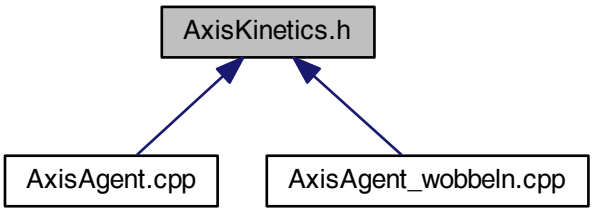
Enthält Funktionen zur Bewegungssteuerung der Spielstangen.

```
#include <math.h>
#include <iostream>
#include "KnowledgeBase.h"
#include "Config.h"
#include "Kicker.h"
#include "Ball.h"
```

Include-Abhängigkeitsdiagramm für AxisKinetics.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Funktionen

- bool moveAxisTo (unsigned int axis, float values[4], KnowledgeBase ∗pKB, bool blocking=true, int timeout=1000)

  Bewegt die Spielstange.
- bool moveAxisPTP (unsigned int axis, float values[3], KnowledgeBase ∗pKB, bool blocking=true, int timeout=1000)

  Bewegt die Spielstange.
- bool moveFeetInCircularMotion (unsigned int axis, float values[3], KnowledgeBase ∗pKB, bool blocking=true, int timeout=2000)

  Führt eine Kreisbewegung durch.
- bool prepareShoot (unsigned int axis, float values[2], KnowledgeBase ∗pKB, bool blocking=true, int timeout=1000)

  Bereiten einen Schuss vor.
- bool shootBall (unsigned int axis, float values[4], KnowledgeBase ∗pKB, bool blocking=true, int timeout=1000)

  Schießt den Ball.
- bool checkMaxTransVelocity (float &velocity, KnowledgeBase ∗pKB)

  Überprüft die Geschwindigkeit.
- bool checkMaxRotVelocity (float &velocity, KnowledgeBase ∗pKB)

  Überprüft die Geschwindigkeit.
- int chooseFigure (unsigned int axis, float &posY, KnowledgeBase ∗pKB)

  Wählt die Spielfigur die posY am besten erreicht.
- float calcPosWhereBallCrossesAxis (unsigned int axis, KnowledgeBase ∗pKB)

  Berechnet die Position an der der Ball eine Spielstange kreuzt.
- int checkPosReached (unsigned int axis, float posY, float tolerance, KnowledgeBase ∗pKB)

Überprüft ob eine Position von einer Spielfigur erreicht wurde.

- int checkBallShootable (KnowledgeBase *pKB)

  Überprüft ob der Ball von einer Spielstange geschossen werden kann.

- bool findAngleFromBallToGoal (unsigned int axis, float &angle, KnowledgeBase *pKB)

  Sucht den direkten Weg vom Ball ins Tor.

6.5.1 Ausführliche Beschreibung

Enthält Funktionen zur Bewegungssteuerung der Spielstangen.

Autor

Scharel Clemens

Datum

08.05.2012

Version

0.1 Entwicklungsphase

Definiert in Datei AxisKinetics.h.

6.5.2 Dokumentation der Funktionen

6.5.2.1 float calcPosWhereBallCrossesAxis (unsigned int *axis*, KnowledgeBase * *pKB*)

Berechnet die Position an der der Ball eine Spielstange kreuzt.

Parameter

in	*axis*	Spielstange die gekreuzt werden soll
in	*pKB*	Zeiger auf die Wissensbasis

Rückgabe

die Position an der die Spielstange vom Ball gekreuzt wird. Wenn der Ball die Spielstange nicht kreuzt (z.B. wenn er in die andere Richtung rollt), wird -1 zurückgegeben.

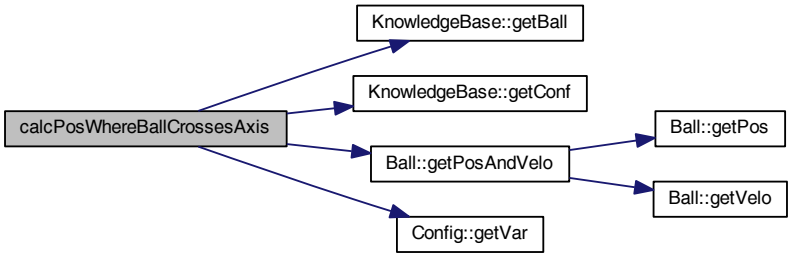
Die Position wird durch den aktuellen Geschwindigkeitsvektor des Balles berechnet. Die Banden werden berücksichtigt (Abprallwinkel = Aufprallwinkel).

Definiert in Zeile 642 der Datei AxisKinetics.h.

Benutzt AxisDistance, COORDINATE_Y, FieldSize, KnowledgeBase::getBall(), KnowledgeBase::getConf(), Ball::getPosAndVelo() und Config::getVar().

Wird benutzt von AxisAgent::runAgent().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

6.5.2.2 int checkBallShootable (KnowledgeBase * *pKB*)

Überprüft ob der Ball von einer Spielstange geschossen werden kann.

Parameter

in	*pKB*	Zeiger auf die Wissensbasis

Rückgabe

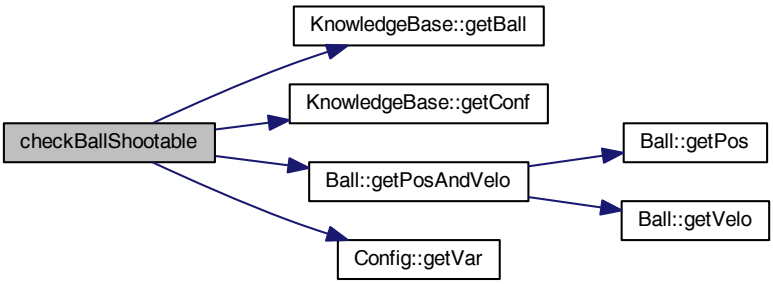
Die Spielstange die den Ball schiessen kann, kann keine Spielstange den Ball schießen, wird -1 zurückgegeben.

Definiert in Zeile 695 der Datei AxisKinetics.h.

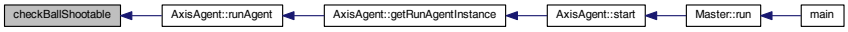
Benutzt AXIS_FOUR, AXIS_ONE, AxisPosX, AxisShift, BallToAxisReachable, BallToAxisShootable, FIGURE_ONE, FigurePos, KnowledgeBase::getBall(), KnowledgeBase::getConf(), Ball::getPosAndVelo() und Config::getVar().

Wird benutzt von AxisAgent::runAgent().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



6.5.2.3 bool checkMaxRotVelocity (float & *velocity*, KnowledgeBase * *pKB*)

Überprüft die Geschwindigkeit.

Parameter

in,out	*velocity*	Geschwindigkeit die überprüft werden soll
in	*pKB*	Zeiger auf die Wissensbasis

Rückgabe

true, wenn die Geschwindigkeit überprüft werden konnte

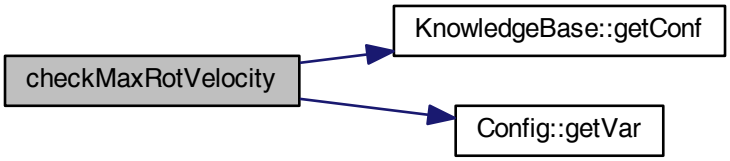
Wenn die Rotations-Geschwindigkeit größer als die maximale Geschwindigkeit ist, oder kleiner als 1 °/s ist, wird der entsprechende Minimal- Maximalwert gesetzt.

Definiert in Zeile 543 der Datei AxisKinetics.h.

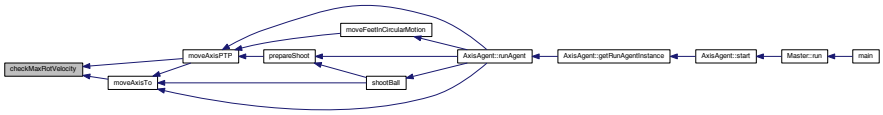
Benutzt KnowledgeBase::getConf(), Config::getVar() und VelocityMaxRot.

Wird benutzt von moveAxisPTP() und moveAxisTo().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



6.5.2.4 bool checkMaxTransVelocity (float & *velocity*, KnowledgeBase * *pKB*)

Überprüft die Geschwindigkeit.

Parameter

in,out	*velocity*	Geschwindigkeit die überprüft werden soll#
in	*pKB*	Zeiger auf die Wissensbasis

Rückgabe

true, wenn die Geschwindigkeit überprüft werden konnte

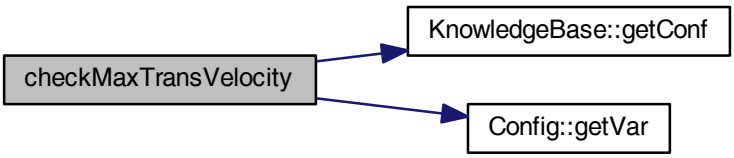
Wenn die Translations-Geschwindigkeit größer als die maximale Geschwindigkeit ist, oder kleiner als 1mm/s ist, wird der entsprechende Minimal- Maximalwert gesetzt.

Definiert in Zeile 527 der Datei AxisKinetics.h.

Benutzt KnowledgeBase::getConf(), Config::getVar() und VelocityMaxTrans.

Wird benutzt von moveAxisPTP() und moveAxisTo().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

6.5.2.5 int checkPosReached (unsigned int *axis*, float *posY*, float *tolerance*, KnowledgeBase * *pKB*)

Überprüft ob eine Position von einer Spielfigur erreicht wurde.

Parameter

in	*axis*	Spielstange die die Position erreichen soll
in	*posY*	die Position, die erreicht werden soll
in	*tolerance*	Erlaubte Abweichung vom Zielwert
in	*pKB*	Zeiger auf die Wissensbasis

Rückgabe

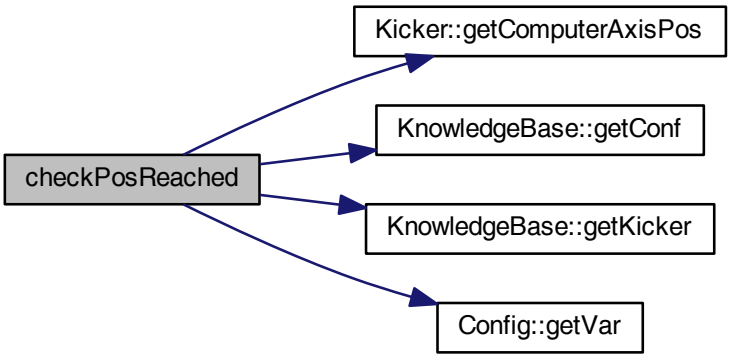
Die Spielfigur, die die Position erreicht hat. Wurde die Position noch nicht erreicht, wird -1 zurückgegeben.

Es wird überprüft ob eine der Spielfiguren an der Spielstange die angegebene Position auf dem Spielfeld hat.

Definiert in Zeile 668 der Datei AxisKinetics.h.

Benutzt FIGURE_ONE, FigurePos, Kicker::getComputerAxisPos(), KnowledgeBase::getConf(), KnowledgeBase::getKicker() und Config::getVar().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



6.5.2.6 int chooseFigure (unsigned int *axis*, float & *posY*, KnowledgeBase * *pKB*)

Wählt die Spielfigur die posY am besten erreicht.

Parameter

in	*axis*	Spielstange, welche der Agent ansteuern soll
in,out	*posY*	Position die erreicht werden soll, nachdem die Funktion erfolgreich ausgeführt wurde, beinhaltet die Variable die Position die die Spielstange annehmen muss um die Position mit der ausgewählten Spielfigur zu erreichen.
in	*pKB*	Zeiger auf die Wissensbasis

Rückgabe

die Spielfigur, die gewählt wurde. Wenn keine Spielfigur die Position erreichen kann, wird diejenige zurückgegeben, die am ehesten dran kommt. Im Fehlerfall wird -1 zurückgegeben.

Noch zu erledigen Nähe der Spielfigur zur Zielposition im Gütefaktor überprüfen!

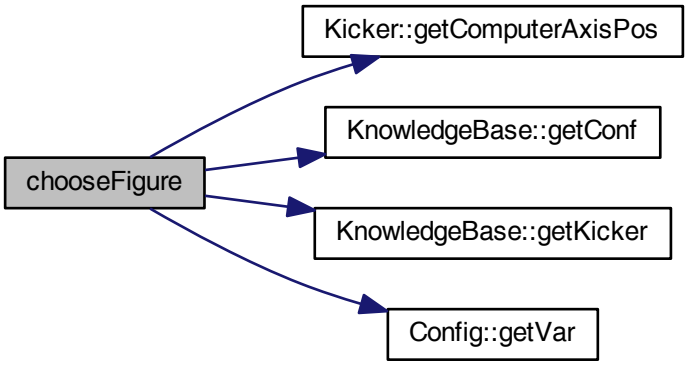
Die Spielfigur wird nach bestimmten Kriterien ausgesucht, z.B. welche schneller an der Position ist und welche dort den größeren Bewegungsfreiraum hat.

Definiert in Zeile 559 der Datei AxisKinetics.h.

Benutzt AxisOffset, AxisShift, COORDINATE_Y, FieldSize, FIGURE_FIVE, FIGURE_ONE, FigurePos, Kicker::getComputerAxisPos(), KnowledgeBase::getConf(), KnowledgeBase::getKicker() und Config::getVar().

Wird benutzt von prepareShoot() und AxisAgent::runAgent().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

6.5.2.7 bool findAngleFromBallToGoal (unsigned int *axis*, float & *angle*, KnowledgeBase * *pKB*)

Sucht den direkten Weg vom Ball ins Tor.

Parameter

in	*axis*	Spielstange die den Ball schiessen soll
out	*angle*	Der Winkel in dem geschossen werden muss, damit das Tor getroffen wird
in	*pKB*	Zeiger auf die Wissensbasis

Rückgabe

true wenn ein Weg ins Tor gefunden wurde, wenn der Weg zu Tor blockiert ist wird false zurückgegeben. Trotzdem wird angle mit dem am besten geeigneten Winkel gesetzt.

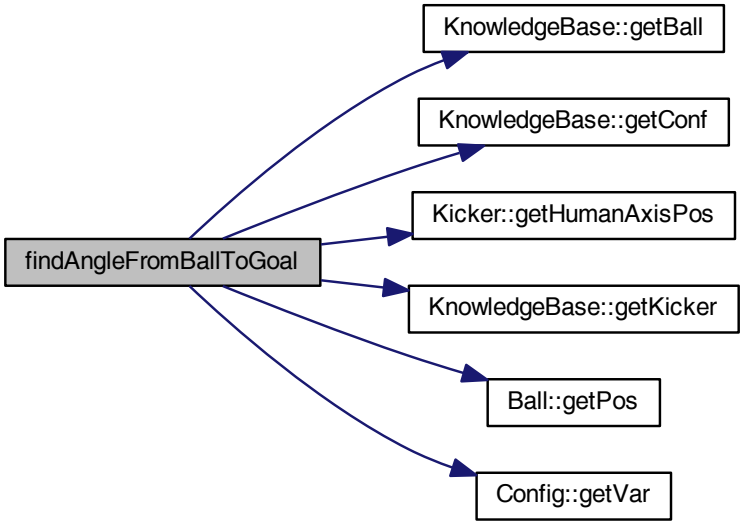
Noch zu erledigen Bei der Schnittpunktberechnung auf Werte ausserhalb des Spielfeldes überprüfen!

Definiert in Zeile 737 der Datei AxisKinetics.h.

Benutzt AXIS_FOUR, AXIS_ONE, AxisPosX, COORDINATE_X, COORDINATE_Y, FieldSize, FIGURE_FIVE, FIGURE_ONE, FigurePos, KnowledgeBase::getBall(), KnowledgeBase::getConf(), Kicker::getHumanAxisPos(), KnowledgeBase::getKicker(), Ball::getPos(), Config::getVar(), GoalSize und PI.

Wird benutzt von AxisAgent::runAgent().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

6.5.2.8 bool **moveAxisPTP** (unsigned int *axis*, float *values[3]*, **KnowledgeBase** * *pKB*, bool *blocking* = `true`, int *timeout* = `1000`)

Bewegt die Spielstange.

Parameter

in	*axis*	Spielstange, welche der Agent ansteuern soll
in	*values*	Zielposition, Zielwinkel und Zeitraum
in	*pKB*	Zeiger auf die Wissensbasis, zur Kommunikation
in	*blocking*	bei true wird die Funktion ausgeführt bis die Zielposition erreicht wurde, ansonsten kehrt die Funktion gleich nach Senden des Befehls zurück
in	*timeout*	wenn blocking true ist, wird die Funktion nach timeout Millisekunden abgebrochen

Rückgabe

true wenn die Zielposition erreicht wurde, false wenn ein Fehler auftrat oder die Zielposition nach timeout Millisekunden nicht erreicht wurde

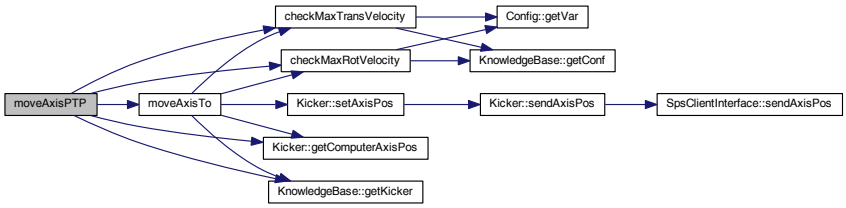
Bewegt die Spielstange per synchron Punkt zu Punkt Steuerung innerhalb des angegebenen Zeitraums an die angegebene Position.

Definiert in Zeile 239 der Datei AxisKinetics.h.

Benutzt checkMaxRotVelocity(), checkMaxTransVelocity(), Kicker::getComputerAxisPos(), KnowledgeBase::getKicker() und moveAxisTo().

Wird benutzt von moveFeetInCircularMotion(), prepareShoot() und AxisAgent::runAgent().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

6.5.2.9 bool moveAxisTo (unsigned int *axis*, float *values[4]*, KnowledgeBase * *pKB*, bool *blocking* = `true`, int *timeout* = `1000`)

Bewegt die Spielstange.

Parameter

in	*axis*	Spielstange, welche der Agent ansteuern soll
in	*values*	Zielposition und Geschwindigkeit, Zielwinkel und Geschwindigkeit
in	*pKB*	Zeiger auf die Wissensbasis, zur Kommunikation
in	*blocking*	bei true wird die Funktion ausgeführt bis die Zielposition erreicht wurde, ansonsten kehrt die Funktion gleich nach Senden des Befehls zurück
in	*timeout*	wenn blocking true ist, wird die Funktion nach timeout Millisekunden abgebrochen

Rückgabe

true wenn die Zielposition erreicht wurde, false wenn ein Fehler auftrat oder die Zielposition nach timeout Millisekunden nicht erreicht wurde

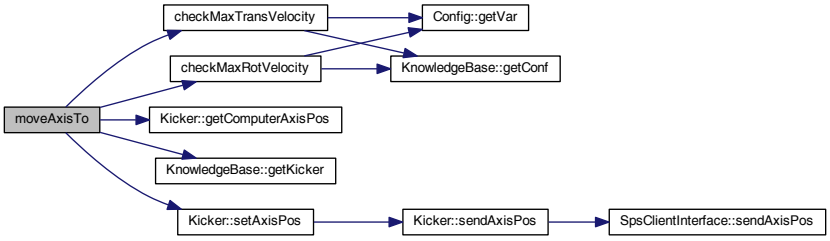
Bewegt die Spielstange mit einer vorgegebenen Geschwindigkeit an eine bestimmte Position.

Definiert in Zeile 203 der Datei AxisKinetics.h.

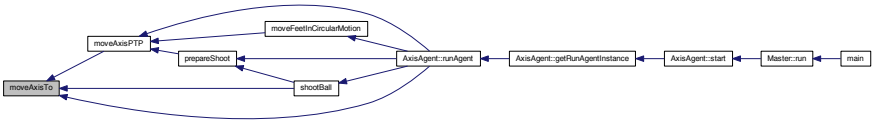
Benutzt checkMaxRotVelocity(), checkMaxTransVelocity(), Kicker::getComputerAxisPos(), KnowledgeBase::getKicker(), Kicker::setAxisPos() und VALUE.

Wird benutzt von moveAxisPTP(), AxisAgent::runAgent() und shootBall().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



6.5.2.10 bool moveFeetInCircularMotion (unsigned int *axis*, float *values[3]*, KnowledgeBase ∗ *pKB*, bool *blocking* = `true`, int *timeout* = `2000`)

Führt eine Kreisbewegung durch.

Parameter

in	*axis*	Spielstange, welche der Agent ansteuern soll
in	*values*	Radius des Kreises, Zeit pro Grad der Umdrehung und Gradanzahl der Umdrehungen
in	*pKB*	Zeiger auf die Wissensbasis, zur Kommunikation
in	*blocking*	wird ignoriert da diese Funktion immer blockierend ist
in	*timeout*	die Funktion wird spätestens nach timeout Millisekunden abgebrochen

Rückgabe

true wenn die Zielposition erreicht wurde, false wenn ein Fehler auftrat oder die Zielposition nach timeout Millisekunden nicht erreicht wurde

Führt eine Kreisbewegung mir den Füßen der Spielfiguren durch.

Ist der Radius positiv, so ist die aktuelle Postition der Translation der Spielstange die kleinste Position der Kreisbewegung.

Ist der Radius negativ, so ist die aktuelle Postition der Translation der Spielstange die größte Position der Kreisbewegung.

Der Mittelpunkt der Kreisbewegung befindet sich also um Radius neben der aktuellen Spielstangenposition.

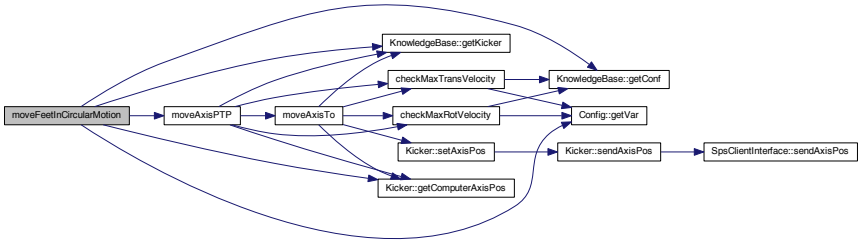
Sowohl Winkelgeschwindigkeit als auch Gradanzahl können negativ sein, das Produkt der beiden Vorzeichen ergibt die tatsächliche Drehrichtung Bis zu 2 Millisekunden pro Grad wird die Drehung richtig ausgeführt, bei Werten darunter kann die Spielstange den Bewegungsangaben nicht mehr folgen.

Definiert in Zeile 266 der Datei AxisKinetics.h.

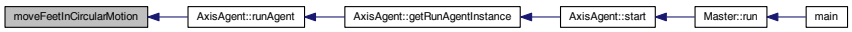
Benutzt FigureFootHeight, Kicker::getComputerAxisPos(), KnowledgeBase::getConf(), KnowledgeBase::getKicker(), Config::getVar(), moveAxisPTP() und PI.

Wird benutzt von AxisAgent::runAgent().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



6.5.2.11 bool prepareShoot (unsigned int *axis*, float *values[2]*, KnowledgeBase * *pKB*, bool *blocking* = `true`, int *timeout* = `1000`)

Bereiten einen Schuss vor.

Parameter

in	*axis*	Spielstange, welche der Agent ansteuern soll
in	*values*	Winkel der Flugbahn des Balles, Anfangswinkel
in	*pKB*	Zeiger auf die Wissensbasis, zur Kommunikation
in	*blocking*	bei true wird die Funktion ausgeführt bis die Zielposition erreicht wurde, ansonsten kehrt die Funktion gleich nach Senden des Befehls zurück
in	*timeout*	wenn blocking true ist, wird die Funktion nach timeout Millisekunden abgebrochen

Rückgabe

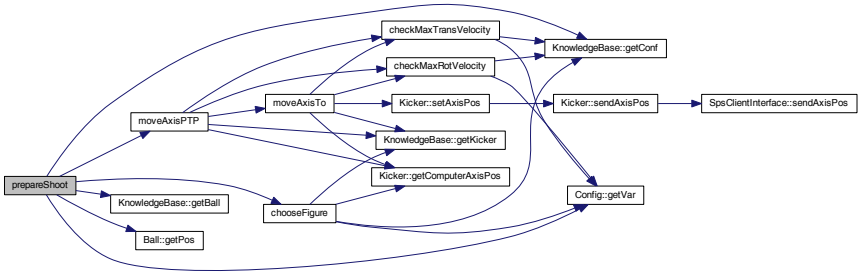
true wenn die Zielposition erreicht wurde, false wenn ein Fehler auftrat oder die Zielposition nach timeout Millisekunden nicht erreicht wurde

Definiert in Zeile 299 der Datei AxisKinetics.h.

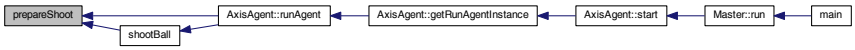
Benutzt AxisPosX, BallToAxisReachable, BallToAxisShootable, chooseFigure(), KnowledgeBase::getBall(), KnowledgeBase::getConf(), Ball::getPos(), Config::getVar(), moveAxisPTP(), NumShootAnglesByLin, NumShootAnglesByRot, SHOOT_ALL, SHOOT_ANGLE, SHOOT_OFFSET, ShootAnglesByLin, ShootAnglesByRot und VelocityMaxTrans.

Wird benutzt von AxisAgent::runAgent() und shootBall().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



6.5.2.12 bool shootBall (unsigned int *axis*, float *values[4]*, KnowledgeBase * *pKB*, bool *blocking* = `true`, int *timeout* = `1000`)

Schießt den Ball.

Parameter

in	*axis*	Spielstange, welche der Agent ansteuern soll
in	*values*	Winkel der Flugbahn des Balles, Schussgeschwindigkeit, Anfangswinkel, Endwinkel
in	*pKB*	Zeiger auf die Wissensbasis, zur Kommunikation
in	*blocking*	bei true wird die Funktion ausgeführt bis die Zielposition erreicht wurde, ansonsten kehrt die Funktion gleich nach Senden des Befehls zurück Achtung, wenn die Aktuelle Position der Spielfigur nicht der Startposition entspricht, die zum Ausführen des Schusses benötigt wird, blockierd die Funktion mindestens solange, bis die Startposition erreicht wurde!
in	*timeout*	wenn blocking true ist, wird die Funktion nach timeout Millisekunden abgebrochen

Rückgabe

true wenn die Zielposition erreicht wurde, false wenn ein Fehler auftrat oder die Zielposition nach timeout Millisekunden nicht erreicht wurde

Noch zu erledigen Andere Schussrichtungen wie nach vorne implementieren (seitlich und nach hinten).

Sießt den Ball so, dass er in einem bestimmten Winkel fliegt.

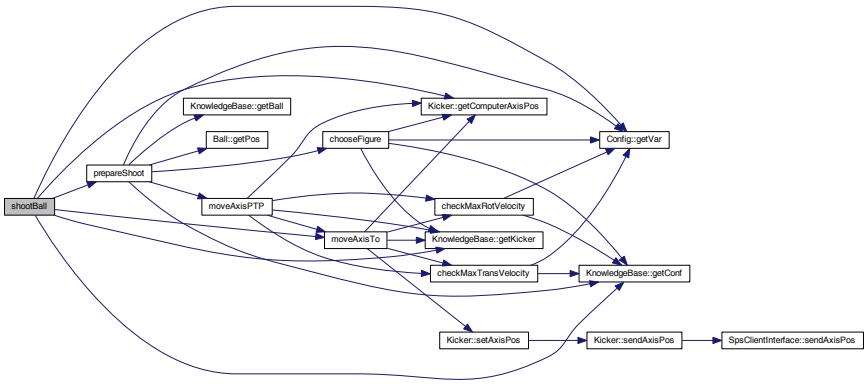
Damit die Funktion funktionniert, muss für den übergebenen Winkel ein Offset-Wert in der Konfigurationsdatei vorliegen. Liegt eine solche Konfiguration nicht vor, wird die Funktion nicht ausgeführt.\ Der Winkel der Flugbahn wird im mathematisch positiven Sinn, mit 0° auf der Längsachse angegeben.

Definiert in Zeile 407 der Datei AxisKinetics.h.

Benutzt Kicker::getComputerAxisPos(), KnowledgeBase::getConf(), KnowledgeBase::getKicker(), Config::getVar(), moveAxisTo(), prepareShoot() und VelocityMaxTrans.

Wird benutzt von AxisAgent::runAgent().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:



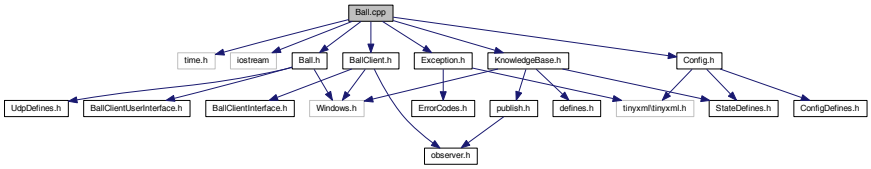
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

6.6 Ball.cpp-Dateireferenz

Enthält die Implementierung der Klasse Ball.

```
#include <time.h>
#include <iostream>
#include "Ball.h"
#include "BallClient.h"
#include "Exception.h"
#include "Config.h"
#include "KnowledgeBase.h"
```

Include-Abhängigkeitsdiagramm für Ball.cpp:



6.6.1 Ausführliche Beschreibung

Enthält die Implementierung der Klasse Ball.

Autor

Scharel Clemens

Datum

26.03.2012

Version

0.1 Entwicklungsphase

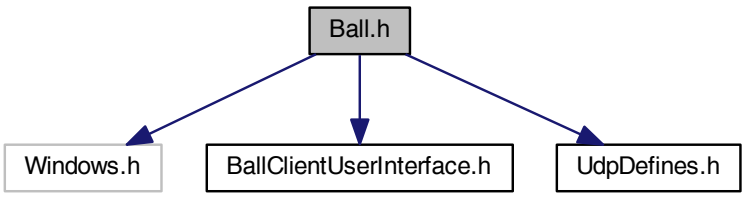
Definiert in Datei Ball.cpp.

6.7 Ball.h-Dateireferenz

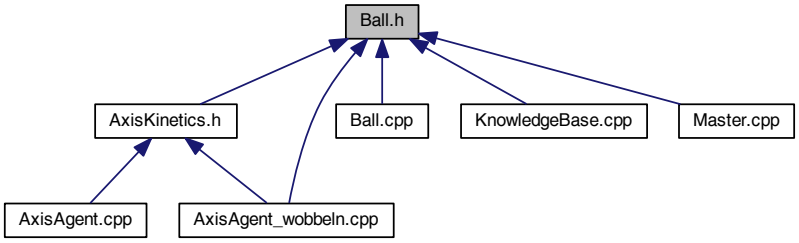
Enthält die Definition der Klasse Ball.

```
#include <Windows.h>
#include "BallClientUserInterface.h"
#include "UdpDefines.h"
```

Include-Abhängigkeitsdiagramm für Ball.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Ball

  Zugriff zur Ballerkennung.

6.7.1 Ausführliche Beschreibung

Enthält die Definition der Klasse Ball.

Autor

Scharel Clemens

Datum

26.03.2012

Version

0.1 Entwicklungsphase

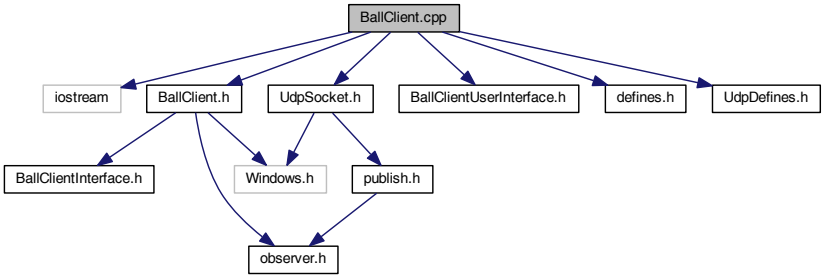
Definiert in Datei Ball.h.

6.8 BallClient.cpp-Dateireferenz

Enthält die Implementierung der Klasse BallClient.

```
#include <iostream>
#include "BallClient.h"
#include "UdpSocket.h"
#include "BallClientUserInterface.h"
#include "defines.h"
#include "UdpDefines.h"
```

Include-Abhängigkeitsdiagramm für BallClient.cpp:



6.8.1 Ausführliche Beschreibung

Enthält die Implementierung der Klasse BallClient.

Autor

Scharel Clemens

Datum

03.04.2012

Version

0.3 Testphase

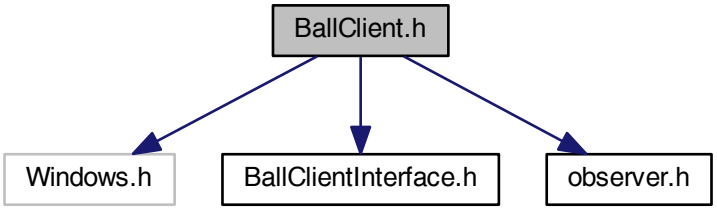
Definiert in Datei BallClient.cpp.

6.9 BallClient.h-Dateireferenz

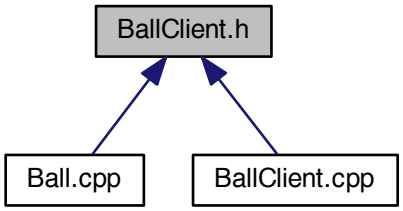
Enthält die Definition der Klasse BallClient.

```
#include <Windows.h>
#include "BallClientInterface.h"
#include "observer.h"
```

Include-Abhängigkeitsdiagramm für BallClient.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class BallClient

  ein UDP-BallClient

6.9.1 Ausführliche Beschreibung

Enthält die Definition der Klasse BallClient.

Autor

Scharel Clemens

Datum

03.04.2012

Version

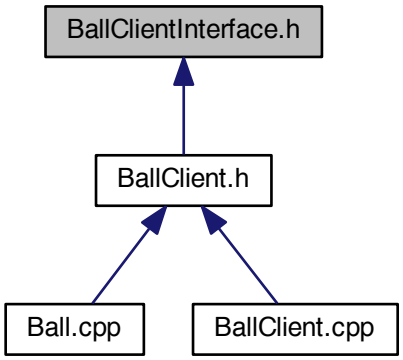
0.3 Testphase

Definiert in Datei BallClient.h.

6.10 BallClientInterface.h-Dateireferenz

Enthält das Interface zur Kommunikation mit dem UDP-Client.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class BallClientInterface

6.10.1 Ausführliche Beschreibung

Enthält das Interface zur Kommunikation mit dem UDP-Client.

Autor

Scharel Clemens

Datum

07.02.2012

Version

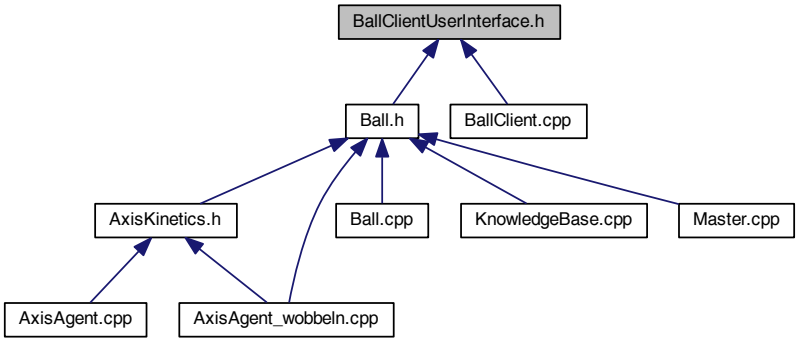
0.2 Testphase

Definiert in Datei BallClientInterface.h.

6.11 BallClientUserInterface.h-Dateireferenz

Enthält das Interface zur Kommunikation mit dem Benutzer des UDP-Clients.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- interface BallClientUserInterface

  Interface zur Kommunikation mit dem Benutzer des Ball-Clients.

6.11.1 Ausführliche Beschreibung

Enthält das Interface zur Kommunikation mit dem Benutzer des UDP-Clients.

Autor

Scharel Clemens

Datum

07.02.2012

Version

0.2 Testphase

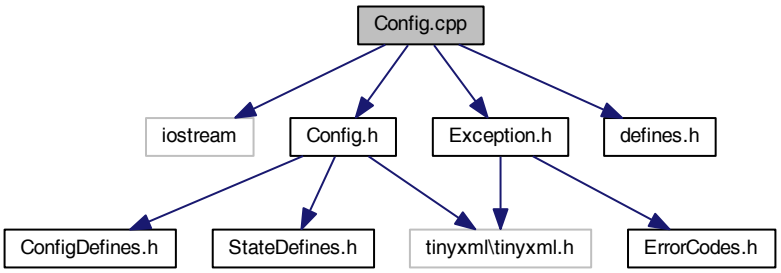
Definiert in Datei BallClientUserInterface.h.

6.12 Config.cpp-Dateireferenz

Enthält die Implementierung der Klasse Config.

```
#include <iostream>
#include "Config.h"
#include "Exception.h"
#include "defines.h"
```

Include-Abhängigkeitsdiagramm für Config.cpp:



6.12.1 Ausführliche Beschreibung

Enthält die Implementierung der Klasse Config.

Autor

Scharel Clemens

Datum

03.04.2012

Version

0.2 Entwicklungsphase

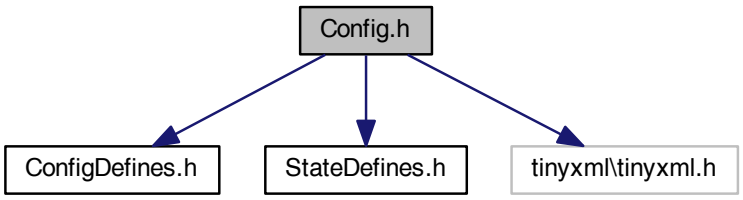
Definiert in Datei Config.cpp.

6.13 Config.h-Dateireferenz

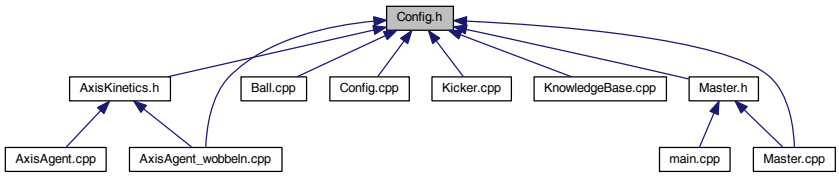
Enthält die Definition der Klasse Config.

```
#include "ConfigDefines.h"
#include "StateDefines.h"
#include "tinyxml\tinyxml.h"
```

Include-Abhängigkeitsdiagramm für Config.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Config

  Configurationsverwaltung.

6.13.1 Ausführliche Beschreibung

Enthält die Definition der Klasse Config.

Autor

Scharel Clemens

Datum

03.04.2012

Version

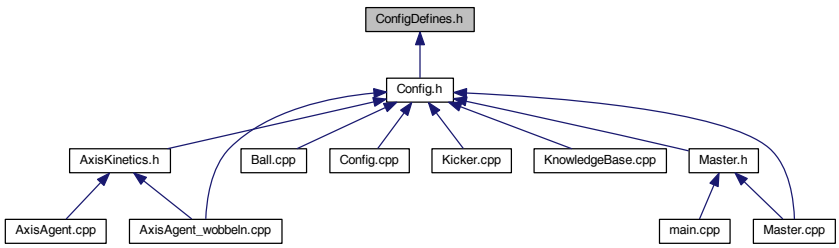
0.2 Entwicklungsphase

Definiert in Datei Config.h.

6.14 ConfigDefines.h-Dateireferenz

Enthält die Defines der Klasse Config.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Aufzählungen

- enum ConfigVar {
  SPServerPort, SPServerHost, BallPort, BallHost,
  GuiPort, GuiHost, KiSPServerPort, KiSPServerHost,
  KiBallPort, KiBallHost, KiGuiPort, KiGuiHost,
  AxisShift, AxisPosX, AxisOffset, AxisDistance,
  FigurePos, FieldSize, GoalSize, VelocityMaxTrans,
  VelocityMaxRot, FigureFootBreadth, FigureFootHeight, FigureFootThickness,
  FigureFieldDistance, BallDiameter, BallToAxisShootable, BallToAxisReachable,
  States, StatesPerGroup, ShootAnglesByRot, NumShootAnglesByRot,
  ShootAnglesByLin, NumShootAnglesByLin, All }

6.14.1 Ausführliche Beschreibung

Enthält die Defines der Klasse Config.

Autor

Scharel Clemens

Datum

23.03.2012

Version

0.1 Entwicklungsphase

Definiert in Datei ConfigDefines.h.

6.14.2 Dokumentation der Aufzählungstypen

6.14.2.1 enum ConfigVar

Aufzählungswerte:

SPServerPort

SPServerHost

BallPort

BallHost

GuiPort

GuiHost

KiSPServerPort

KiSPServerHost

KiBallPort

KiBallHost

KiGuiPort

KiGuiHost

AxisShift

AxisPosX

AxisOffset

AxisDistance

FigurePos

FieldSize

GoalSize

VelocityMaxTrans

VelocityMaxRot

FigureFootBreadth

FigureFootHeight

FigureFootThickness

FigureFieldDistance

BallDiameter

BallToAxisShootable

BallToAxisReachable

States

StatesPerGroup

ShootAnglesByRot

NumShootAnglesByRot

ShootAnglesByLin

NumShootAnglesByLin

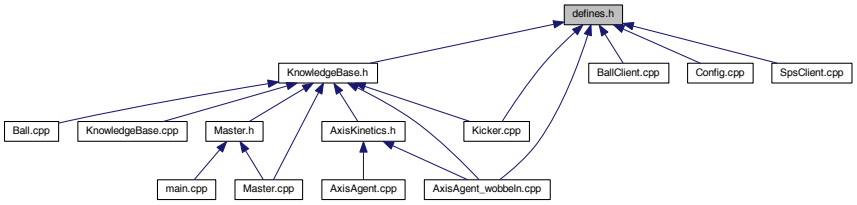
All

Definiert in Zeile 12 der Datei ConfigDefines.h.

6.15 defines.h-Dateireferenz

Enthält alle nötigen Defines.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Makrodefinitionen

- #define BUFFER_SIZE 256
- #define MAX_SHOOT_OFFSETS 10

Aufzählungen

- enum Gamer { COMPUTER, HUMAN }

  Spieler, Computer oder Mensch.
- enum Axis { AXIS_ONE, AXIS_TWO, AXIS_THREE, AXIS_FOUR }

  Spielstangen, erste bis vierte.
- enum Figure { FIGURE_ONE, FIGURE_TWO, FIGURE_THREE, FIGURE_FOUR, FIGURE_FIVE }

  Spielfiguren, erste bis fünfte.
- enum Coordinate { COORDINATE_X, COORDINATE_Y, COORDINATE_Z }

  Koordinaten, X, Y und Z.
- enum ShootOffset { SHOOT_ANGLE, SHOOT_OFFSET, SHOOT_ALL }

  Schießwinkel und zugehöriger Offset.
- enum Data_Value { POSITION, ANGLE }

  Um welchen Wert es sich handelt.
- enum Data_Type { VALUE, VELOCITY }

  Um welchen Wert es sich handelt.
- enum Safety { STOP, GRID }

  Um welche Sicherheitsfunktion es sich handelt.

Variablen

- const float PI = 3.14159265358979323846264338327 95f
- const float FIELDSIZE_X = 1200.f
- const float FIELDSIZE_Z = 690.f
- const float GOAL_WIDTH = 194.f
- const float GOAL_POSITION_POSITIVE_Z = FIELDSIZE_Z / 2 - GOAL_WIDTH / 4.f
- const float GOAL_POSITION_NEGATIVE_Z = FIELDSIZE_Z / 2 + GOAL_WIDTH / 4.f

6.15.1 Ausführliche Beschreibung

Enthält alle nötigen Defines.

Autor

Scharel Clemens

Datum

23.02.2012

Version

0.1 Entwicklungsphase

Definiert in Datei defines.h.

6.15.2 Makro-Dokumentation

6.15.2.1 #define BUFFER_SIZE 256

Definiert in Zeile 12 der Datei defines.h.

Wird benutzt von BallClient::BallClient(), Config::Config() und SpsClient::SpsClient().

6.15.2.2 #define MAX_SHOOT_OFFSETS 10

Definiert in Zeile 62 der Datei defines.h.

Wird benutzt von Config::getConfFromFile().

6.15.3 Dokumentation der Aufzählungstypen

6.15.3.1 enum Axis

Spielstangen, erste bis vierte.

Aufzählungswerte:

AXIS_ONE
AXIS_TWO
AXIS_THREE
AXIS_FOUR

Definiert in Zeile 30 der Datei defines.h.

6.15.3.2 enum Coordinate

Koordinaten, X, Y und Z.

Aufzählungswerte:

COORDINATE_X
COORDINATE_Y
COORDINATE_Z

Definiert in Zeile 55 der Datei defines.h.

6.15.3.3 enum Data_Type

Um welchen Wert es sich handelt.

Wird verwendet um in einem Array auf die entsprechenden Daten zu verweisen.

Aufzählungswerte:

VALUE Position der Spielstange
VELOCITY Geschwindigkeit der Spielstange

Definiert in Zeile 95 der Datei defines.h.

6.15.3.4 enum Data_Value

Um welchen Wert es sich handelt.

Wird verwendet um in einem Array auf die entsprechenden Daten zu verweisen.

Aufzählungswerte:

POSITION Position der Spielstange
ANGLE Winkel der Spielstange

Definiert in Zeile 81 der Datei defines.h.

6.15.3.5 enum Figure

Spielfiguren, erste bis fünfte.

Aufzählungswerte:

FIGURE_ONE
FIGURE_TWO
FIGURE_THREE
FIGURE_FOUR
FIGURE_FIVE

Definiert in Zeile 42 der Datei defines.h.

6.15.3.6 enum Gamer

Spieler, Computer oder Mensch.

Aufzählungswerte:

COMPUTER
HUMAN

Definiert in Zeile 20 der Datei defines.h.

6.15.3.7 enum Safety

Um welche Sicherheitsfunktion es sich handelt.

Wird verwendet um in einem Array auf die entsprechenden Daten zu verweisen.

Aufzählungswerte:

STOP Notaus
GRID Lichtgitter (Eingreifschutz)

Definiert in Zeile 109 der Datei defines.h.

6.15.3.8 enum ShootOffset

Schießwinkel und zugehöriger Offset.

Aufzählungswerte:

SHOOT_ANGLE
SHOOT_OFFSET
SHOOT_ALL

Definiert in Zeile 68 der Datei defines.h.

6.15.4 Variablen-Dokumentation

6.15.4.1 const float FIELDSIZE_X = 1200.f

Definiert in Zeile 121 der Datei defines.h.

Wird benutzt von AxisAgent::runAgent().

6.15.4.2 const float FIELDSIZE_Z = 690.f

Definiert in Zeile 122 der Datei defines.h.

Wird benutzt von AxisAgent::runAgent().

6.15.4.3 const float GOAL_POSITION_NEGATIVE_Z = FIELDSIZE_Z / 2 + GOAL_WIDTH / 4.f

Definiert in Zeile 125 der Datei defines.h.

Wird benutzt von AxisAgent::runAgent().

6.15.4.4 const float GOAL_POSITION_POSITIVE_Z = FIELDSIZE_Z / 2 - GOAL_WIDTH / 4.f

Definiert in Zeile 124 der Datei defines.h.

Wird benutzt von AxisAgent::runAgent().

6.15.4.5 const float GOAL_WIDTH = 194.f

Definiert in Zeile 123 der Datei defines.h.

6.15.4.6 const float PI = 3.14159265358979323846264338327 95f

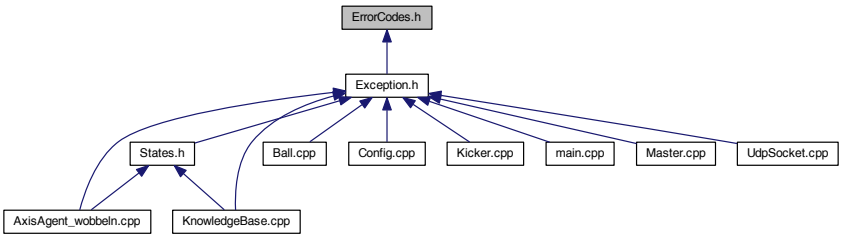
Definiert in Zeile 14 der Datei defines.h.

Wird benutzt von findAngleFromBallToGoal() und moveFeetInCircularMotion().

6.16 ErrorCodes.h-Dateireferenz

Enthält die Defines der Fehlercodes.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Makrodefinitionen

- #define ERROR_NONE 000

  Fehlercode wenn kein Fehler vorliegt.
- #define ERROR_UNKNOWN 001

  Fehlercode wenn ein unbekannter Fehler aufgetreten ist.
- #define ERROR_FILE 002
- #define ERROR_CONFIG 003

- #define ERROR_MUTEX 004
- #define ERROR_MUTEX_WAIT 005
- #define ERROR_UDP_NO_HOST 006
- #define ERROR_THREAD 007
- #define ERROR_BALL 100
- #define ERROR_KICKER 200
- #define ERROR_AXIS_BLOCK 201
- #define ERROR_KNOWLEDGEBASE 300
- #define ERROR_STATE 301

6.16.1 Ausführliche Beschreibung

Enthält die Defines der Fehlercodes.

Autor

Scharel Clemens

Datum

22.03.2012

Version

0.1 Entwicklungsphase

Definiert in Datei ErrorCodes.h.

6.16.2 Makro-Dokumentation

6.16.2.1 #define ERROR_AXIS_BLOCK 201

Definiert in Zeile 27 der Datei ErrorCodes.h.

6.16.2.2 #define ERROR_BALL 100

Definiert in Zeile 23 der Datei ErrorCodes.h.

Wird benutzt von Ball::Ball().

6.16.2.3 #define ERROR_CONFIG 003

Definiert in Zeile 16 der Datei ErrorCodes.h.

Wird benutzt von Ball::Ball(), Kicker::Kicker() und Config::start().

6.16.2.4 #define ERROR_FILE 002

Definiert in Zeile 15 der Datei ErrorCodes.h.

6.16.2.5 #define ERROR_KICKER 200

Definiert in Zeile 26 der Datei ErrorCodes.h.

Wird benutzt von Kicker::Kicker().

6.16.2.6 #define ERROR_KNOWLEDGEBASE 300

Definiert in Zeile 30 der Datei ErrorCodes.h.

6.16.2.7 #define ERROR_MUTEX 004

Definiert in Zeile 17 der Datei ErrorCodes.h.

6.16.2.8 #define ERROR_MUTEX_WAIT 005

Definiert in Zeile 18 der Datei ErrorCodes.h.

Wird benutzt von State::get() und State::set().

6.16.2.9 #define ERROR_NONE 000

Fehlercode wenn kein Fehler vorliegt.

Default Wert, erst wenn ein Fehler auftritt wird ein anderer Wert gesetzt.

6.16.2.10 #define ERROR_STATE 301

Definiert in Zeile 31 der Datei ErrorCodes.h.

6.16.2.11 #define ERROR_THREAD 007

Definiert in Zeile 20 der Datei ErrorCodes.h.

6.16.2.12 #define ERROR_UDP_NO_HOST 006

Definiert in Zeile 19 der Datei ErrorCodes.h.

6.16.2.13 #define ERROR_UNKNOWN 001

Fehlercode wenn ein unbekannter Fehler aufgetreten ist.

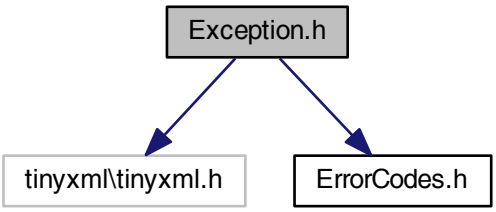
Wird benutzt von Exception::getErrStr().

6.17 Exception.h-Dateireferenz

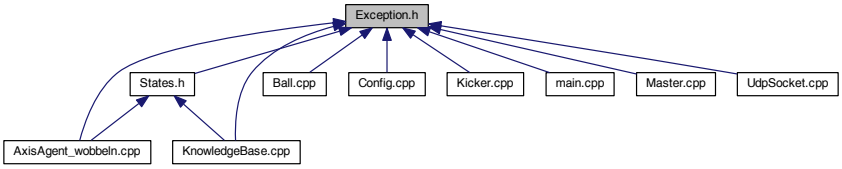
Enthält die Definition der Exception-Klassen.

```
#include "tinyxml\tinyxml.h"
#include "ErrorCodes.h"
```

Include-Abhängigkeitsdiagramm für Exception.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Exception

  Basis-Klasse von der alle Exception-Klassen abgeleitet werden sollten.
- class FileException

  Exception-Klasse für Dateizugriff.
- class ConfigException

  Exception-Klasse für die Konfigurationsdateien.
- class MutexException

  Exception-Klasse für die Mutexzugriffe.
- class HostException

  Exception-Klasse für die Host-Auflösung.
- class ThreadException

  Exception-Klasse für Threads.
- class StateException

  Exception-Klasse für Zustände.

Typdefinitionen

- typedef unsigned long DWORD

6.17.1 Ausführliche Beschreibung

Enthält die Definition der Exception-Klassen.

Autor

Scharel Clemens

Datum

03.04.2012

Version

0.2 Entwicklungsphase

Definiert in Datei Exception.h.

6.17.2 Dokumentation der benutzerdefinierten Typen

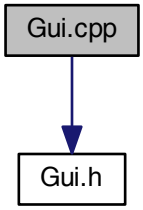
6.17.2.1 typedef unsigned long DWORD

Definiert in Zeile 12 der Datei Exception.h.

6.18 Gui.cpp-Dateireferenz

Enthält die Implementierung der Klasse Gui.

`#include "Gui.h"`
Include-Abhängigkeitsdiagramm für Gui.cpp:



6.18.1 Ausführliche Beschreibung

Enthält die Implementierung der Klasse Gui.

Autor

Scharel Clemens

Datum

22.03.2012

Version

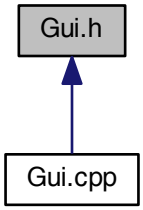
0.1 Entwicklungsphase

Definiert in Datei Gui.cpp.

6.19 Gui.h-Dateireferenz

Enthält die Definition der Klasse Gui.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Gui

6.19.1 Ausführliche Beschreibung

Enthält die Definition der Klasse Gui.

Autor

Scharel Clemens

Datum

22.03.2012

Version

0.1 Entwicklungsphase

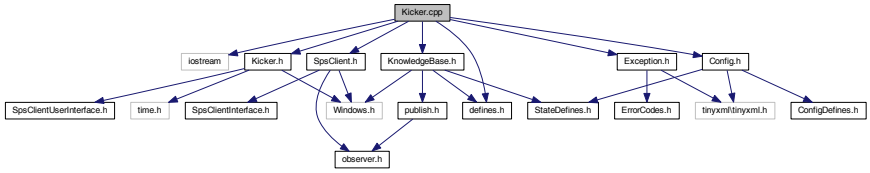
Definiert in Datei Gui.h.

6.20 Kicker.cpp-Dateireferenz

Enthält die Implementierung der Klasse Kicker.

```
#include <iostream>
#include "Kicker.h"
#include "SpsClient.h"
#include "Exception.h"
#include "Config.h"
#include "KnowledgeBase.h"
#include "defines.h"
```

Include-Abhängigkeitsdiagramm für Kicker.cpp:



6.20.1 Ausführliche Beschreibung

Enthält die Implementierung der Klasse Kicker.

Autor

Scharel Clemens

Datum

02.04.2012

Version

0.2 Entwicklungsphase

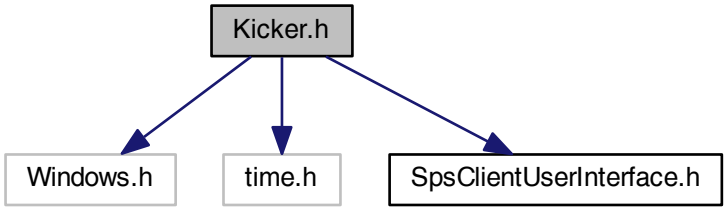
Definiert in Datei Kicker.cpp.

6.21 Kicker.h-Dateireferenz

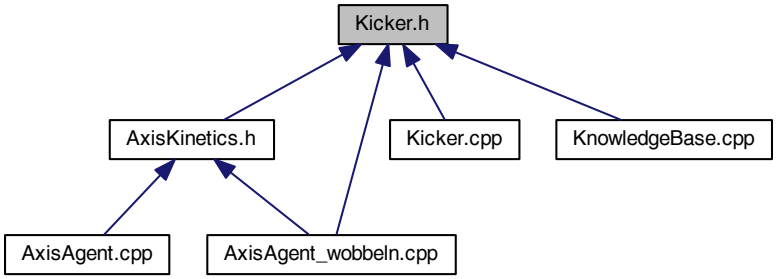
Enthält die Definition der Klasse Kicker.

```
#include <Windows.h>
#include <time.h>
#include "SpsClientUserInterface.h"
```

Include-Abhängigkeitsdiagramm für Kicker.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Kicker

  Eine Klasse, die den UDP-Client anlegt und benutzt.

6.21.1 Ausführliche Beschreibung

Enthält die Definition der Klasse Kicker.

Autor

Scharel Clemens

Datum

02.04.2012

Version

0.2 Entwicklungsphase

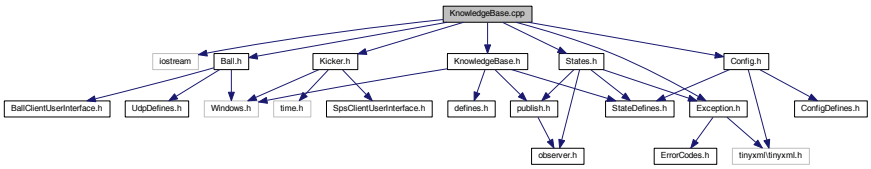
Definiert in Datei Kicker.h.

6.22 KnowledgeBase.cpp-Dateireferenz

Enthält die Implementierung der Klasse KnowledgeBase.

```
#include <iostream>
#include "KnowledgeBase.h"
#include "Exception.h"
#include "Config.h"
#include "Ball.h"
#include "Kicker.h"
#include "States.h"
```

Include-Abhängigkeitsdiagramm für KnowledgeBase.cpp:



6.22.1 Ausführliche Beschreibung

Enthält die Implementierung der Klasse KnowledgeBase.

Autor

Scharel Clemens

Datum

26.03.2012

Version

0.1 Entwicklungsphase

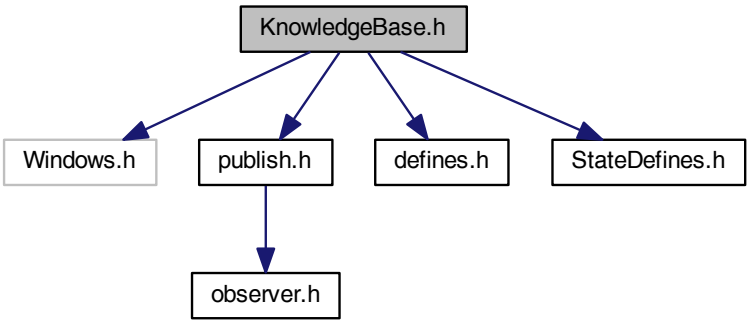
Definiert in Datei KnowledgeBase.cpp.

6.23 KnowledgeBase.h-Dateireferenz

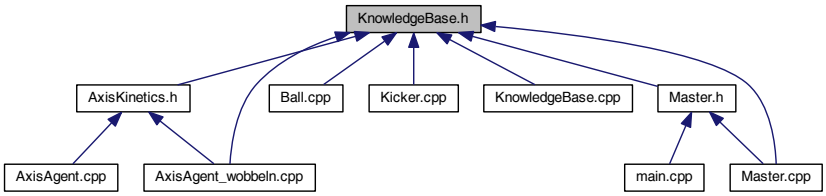
Enthält die Definition der Klasse KnowledgeBase.

```
#include <Windows.h>
#include "publish.h"
#include "defines.h"
#include "StateDefines.h"
```

Include-Abhängigkeitsdiagramm für KnowledgeBase.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class KnowledgeBase

6.23.1 Ausführliche Beschreibung

Enthält die Definition der Klasse KnowledgeBase.

Autor

Scharel Clemens

Datum

26.03.2012

Version

0.1 Entwicklungsphase

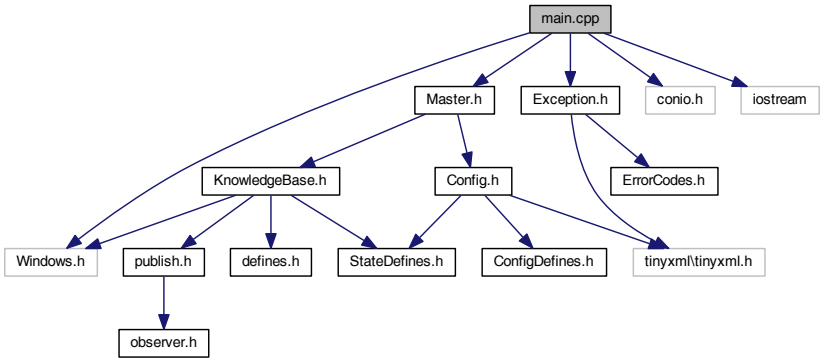
Definiert in Datei KnowledgeBase.h.

6.24 main.cpp-Dateireferenz

Enthält die main-Funktion.

```
#include <Windows.h>
#include <conio.h>
#include <iostream>
#include "Exception.h"
#include "Master.h"
```

Include-Abhängigkeitsdiagramm für main.cpp:



Funktionen

- int main (int argc, char *argv[])

  Startet das Programm.

6.24.1 Ausführliche Beschreibung

Enthält die main-Funktion.

Autor

Scharel Clemens

Datum

22.03.2012

Version

0.1 Entwicklungsphase

Definiert in Datei main.cpp.

6.24.2 Dokumentation der Funktionen

6.24.2.1 int main (int *argc*, char * *argv[]*)

Startet das Programm.

Parameter

in	*argc*	Anzahl der an die Applikation übergebenen Parameter
in	*argv*	Pointer auf den Beginn des Array mit den Parametern

Rückgabe

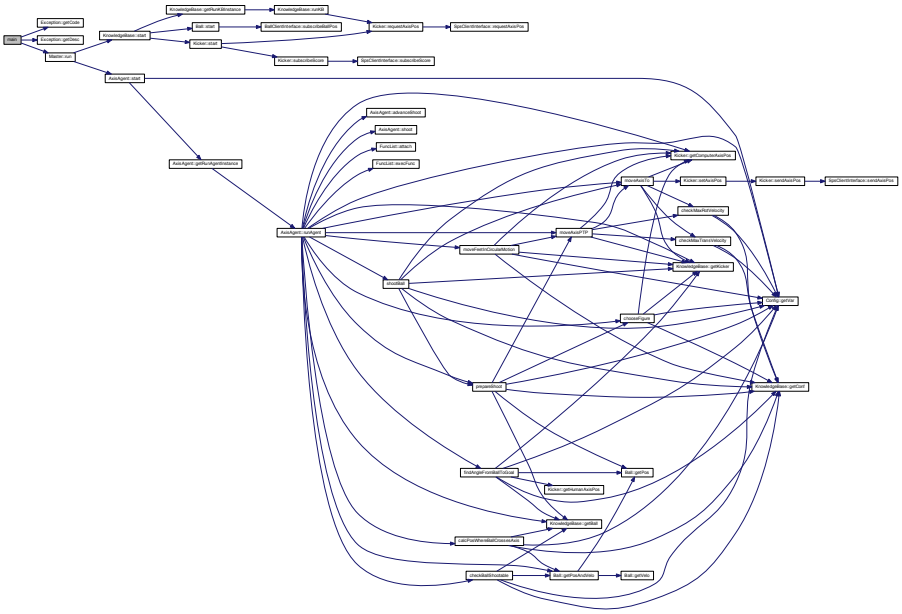
EXIT_SUCCESS wenn die Applikation sachgemäß beendet wird

Legt die Klasse Master an.

Definiert in Zeile 28 der Datei main.cpp.

Benutzt Exception::getCode(), Exception::getDesc() und Master::run().

Hier ist ein Graph der zeigt, was diese Funktion aufruft:

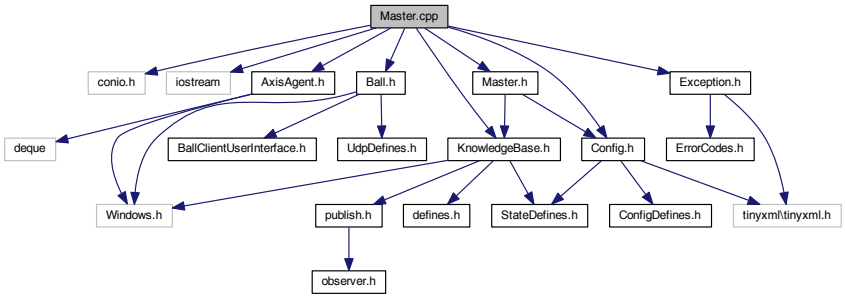


6.25 Master.cpp-Dateireferenz

Enthält die Implementierung der Klasse Master.

```
#include <conio.h>
#include <iostream>
#include "Master.h"
#include "KnowledgeBase.h"
#include "AxisAgent.h"
#include "Config.h"
#include "Ball.h"
#include "Exception.h"
```

Include-Abhängigkeitsdiagramm für Master.cpp:



6.25.1 Ausführliche Beschreibung

Enthält die Implementierung der Klasse Master.

Autor

Scharel Clemens

Datum

22.03.2012

Version

0.1 Entwicklungsphase

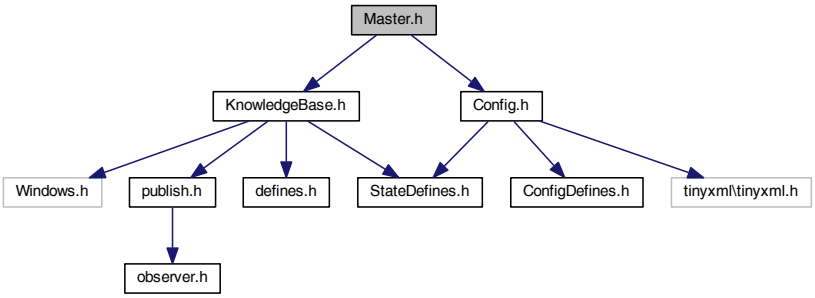
Definiert in Datei Master.cpp.

6.26 Master.h-Dateireferenz

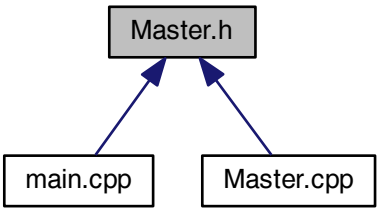
Enthält die Definition der Klasse Master.

```
#include "KnowledgeBase.h"
#include "Config.h"
```

Include-Abhängigkeitsdiagramm für Master.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Master

6.26.1 Ausführliche Beschreibung

Enthält die Definition der Klasse Master.

Autor

Scharel Clemens

Datum

22.03.2012

Version

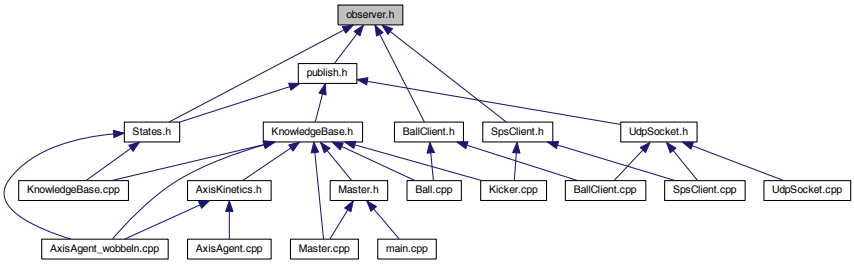
0.1 Entwicklungsphase

Definiert in Datei Master.h.

6.27 observer.h-Dateireferenz

Enthält die Definition und Implementierung der Klasse Observer.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Observer

  Observer Klasse für das Observer Entwurfsmodell.

6.27.1 Ausführliche Beschreibung

Enthält die Definition und Implementierung der Klasse Observer.

Autor

Scharel Clemens

Datum

08.08.2011

Version

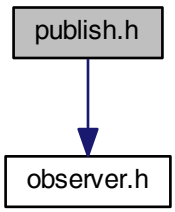
0.1 Entwicklungsphase

Definiert in Datei observer.h.

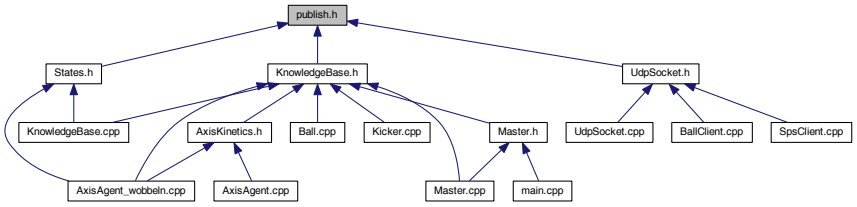
6.28 publish.h-Dateireferenz

Enthält die Definition und Implementierung der Klasse Publish.

`#include "observer.h"`
Include-Abhängigkeitsdiagramm für publish.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- struct ObsListKnot
- class Publish

  Publisher Klasse für das Observer Entwurfsmodell.

Makrodefinitionen

- #define NULL 0

Typdefinitionen

- typedef struct ObsListKnot ObsListKnot_t

6.28.1 Ausführliche Beschreibung

Enthält die Definition und Implementierung der Klasse Publish.

Autor

Scharel Clemens

Datum

10.01.2012

Version

1.0 Grundfunktion vollständig implementiert

Definiert in Datei publish.h.

6.28.2 Makro-Dokumentation

6.28.2.1 #define NULL 0

Definiert in Zeile 15 der Datei publish.h.

Wird benutzt von Publish::attach(), FuncList::attach(), AxisAgent::AxisAgent(), Ball::Ball(), BallClient::BallClient(), Config::Config(), KnowledgeBase::deleteStates(), Publish::detach(), FuncList::FuncList(), UDPsocket::getAddrFromString(), Config::getAxisDistance(), Config::getAxisOffset(), Config::getAxisVal(), Config::getConnectionInfo(), Config::getDimension(), Config::getFigureVal(), Config::getGoalSize(), UDPsocket::getRecvDataInstance(), AxisAgent::getRunAgentInstance(), KnowledgeBase::getRunKBInstance(), Config::getShootAngleInfo(), Config::getStateInfo(), KnowledgeBase::initStates(), Kicker::Kicker(), Publish::Publish(), AxisAgent::runAgent(), SpsClient::SpsClient(), KnowledgeBase::start(), UDPsocket::start(), AxisAgent::start(), State::State(), BallClient::~BallClient(), FuncList::~FuncList(), KnowledgeBase::~KnowledgeBase(), SpsClient::~SpsClient() und UDPsocket::~UDPsocket().

6.28.3 Dokumentation der benutzerdefinierten Typen

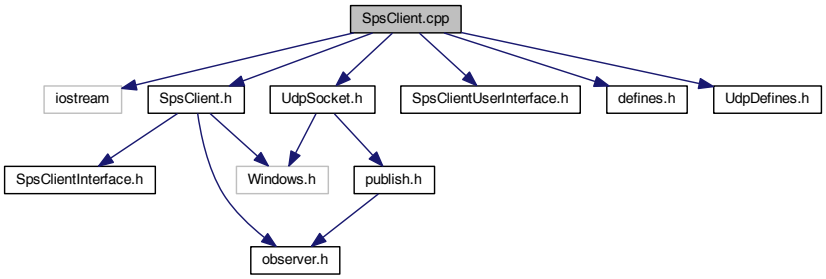
6.28.3.1 typedef struct ObsListKnot ObsListKnot_t

6.29 SpsClient.cpp-Dateireferenz

Enthält die Implementierung der Klasse Client.

```
#include <iostream>
#include "SpsClient.h"
#include "UdpSocket.h"
#include "SpsClientUserInterface.h"
#include "defines.h"
#include "UdpDefines.h"
```

Include-Abhängigkeitsdiagramm für SpsClient.cpp:



6.29.1 Ausführliche Beschreibung

Enthält die Implementierung der Klasse Client.

Autor

Scharel Clemens

Datum

03.04.2012

Version

0.2 Testphase

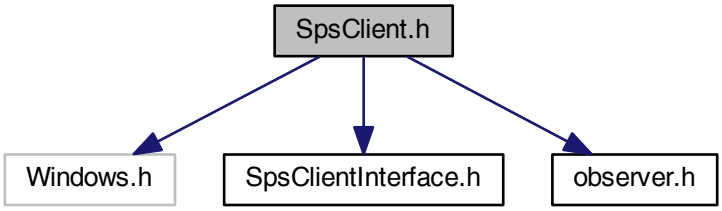
Definiert in Datei SpsClient.cpp.

6.30 SpsClient.h-Dateireferenz

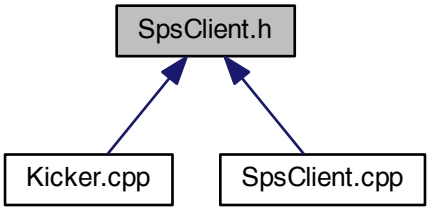
Enthält die Definition der Klasse Client.

```
#include <Windows.h>
#include "SpsClientInterface.h"
#include "observer.h"
```

Include-Abhängigkeitsdiagramm für SpsClient.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class SpsClient

  ein UDP-Client

6.30.1 Ausführliche Beschreibung

Enthält die Definition der Klasse Client.

Autor

Scharel Clemens

Datum

03.04.2012

Version

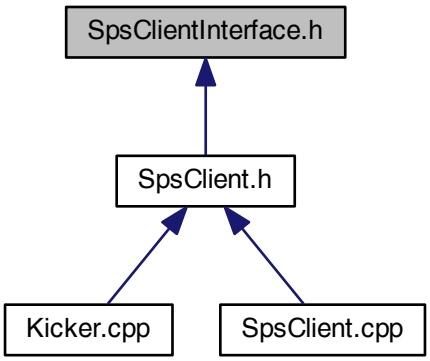
0.2 Testphase

Definiert in Datei SpsClient.h.

6.31 SpsClientInterface.h-Dateireferenz

Enthält das Interface zur Kommunikation mit dem UDP-Client.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class SpsClientInterface

6.31.1 Ausführliche Beschreibung

Enthält das Interface zur Kommunikation mit dem UDP-Client.

Autor

Scharel Clemens

Datum

08.02.2012

Version

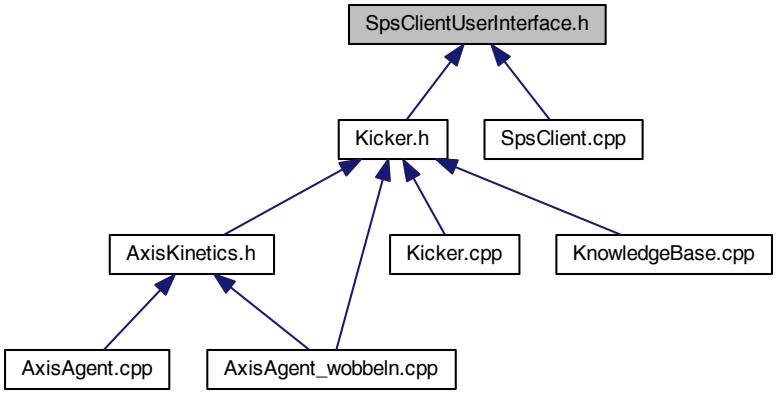
0.1 Testphase

Definiert in Datei SpsClientInterface.h.

6.32 SpsClientUserInterface.h-Dateireferenz

Enthält das Interface zur Kommunikation mit dem Benutzer des UDP-Clients.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- interface SpsClientUserInterface

  Interface zur Kommunikation mit dem Benutzer des UDP-Clients.

6.32.1 Ausführliche Beschreibung

Enthält das Interface zur Kommunikation mit dem Benutzer des UDP-Clients.

Autor

Scharel Clemens

Datum

08.02.2012

Version

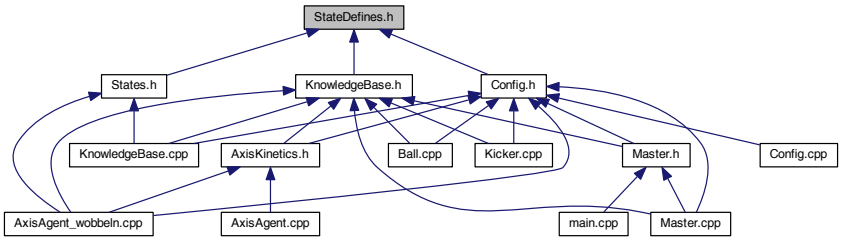
0.1 Testphase

Definiert in Datei SpsClientUserInterface.h.

6.33 StateDefines.h-Dateireferenz

Enthält die Defines der Klasse State.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Makrodefinitionen

- #define MAX_STATES_PER_GROUP 5

Aufzählungen

- enum StateVals { STATE_VALUE, STATE_MIN, STATE_MAX }
- enum StateIDs {
  STATES_GAME, STATES_GLOBAL, STATES_CONFIG, STATES_KNOWLEDGEBASE,
  STATES_BALL, STATES_KICKER, STATES_AXIS1COMPUTER, STATES_AXIS2COMPUTER,
  STATES_AXIS3COMPUTER, STATES_AXIS4COMPUTER, STATES_AXIS1HUMAN, STATES_AXIS2HUMAN,
  STATES_AXIS3HUMAN, STATES_AXIS4HUMAN, STATES_BALLPOS, STATES_BALLPOSTOAXIS1COMPUTER,
  STATES_BALLPOSTOAXIS2COMPUTER, STATES_BALLPOSTOAXIS3COMPUTER, STATES_BALLPOSTOAXIS4COMPUTER, STATES_BALLPOSTOAXIS1HUMAN,
  STATES_BALLPOSTOAXIS2HUMAN, STATES_BALLPOSTOAXIS3HUMAN, STATES_BALLPOSTOAXIS4HUMAN, STATES_ALL }

  Alle möglichen Status-Gruppen.
- enum GameStates { GAME_RUNNING, GAME_COMPUTERSCORE, GAME_HUMANSCORE, GAME_ALL }

  Zustände betreffend den Spielablauf.
- enum GlobalStates { GLOBAL_READY, GLOBAL_ALL }

  Globale Zustände.
- enum ConfigStates { CONFIG_READY, CONFIG_ALL }

  Zustände betreffend die Konfiguration.
- enum KnowledgeBaseStates { KNOWLEDGEBASE_READY, KNOWLEDGEBASE_ALL }

  Zustände betreffend die Wissensbasis.
- enum BallStates { BALL_READY, BALL_ALL }

  Zustände betreffend der Ballerkennung.
- enum KickerStates { KICKER_READY, KICKER_ALL }

  Zustände betreffend der SPSchnittstelle.
- enum AxisStates {
  AXIS_READY, AXIS_BLOCKING, AXIS_MOVING, AXIS_SHOOTING,
  AXIS_WOBBLING, AXIS_ALL }

  Zustände betreffend betreffend der 4 Spielstangen.
- enum BallPosStates { BALLPOS_COMPUTER_POSSESSION, BALLPOS_HUMAN_POSSESSION, BALLPOS_SPEED, BALLPOS_ALL }

  Zustände betreffend der Ballposition.

- enum BallPosToAxisStates {
  BALLPOSTOAXIS_HOLDABLE, BALLPOSTOAXIS_PASSABLE, BALLPOSTOAXIS_SHOOTABLE, BALLPOSTOAXIS_HANDABLE,
  BALLPOSTOAXIS_ALL }

  Zustände betreffend der Ballposition gegenüber der Spielstangen.

6.33.1 Ausführliche Beschreibung

Enthält die Defines der Klasse State.

Autor

Scharel Clemens

Datum

07.06.2012

Version

0.1 Entwicklungsphase

Definiert in Datei StateDefines.h.

6.33.2 Makro-Dokumentation

6.33.2.1 #define MAX_STATES_PER_GROUP 5

Definiert in Zeile 12 der Datei StateDefines.h.

Wird benutzt von Config::getConfFromFile(), Config::getState() und Config::setState().

6.33.3 Dokumentation der Aufzählungstypen

6.33.3.1 enum AxisStates

Zustände betreffend betreffend der 4 Spielstangen.

Zustände der Spielstangen, jeweils 4 Spielstangen pro Spieler.

Aufzählungswerte:

AXIS_READY Gibt an ob die AxisAgent bereit sind.

AXIS_BLOCKING Gibt an ob die Spielstange blockiert ist.

AXIS_MOVING Gibt an ob sich die Spielstange in positiver Richtung bewegt.

AXIS_SHOOTING Gibt an ob die Spielstange einen Schuss durchführt.

AXIS_WOBBLING Gibt an ob, und in welche Richtung die Spielstange "wobbelt".

AXIS_ALL Repräsentiert die Anzahl der Zustände, wird verwendet um Arrays zu erzeugen.

Definiert in Zeile 136 der Datei StateDefines.h.

6.33.3.2 enum BallPosStates

Zustände betreffend der Ballposition.

Aufzählungswerte:

BALLPOS_COMPUTER_POSSESSION Gibt an ob der Ball in Kontrolle des Computers ist.

BALLPOS_HUMAN_POSSESSION Gibt an ob der Ball in Kontrolle des Menschen ist.

BALLPOS_SPEED Gibt an wie schnell sich der Ball bewegt.

BALLPOS_ALL Repräsentiert die Anzahl der Zustände, wird verwendet um Arrays zu erzeugen.

Definiert in Zeile 156 der Datei StateDefines.h.

6.33.3.3 enum BallPosToAxisStates

Zustände betreffend der Ballposition gegenüber der Spielstangen.

Zustände der Ballposition gegenüber der Spielstangen, jeweils 4 Spielstangen pro Spieler.

Aufzählungswerte:

BALLPOSTOAXIS_HOLDABLE Gibt an ob der Ball von der entsprechenden Spielstange haltbar ist.

BALLPOSTOAXIS_PASSABLE Gibt an ob der Ball von der entsprechenden Spielstange durchgelassen werden kann.

BALLPOSTOAXIS_SHOOTABLE Gibt an ob der Ball von der entsprechenden Spielstange geschossen werden kann.

BALLPOSTOAXIS_HANDABLE Gibt an ob der Ball von der entsprechenden Spielstange kontrolliert werden kann.

BALLPOSTOAXIS_ALL Repräsentiert die Anzahl der Zustände, wird verwendet um Arrays zu erzeugen.

Definiert in Zeile 174 der Datei StateDefines.h.

6.33.3.4 enum BallStates

Zustände betreffend der Ballerkennung.

Aufzählungswerte:

BALL_READY Gibt an ob die Ballerkennung bereit ist.

BALL_ALL Repräsentiert die Anzahl der Zustände, wird verwendet um Arrays zu erzeugen.

Definiert in Zeile 110 der Datei StateDefines.h.

6.33.3.5 enum ConfigStates

Zustände betreffend die Konfiguration.

Aufzählungswerte:

CONFIG_READY Gibt an ob die Konfiguration bereit ist.

CONFIG_ALL Repräsentiert die Anzahl der Zustände, wird verwendet um Arrays zu erzeugen.

Definiert in Zeile 86 der Datei StateDefines.h.

6.33.3.6 enum GameStates

Zustände betreffend den Spielablauf.

Aufzählungswerte:

GAME_RUNNING Gibt an ob gerade gespielt wird.

GAME_COMPUTERSCORE Punktestand des Computers.

GAME_HUMANSCORE Punktestand des Menschen.

GAME_ALL Repräsentiert die Anzahl der Zustände, wird verwendet um Arrays zu erzeugen.

Definiert in Zeile 58 der Datei StateDefines.h.

6.33.3.7 enum GlobalStates

Globale Zustände.

Aufzählungswerte:

GLOBAL_READY Gibt an ob das gesamte System spielbereit ist.

GLOBAL_ALL Repräsentiert die Anzahl der Zustände, wird verwendet um Arrays zu erzeugen.

Definiert in Zeile 74 der Datei StateDefines.h.

6.33.3.8 enum KickerStates

Zustände betreffend der SPSchnittstelle.

Aufzählungswerte:

KICKER_READY Gibt an ob die SPSchnittstelle bereit ist.

KICKER_ALL Repräsentiert die Anzahl der Zustände, wird verwendet um Arrays zu erzeugen.

Definiert in Zeile 122 der Datei StateDefines.h.

6.33.3.9 enum KnowledgeBaseStates

Zustände betreffend die Wissensbasis.

Aufzählungswerte:

KNOWLEDGEBASE_READY Gibt an ob die Wissensbasis bereit ist.

KNOWLEDGEBASE_ALL Repräsentiert die Anzahl der Zustände, wird verwendet um Arrays zu erzeugen.

Definiert in Zeile 98 der Datei StateDefines.h.

6.33.3.10 enum StateIDs

Alle möglichen Status-Gruppen.

Aufzählungswerte:

STATES_GAME

STATES_GLOBAL

STATES_CONFIG

STATES_KNOWLEDGEBASE

STATES_BALL

STATES_KICKER

STATES_AXIS1COMPUTER

STATES_AXIS2COMPUTER

STATES_AXIS3COMPUTER

STATES_AXIS4COMPUTER

STATES_AXIS1HUMAN

STATES_AXIS2HUMAN

STATES_AXIS3HUMAN

STATES_AXIS4HUMAN

STATES_BALLPOS

STATES_BALLPOSTOAXIS1COMPUTER

STATES_BALLPOSTOAXIS2COMPUTER

STATES_BALLPOSTOAXIS3COMPUTER

STATES_BALLPOSTOAXIS4COMPUTER

STATES_BALLPOSTOAXIS1HUMAN

STATES_BALLPOSTOAXIS2HUMAN

STATES_BALLPOSTOAXIS3HUMAN

STATES_BALLPOSTOAXIS4HUMAN

STATES_ALL

Definiert in Zeile 25 der Datei StateDefines.h.

6.33.3.11 enum StateVals

Aufzählungswerte:

STATE_VALUE

STATE_MIN

STATE_MAX

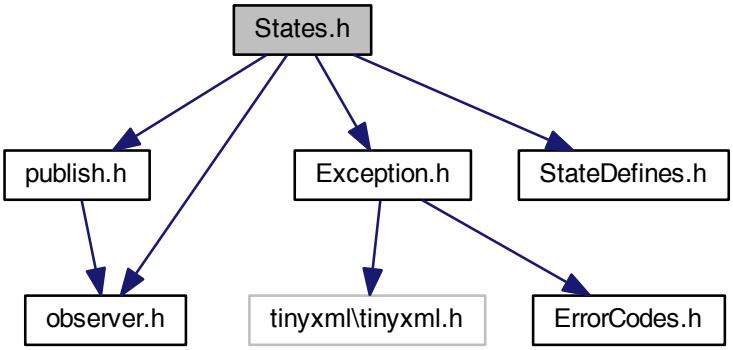
Definiert in Zeile 14 der Datei StateDefines.h.

6.34 States.h-Dateireferenz

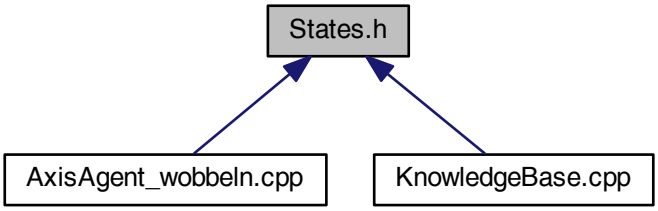
Enthält die Definition der einzelnen Zustände.

```
#include "publish.h"
#include "observer.h"
#include "Exception.h"
#include "StateDefines.h"
```

Include-Abhängigkeitsdiagramm für States.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class State

  Status einer Eigenschaft.

6.34.1 Ausführliche Beschreibung

Enthält die Definition der einzelnen Zustände.

Autor

Scharel Clemens

Datum

12.04.2012

Version

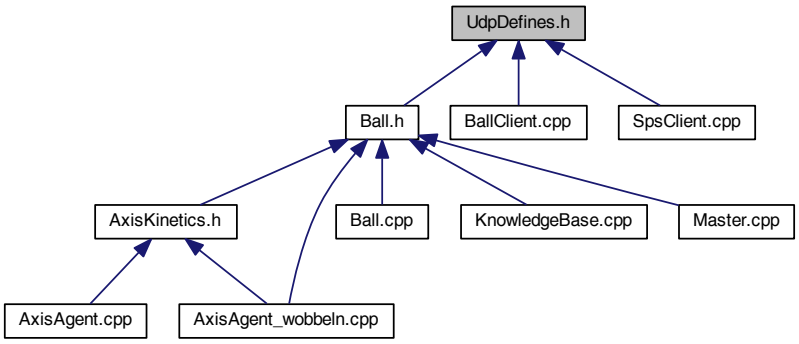
0.1 Entwicklungsphase

Definiert in Datei States.h.

6.35 UdpDefines.h-Dateireferenz

Enthält die defines der verschiedenen Strukturen.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- union ubf16_request

  Bitfeld, bei dem jedes Bit bestimmte Daten anfordert.
- struct ubf16_request::bitfeld_16
- struct requestPacket

  Anfrage-Paket für bestimmte Daten.
- struct bool_1

  Anfrage-Paket für bestimmte Daten.
- struct bool_2

  Anfrage-Paket für bestimmte Daten.
- struct unsignedShort_2

  Anfrage-Paket für bestimmte Daten.
- struct float_1

  Anfrage-Paket für bestimmte Daten.
- struct float_4

  Anfrage-Paket für bestimmte Daten.
- struct float_2_4

  Anfrage-Paket für bestimmte Daten.
- struct float_2_2_4

  Anfrage-Paket für bestimmte Daten.
- struct ballCoordinate_t

  Struktur der Ballposition.
- struct ballPacket

  Paket mit der Ballposition.

Typdefinitionen

- typedef union ubf16_request ubf16_request

  Bitfeld, bei dem jedes Bit bestimmte Daten anfordert.
- typedef struct requestPacket requestPacket

  Anfrage-Paket für bestimmte Daten.
- typedef struct bool_1 bool_1

  Anfrage-Paket für bestimmte Daten.
- typedef struct bool_2 bool_2

  Anfrage-Paket für bestimmte Daten.
- typedef struct unsignedShort_2 unsignedShort_2

  Anfrage-Paket für bestimmte Daten.
- typedef struct float_1 float_1

  Anfrage-Paket für bestimmte Daten.
- typedef struct float_4 float_4

  Anfrage-Paket für bestimmte Daten.
- typedef struct float_2_4 float_2_4

  Anfrage-Paket für bestimmte Daten.
- typedef struct float_2_2_4 float_2_2_4

  Anfrage-Paket für bestimmte Daten.
- typedef struct ballCoordinate_t ballCoordinate_t

  Struktur der Ballposition.
- typedef struct ballPacket ballPacket

  Paket mit der Ballposition.

Aufzählungen

- enum PacketID {
  SUBSCRIBE_SCORE, SUBSCRIBE_BALLSQUEEZE, SUBSCRIBE_SAFETY, REQUEST, SET_AXISPOS, SET_AXISRAMP, SET_STACKMODE, SET_SCORE, GET_HUMAN_AXISPOS, GET_COMPUTER_AXISPOS, GET_BOTH_AXISPOS, GET_AXISRAMP, GET_STACKMODE, GET_BALLSQUEEZE, GET_SCORE, GET_SAFETY, SUBSCRIBE_BALLPOS, REQUEST_BALLPOS, GET_BALLPOS }

  ID eines UDP-Pakets.

Variablen

- const int TRAVERSE_DISTANCE_1 = 241

  Verfahrweg des Torwards.
- const int TRAVERSE_DISTANCE_2 = 377

  Verfahrweg des Abwehr.
- const int TRAVERSE_DISTANCE_3 = 233

  Verfahrweg des Mittelfeld.
- const int TRAVERSE_DISTANCE_5 = 120

  Verfahrweg des Angriff.
- const float MAX_VELOCITY_TRANS = 2000.f

  Maximalgeschwindigkeit der Translation.
- const float MAX_VELOCITY_ROT = 4285.f

  Maximalgeschwindigkeit der Rotation.

6.35.1 Ausführliche Beschreibung

Enthält die defines der verschiedenen Strukturen.

Autor

Scharel Clemens

Datum

21.03.2012

Version

0.1 Testphase

Definiert in Datei UdpDefines.h.

6.35.2 Dokumentation der benutzerdefinierten Typen

6.35.2.1 typedef struct ballCoordinate_t ballCoordinate_t

Struktur der Ballposition.

6.35.2.2 typedef struct ballPacket ballPacket

Paket mit der Ballposition.

Beinhaltet die Paket-ID und eine Struktur mit der Ballposition

6.35.2.3 typedef struct bool_1 bool_1

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein bool-Wert.

6.35.2.4 typedef struct bool_2 bool_2

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und zwei bool-Werte.

6.35.2.5 typedef struct float_1 float_1

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein float-Wert.

6.35.2.6 typedef struct float_2_2_4 float_2_2_4

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein dreidimensionales Array aus zwei mal zwei mal vier float-Werten.

6.35.2.7 typedef struct float_2_4 float_2_4

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein zweidimensionales Array aus zwei mal vier float-Werten.

6.35.2.8 typedef struct float_4 float_4

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein Array aus vier float-Werten.

6.35.2.9 typedef struct requestPacket requestPacket

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein Ánfrage-Bitfeld ubf8_request.

6.35.2.10 typedef union ubf16_request ubf16_request

Bitfeld, bei dem jedes Bit bestimmte Daten anfordert.

6.35.2.11 typedef struct unsignedShort_2 unsignedShort_2

Anfrage-Paket für bestimmte Daten.

Beinhaltet die Paket-ID und ein Array aus zwei unsigned short-Werten.

6.35.3 Dokumentation der Aufzählungstypen

6.35.3.1 enum PacketID

ID eines UDP-Pakets.

Jedes UDP-Paket hat eine eindeutige ID, die beschreibt welche Daten enthalten sind.

Aufzählungswerte:

SUBSCRIBE_SCORE Zum Abonnieren des Spielstands

SUBSCRIBE_BALLSQUEEZE Zum Abonnieren der Information ob ein Ball eingeklemmt ist

SUBSCRIBE_SAFETY Zum Abonnieren der Informationen über die Sicherheitsfunktionen

REQUEST Anfrage für verschiedene Daten

SET_AXISPOS Veranlasst die SPS die Spielstangen an eine bestimmte Position zu fahren

SET_AXISRAMP Setzt die Beschleunigung der Spielstangen

SET_STACKMODE Setzt den Modus in dem eintreffende Steuerbefehle nacheinander ausgeführt und nicht überschrieben werden

SET_SCORE Setzt den Spielstand

GET_HUMAN_AXISPOS Enthält die Spielstangenpositionen des menschlichen Spielers

GET_COMPUTER_AXISPOS Enthält die Spielstangenpositionen des Computers

GET_BOTH_AXISPOS Enthält die Spielstangenpositionen beider Spieler

GET_AXISRAMP Enthält die Beschleunigung der Spielstangen vom Computer

GET_STACKMODE Enthält die Information ob der Modus, in dem eintreffende Steuerbefehle nacheinander ausgeführt und nicht überschrieben werden, gesetzt ist

GET_BALLSQUEEZE Enthält die Information ob der Ball eingeklemmt ist

GET_SCORE Enthält den Spielstand

GET_SAFETY Enthält die Sicherheitsinformationen

SUBSCRIBE_BALLPOS Zum Abonnieren der Ballposition

REQUEST_BALLPOS Anfrage für die Ballposition

GET_BALLPOS Enthält die Ballposition

Definiert in Zeile 52 der Datei UdpDefines.h.

6.35.4 Variablen-Dokumentation

6.35.4.1 const float MAX_VELOCITY_ROT = 4285.f

Maximalgeschwindigkeit der Rotation.

Maximalgeschwindigkeit der Rotation, Grad/sek.

Definiert in Zeile 44 der Datei UdpDefines.h.

6.35.4.2 const float MAX_VELOCITY_TRANS = 2000.f

Maximalgeschwindigkeit der Translation.

Maximalgeschwindigkeit der Translation in mm/sek.

Definiert in Zeile 37 der Datei UdpDefines.h.

6.35.4.3 const int TRAVERSE_DISTANCE_1 = 241

Verfahrweg des Torwards.

Definiert in Zeile 15 der Datei UdpDefines.h.

Wird benutzt von AxisAgent::runAgent().

6.35.4.4 const int TRAVERSE_DISTANCE_2 = 377

Verfahrweg des Abwehr.

Definiert in Zeile 20 der Datei UdpDefines.h.

Wird benutzt von AxisAgent::runAgent().

6.35.4.5 const int TRAVERSE_DISTANCE_3 = 233

Verfahrweg des Mittelfeld.

Definiert in Zeile 25 der Datei UdpDefines.h.

Wird benutzt von AxisAgent::runAgent().

6.35.4.6 const int TRAVERSE_DISTANCE_5 = 120

Verfahrweg des Angriff.

Definiert in Zeile 30 der Datei UdpDefines.h.

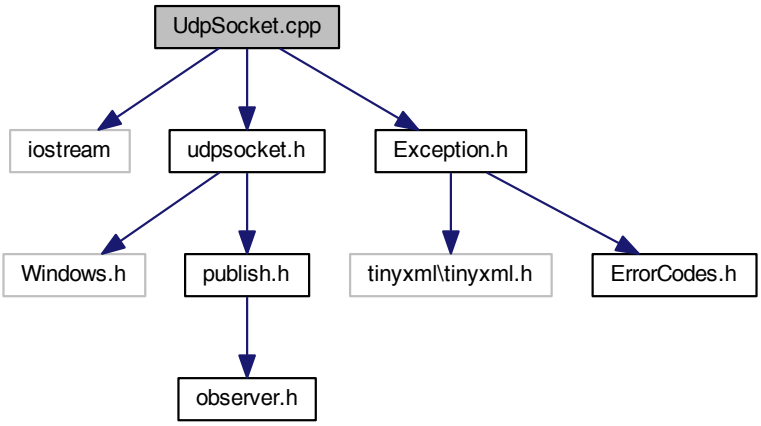
Wird benutzt von AxisAgent::runAgent().

6.36 UdpSocket.cpp-Dateireferenz

Enthält die Implementierung der Klasse UDPsocket.

```
#include <iostream>
#include "udpsocket.h"
#include "Exception.h"
```

Include-Abhängigkeitsdiagramm für UdpSocket.cpp:



6.36.1 Ausführliche Beschreibung

Enthält die Implementierung der Klasse UDPsocket.

Autor

Scharel Clemens

Datum

17.01.2012

Version

1.0 Grundfunktion vollständig implementiert

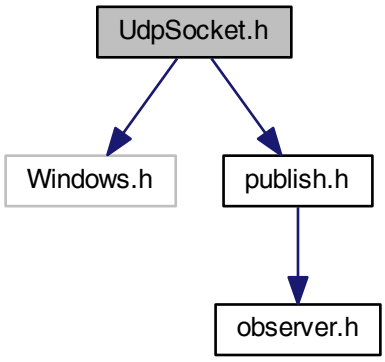
Definiert in Datei UdpSocket.cpp.

6.37 UdpSocket.h-Dateireferenz

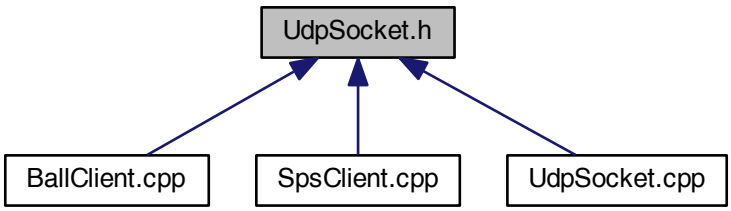
Enthält die Definition der Klasse UDPsocket.

```
#include <Windows.h>
#include "publish.h"
```

Include-Abhängigkeitsdiagramm für UdpSocket.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class UDPsocket

  Stellt ein UDP-Socket dar.

6.37.1 Ausführliche Beschreibung

Enthält die Definition der Klasse UDPsocket.

Autor

Scharel Clemens

Datum

17.01.2012

Version

1.0 Grundfunktion vollständig implementiert

Definiert in Datei UdpSocket.h.

Index

~AxisAgent
AxisAgent, 11, 12
~Ball
Ball, 21
~BallClient
BallClient, 30
~BallClientInterface
BallClientInterface, 35
~BallClientUserInterface
BallClientUserInterface, 38
~Config
Config, 49
~FuncList
FuncList, 100
~Gui
Gui, 103
~Kicker
Kicker, 110
~KnowledgeBase
KnowledgeBase, 122
~Master
Master, 131
~Observer
Observer, 137
~Publish
Publish, 142
~SpsClient
SpsClient, 151
~SpsClientInterface
SpsClientInterface, 156
~SpsClientUserInterface
SpsClientUserInterface, 160
~UDPsocket
UDPsocket, 181

ANGLE
defines.h, 218
AXIS_ALL
StateDefines.h, 241
AXIS_BLOCKING
StateDefines.h, 241
AXIS_FOUR
defines.h, 218
AXIS_MOVING
StateDefines.h, 241
AXIS_ONE
defines.h, 218
AXIS_READY
StateDefines.h, 241
AXIS_SHOOTING
StateDefines.h, 241
AXIS_THREE
defines.h, 218
AXIS_TWO
defines.h, 218
AXIS_WOBBLING
StateDefines.h, 241
aBallPos
Ball, 25
aaStates
KnowledgeBase, 129
aafComputerSendPosRot
Kicker, 118
aafComputerSendPosTrans
Kicker, 118
advanceShoot
AxisAgent, 12
afComputerAxisPosRot
Kicker, 118
afComputerAxisPosTrans
Kicker, 118
afHumanAxisPosRot
Kicker, 118
afHumanAxisPosTrans
Kicker, 118
afTargetPos
AxisAgent, 16
All
ConfigDefines.h, 216
allAxisSet
Kicker, 118
apAA
Master, 132
attach
FuncList, 100, 101
Publish, 142
ausScore
Kicker, 119
Axis
defines.h, 218
axis
AxisAgent, 16
FuncList, 102
AxisDistance
ConfigDefines.h, 215
AxisOffset
ConfigDefines.h, 215
AxisPosX
ConfigDefines.h, 215

AxisShift
ConfigDefines.h, 215
AxisAgent, 9
~AxisAgent, 11, 12
advanceShoot, 12
afTargetPos, 16
axis, 16
AxisAgent, 11
AxisAgent, 11
firstFunc, 16
getRunAgentInstance, 12
m_vecRotTargetPos, 16
mutexHandle, 17
myStopped, 17
myThreadHandle, 17
pConf, 17
pKB, 18
pfComputerTargetPos, 17
runAgent, 13
shoot, 14
start, 15
stop, 16
AxisAgent.cpp, 189
AxisAgent.h, 189
funcElement, 191
moveFunc, 191
AxisAgent_wobbeln.cpp, 191
AxisAgent_wobbeln.h, 191
funcElement, 192
moveFunc, 192
AxisKinetics.h, 192
calcPosWhereBallCrossesAxis, 194
checkBallShootable, 195
checkMaxRotVelocity, 196
checkMaxTransVelocity, 197
checkPosReached, 198
chooseFigure, 199
findAngleFromBallToGoal, 200
moveAxisPTP, 201
moveAxisTo, 202
moveFeetInCircularMotion, 203
prepareShoot, 204
shootBall, 205
AxisStates
StateDefines.h, 241
axisUpdate
KnowledgeBase, 123

BALL_ALL
StateDefines.h, 242
BALL_READY
StateDefines.h, 242
BALLPOS_ALL
StateDefines.h, 242
BALLPOS_COMPUTER_POSSESSION
StateDefines.h, 242
BALLPOS_HUMAN_POSSESSION
StateDefines.h, 242
BALLPOS_SPEED
StateDefines.h, 242
BALLPOSTOAXIS_ALL
StateDefines.h, 242
BALLPOSTOAXIS_HANDABLE
StateDefines.h, 242
BALLPOSTOAXIS_HOLDABLE
StateDefines.h, 242
BALLPOSTOAXIS_PASSABLE
StateDefines.h, 242
BALLPOSTOAXIS_SHOOTABLE
StateDefines.h, 242
bCoordinateOk
ballCoordinate_t, 39
BUFFER_SIZE
defines.h, 217
Ball, 18
~Ball, 21
aBallPos, 25
Ball, 21
client, 25
getPos, 22
getPosAndVelo, 22
getVelo, 23
mutexHandle, 25
pConf, 25
pKB, 26
setBallPos, 24
start, 24
usBallPosIterator, 26
Ball.cpp, 206
Ball.h, 207
BallDiameter
ConfigDefines.h, 216
BallHost
ConfigDefines.h, 215
BallPort
ConfigDefines.h, 215
BallToAxisReachable
ConfigDefines.h, 216
BallToAxisShootable
ConfigDefines.h, 216
BallClient, 26
~BallClient, 30
BallClient, 29
BallClient, 29
myRecvData, 34
myServerAddr, 34
mySocket, 34
myUser, 34
requestBallPos, 30
start, 31
subscribeBallPos, 31
update, 32, 33
BallClient.cpp, 208
BallClient.h, 209
BallClientInterface, 34
~BallClientInterface, 35
requestBallPos, 36

start, 36
subscribeBallPos, 36
BallClientInterface.h, 210
BallClientUserInterface, 37
~BallClientUserInterface, 38
setBallPos, 38
start, 38
BallClientUserInterface.h, 211
ballCoordinate_t, 39
bCoordinateOk, 39
timestamp, 39
UdpDefines.h, 248
ulFrameCnt, 39
x, 39
y, 40
ballPacket, 40
ballPos, 41
id, 41
UdpDefines.h, 248
ballPos
ballPacket, 41
BallPosStates
StateDefines.h, 241
BallPosToAxisStates
StateDefines.h, 242
ballSqueeze_05
ubf16_request::bitfeld_16, 42
BallStates
StateDefines.h, 242
ballUpdate
KnowledgeBase, 123
bl
bool_1, 43
bool_2, 44
blocking
funcElement, 98
bool_1, 43
bl, 43
id, 43
UdpDefines.h, 248
bool_2, 44
bl, 44
id, 44
UdpDefines.h, 248
bothAxisPos_02
ubf16_request::bitfeld_16, 42

COMPUTER
defines.h, 219
CONFIG_ALL
StateDefines.h, 242
CONFIG_READY
StateDefines.h, 242
COORDINATE_X
defines.h, 218
COORDINATE_Y
defines.h, 218
COORDINATE_Z
defines.h, 218
calcPosWhereBallCrossesAxis
AxisKinetics.h, 194
checkBallShootable
AxisKinetics.h, 195
checkMaxRotVelocity
AxisKinetics.h, 196
checkMaxTransVelocity
AxisKinetics.h, 197
checkPosReached
AxisKinetics.h, 198
chooseFigure
AxisKinetics.h, 199
client
Ball, 25
Kicker, 119
ClientInterface, 44
computerAxisPos_01
ubf16_request::bitfeld_16, 42
computerAxisRamp_03
ubf16_request::bitfeld_16, 42
computerAxisStackmode_04
ubf16_request::bitfeld_16, 42
Config, 45
~Config, 49
Config, 48
getAxisDistance, 49
getAxisOffset, 49
getAxisPosX, 50
getAxisShift, 51
getAxisVal, 51
getBallAddr, 52, 53
getBallDiameter, 54
getBallToAxisReachable, 54
getBallToAxisShootable, 55
getConfFromFile, 56
getConnectionInfo, 58
getDimension, 59
getFieldSize, 59
getFigureFieldDistance, 60
getFigureFootBreadth, 61
getFigureFootHeight, 61
getFigureFootThickness, 62
getFigurePos, 63
getFigureVal, 63
getGoalSize, 64
getGuiAddr, 64–66
getKB, 66
getKiBallAddr, 66, 67
getKiGuiAddr, 68, 69
getKiSPServerAddr, 70, 71
getSPServerAddr, 72, 73
getShootAngleInfo, 71
getState, 73, 74
getStateInfo, 74
getStatesInGroup, 75
getVar, 75–79
getVelocityMaxRot, 79
getVelocityMaxTrans, 80

pKB, 83
pProfileHandle, 83
pVar, 83
profileDoc, 83
setState, 80
start, 81
writeConfToFile, 83
Config.cpp, 212
Config.h, 213
ConfigDefines.h
All, 216
AxisDistance, 215
AxisOffset, 215
AxisPosX, 215
AxisShift, 215
BallDiameter, 216
BallHost, 215
BallPort, 215
BallToAxisReachable, 216
BallToAxisShootable, 216
FieldSize, 216
FigureFieldDistance, 216
FigureFootBreadth, 216
FigureFootHeight, 216
FigureFootThickness, 216
FigurePos, 216
GoalSize, 216
GuiHost, 215
GuiPort, 215
KiBallHost, 215
KiBallPort, 215
KiGuiHost, 215
KiGuiPort, 215
KiSPServerHost, 215
KiSPServerPort, 215
NumShootAnglesByLin, 216
NumShootAnglesByRot, 216
SPServerHost, 215
SPServerPort, 215
ShootAnglesByLin, 216
ShootAnglesByRot, 216
States, 216
StatesPerGroup, 216
VelocityMaxRot, 216
VelocityMaxTrans, 216
ConfigDefines.h, 214
ConfigVar, 215
ConfigException, 83
ConfigException, 86
ConfigException, 86
getAttrName, 86
getValName, 86
myAttr, 87
myVal, 87
ConfigStates
StateDefines.h, 242
ConfigVar
ConfigDefines.h, 215
Coordinate
defines.h, 218

DWORD
Exception.h, 224
data
ubf16_request, 176
Data_Type
defines.h, 218
Data_Value
defines.h, 218
dec
State, 165
defines.h
ANGLE, 218
AXIS_FOUR, 218
AXIS_ONE, 218
AXIS_THREE, 218
AXIS_TWO, 218
COMPUTER, 219
COORDINATE_X, 218
COORDINATE_Y, 218
COORDINATE_Z, 218
FIGURE_FIVE, 219
FIGURE_FOUR, 219
FIGURE_ONE, 219
FIGURE_THREE, 219
FIGURE_TWO, 219
GRID, 219
HUMAN, 219
POSITION, 218
SHOOT_ALL, 219
SHOOT_ANGLE, 219
SHOOT_OFFSET, 219
STOP, 219
VALUE, 218
VELOCITY, 218
defines.h, 216
Axis, 218
BUFFER_SIZE, 217
Coordinate, 218
Data_Type, 218
Data_Value, 218
FIELDSIZE_X, 219
FIELDSIZE_Z, 219
Figure, 218
GOAL_WIDTH, 220
Gamer, 219
MAX_SHOOT_OFFSETS, 217
PI, 220
Safety, 219
ShootOffset, 219
deleteStates
KnowledgeBase, 124
detach
Publish, 143
diffToMax
State, 165
diffToMin

State, 165

ERROR_AXIS_BLOCK
ErrorCodes.h, 221
ERROR_BALL
ErrorCodes.h, 221
ERROR_CONFIG
ErrorCodes.h, 221
ERROR_FILE
ErrorCodes.h, 221
ERROR_KICKER
ErrorCodes.h, 221
ERROR_KNOWLEDGEBASE
ErrorCodes.h, 221
ERROR_MUTEX
ErrorCodes.h, 222
ERROR_MUTEX_WAIT
ErrorCodes.h, 222
ERROR_NONE
ErrorCodes.h, 222
ERROR_STATE
ErrorCodes.h, 222
ERROR_THREAD
ErrorCodes.h, 222
ERROR_UDP_NO_HOST
ErrorCodes.h, 222
ERROR_UNKNOWN
ErrorCodes.h, 222
ErrorCodes.h, 220
ERROR_AXIS_BLOCK, 221
ERROR_BALL, 221
ERROR_CONFIG, 221
ERROR_FILE, 221
ERROR_KICKER, 221
ERROR_KNOWLEDGEBASE, 221
ERROR_MUTEX, 222
ERROR_MUTEX_WAIT, 222
ERROR_NONE, 222
ERROR_STATE, 222
ERROR_THREAD, 222
ERROR_UDP_NO_HOST, 222
ERROR_UNKNOWN, 222
Exception, 87
Exception, 88
getCode, 89
getDesc, 89
getErrStr, 90
getExDesc, 90
myCode, 91
myDesc, 92
operator=, 91
setDesc, 91
Exception.h, 222
DWORD, 224
execFunc
FuncList, 101

FIGURE_FIVE
defines.h, 219
FIGURE_FOUR
defines.h, 219
FIGURE_ONE
defines.h, 219
FIGURE_THREE
defines.h, 219
FIGURE_TWO
defines.h, 219
FIELDSIZE_X
defines.h, 219
FIELDSIZE_Z
defines.h, 219
FieldSize
ConfigDefines.h, 216
Figure
defines.h, 218
FigureFieldDistance
ConfigDefines.h, 216
FigureFootBreadth
ConfigDefines.h, 216
FigureFootHeight
ConfigDefines.h, 216
FigureFootThickness
ConfigDefines.h, 216
FigurePos
ConfigDefines.h, 216
FileException, 92
FileException, 93
FileException, 93
getFileName, 94
myFile, 94
findAngleFromBallToGoal
AxisKinetics.h, 200
firstFunc
AxisAgent, 16
FuncList, 102
firstObs
Publish, 145
float_1, 94
flt, 95
id, 95
UdpDefines.h, 248
float_2_2_4, 95
flt, 95
id, 95
UdpDefines.h, 248
float_2_4, 96
flt, 96
id, 96
UdpDefines.h, 248
float_4, 96
flt, 97
id, 97
UdpDefines.h, 249
flt
float_1, 95
float_2_2_4, 95
float_2_4, 96

float_4, 97
func
funcElement, 98
funcElement, 97
AxisAgent.h, 191
AxisAgent_wobbeln.h, 192
blocking, 98
func, 98
next, 98
timeout, 98
values, 98
FuncList, 98
~FuncList, 100
attach, 100, 101
axis, 102
execFunc, 101
firstFunc, 102
FuncList, 100
FuncList, 100
lastFunc, 102
pKB, 102

GAME_ALL
StateDefines.h, 243
GAME_COMPUTERSCORE
StateDefines.h, 243
GAME_HUMANSCORE
StateDefines.h, 243
GAME_RUNNING
StateDefines.h, 243
GET_AXISRAMP
UdpDefines.h, 249
GET_BALLPOS
UdpDefines.h, 250
GET_BALLSQUEEZE
UdpDefines.h, 249
GET_BOTH_AXISPOS
UdpDefines.h, 249
GET_COMPUTER_AXISPOS
UdpDefines.h, 249
GET_HUMAN_AXISPOS
UdpDefines.h, 249
GET_SAFETY
UdpDefines.h, 250
GET_SCORE
UdpDefines.h, 250
GET_STACKMODE
UdpDefines.h, 249
GLOBAL_ALL
StateDefines.h, 243
GLOBAL_READY
StateDefines.h, 243
GRID
defines.h, 219
GOAL_WIDTH
defines.h, 220
GameStates
StateDefines.h, 242
Gamer
defines.h, 219
get
State, 165
getAddrFromString
UDPsocket, 181
getAttrName
ConfigException, 86
getAxisDistance
Config, 49
getAxisOffset
Config, 49
getAxisPosX
Config, 50
getAxisShift
Config, 51
getAxisVal
Config, 51
getBall
KnowledgeBase, 124
getBallAddr
Config, 52, 53
getBallDiameter
Config, 54
getBallToAxisReachable
Config, 54
getBallToAxisShootable
Config, 55
getCode
Exception, 89
getComputerAxisPos
Kicker, 110, 111
getConf
KnowledgeBase, 125
getConfFromFile
Config, 56
getConnectionInfo
Config, 58
getData
UDPsocket, 181
getDesc
Exception, 89
getDimension
Config, 59
getErrStr
Exception, 90
getExDesc
Exception, 90
getFieldSize
Config, 59
getFigureFieldDistance
Config, 60
getFigureFootBreadth
Config, 61
getFigureFootHeight
Config, 61
getFigureFootThickness
Config, 62
getFigurePos

Config, 63
getFigureVal
Config, 63
getFileName
FileException, 94
getGoalSize
Config, 64
getGuiAddr
Config, 64–66
getHostName
HostException, 106
getHumanAxisPos
Kicker, 111, 112
getKB
Config, 66
getKiBallAddr
Config, 66, 67
getKiGuiAddr
Config, 68, 69
getKiSPServerAddr
Config, 70, 71
getKicker
KnowledgeBase, 125
getMax
State, 166
getMin
State, 166
getPos
Ball, 22
getPosAndVelo
Ball, 22
getRecvDataInstance
UDPsocket, 182
getRunAgentInstance
AxisAgent, 12
getRunKBInstance
KnowledgeBase, 126
getSPServerAddr
Config, 72, 73
getScore
Kicker, 112
getShootAngleInfo
Config, 71
getState
Config, 73, 74
StateException, 172
getStateCategory
StateException, 172
getStateInfo
Config, 74
getStatesInGroup
Config, 75
getThreadClassName
ThreadException, 175
getValName
ConfigException, 86
getVar
Config, 75–79
getVelo
Ball, 23
getVelocityMaxRot
Config, 79
getVelocityMaxTrans
Config, 80
getWinError
MutexException, 135
GlobalStates
StateDefines.h, 243
GoalSize
ConfigDefines.h, 216
Gui, 102
∼Gui, 103
Gui, 103
Gui.cpp, 224
Gui.h, 225
GuiHost
ConfigDefines.h, 215
GuiPort
ConfigDefines.h, 215

HUMAN
defines.h, 219
HostException, 103
getHostName, 106
HostException, 105
HostException, 105
myHost, 106
humanAxisPos_00
ubf16_request::bitfeld_16, 42

iGroup
State, 168
iState
State, 168
iStateMax
State, 168
iStateMin
State, 168
id
ballPacket, 41
bool_1, 43
bool_2, 44
float_1, 95
float_2_2_4, 95
float_2_4, 96
float_4, 97
requestPacket, 146
unsignedShort_2, 188
inc
State, 166
initStates
KnowledgeBase, 126

KICKER_ALL
StateDefines.h, 243
KICKER_READY
StateDefines.h, 243

KNOWLEDGEBASE_ALL
StateDefines.h, 243
KNOWLEDGEBASE_READY
StateDefines.h, 243
KiBallHost
ConfigDefines.h, 215
KiBallPort
ConfigDefines.h, 215
KiGuiHost
ConfigDefines.h, 215
KiGuiPort
ConfigDefines.h, 215
KiSPServerHost
ConfigDefines.h, 215
KiSPServerPort
ConfigDefines.h, 215
Kicker, 106
~Kicker, 110
aafComputerSendPosRot, 118
aafComputerSendPosTrans, 118
afComputerAxisPosRot, 118
afComputerAxisPosTrans, 118
afHumanAxisPosRot, 118
afHumanAxisPosTrans, 118
allAxisSet, 118
ausScore, 119
client, 119
getComputerAxisPos, 110, 111
getHumanAxisPos, 111, 112
getScore, 112
Kicker, 110
lastSend, 119
mutexHandle, 119
pConf, 119
pKB, 119
requestAxisPos, 112
requestScore, 113
sendAxisPos, 113
sendScore, 114
setAxisPos, 114
setComputerAxisPos, 115
setHumanAxisPos, 116
setScore, 116
start, 116
subscribeScore, 117
Kicker.cpp, 226
Kicker.h, 226
KickerStates
StateDefines.h, 243
KnowledgeBase, 120
~KnowledgeBase, 122
aaStates, 129
axisUpdate, 123
ballUpdate, 123
deleteStates, 124
getBall, 124
getConf, 125
getKicker, 125
getRunKBInstance, 126
initStates, 126
KnowledgeBase, 122
KnowledgeBase, 122
linkStates, 127
myStopped, 129
myThreadHandle, 129
pBall, 129
pConf, 129
pKicker, 130
runKB, 127
start, 128
unlinkStates, 128
KnowledgeBase.cpp, 228
KnowledgeBase.h, 228
KnowledgeBaseStates
StateDefines.h, 243
KnowlegeBase, 130

lastFunc
FuncList, 102
lastObs
Publish, 145
lastSend
Kicker, 119
linkStates
KnowledgeBase, 127

m_vecRotTargetPos
AxisAgent, 16
MAX_SHOOT_OFFSETS
defines.h, 217
MAX_VELOCITY_ROT
UdpDefines.h, 250
MAX_VELOCITY_TRANS
UdpDefines.h, 250
main
main.cpp, 230
main.cpp, 230
main, 230
Master, 130
~Master, 131
apAA, 132
Master, 131
pKB, 133
run, 132
Master.cpp, 231
Master.h, 232
moveAxisPTP
AxisKinetics.h, 201
moveAxisTo
AxisKinetics.h, 202
moveFeetInCircularMotion
AxisKinetics.h, 203
moveFunc
AxisAgent.h, 191
AxisAgent_wobbeln.h, 192
MutexException, 133
getWinError, 135

MutexException, 134
MutexException, 134
myWinErr, 135
mutexHandle
AxisAgent, 17
Ball, 25
Kicker, 119
State, 168
myAddr
UDPsocket, 185
myAttr
ConfigException, 87
myBuffer
UDPsocket, 185
myBufferSize
UDPsocket, 185
myCode
Exception, 91
myDataSize
UDPsocket, 186
myDesc
Exception, 92
myFile
FileException, 94
myHost
HostException, 106
myPaused
UDPsocket, 186
myRecvData
BallClient, 34
SpsClient, 153
mySenderAddr
UDPsocket, 186
myServerAddr
BallClient, 34
SpsClient, 153
mySock
UDPsocket, 186
mySocket
BallClient, 34
SpsClient, 154
myState
StateException, 172
myStateCategory
StateException, 172
myStopped
AxisAgent, 17
KnowledgeBase, 129
UDPsocket, 186
myThreadClass
ThreadException, 175
myThreadHandle
AxisAgent, 17
KnowledgeBase, 129
UDPsocket, 187
myUser
BallClient, 34
SpsClient, 154
myVal
ConfigException, 87
myWinErr
MutexException, 135
myWsa
UDPsocket, 187
myWsaErr
UDPsocket, 187

NULL
publish.h, 235
next
funcElement, 98
ObsListKnot, 139
notify
Publish, 143, 144
NumShootAnglesByLin
ConfigDefines.h, 216
NumShootAnglesByRot
ConfigDefines.h, 216

obs
ObsListKnot, 139
ObsListKnot, 138
next, 139
obs, 139
prev, 139
ObsListKnot_t
publish.h, 235
Observer, 135
~Observer, 137
Observer, 137
update, 137, 138
observer.h, 233
operator=
Exception, 91

POSITION
defines.h, 218
pBall
KnowledgeBase, 129
pConf
AxisAgent, 17
Ball, 25
Kicker, 119
KnowledgeBase, 129
State, 169
PI
defines.h, 220
pKB
AxisAgent, 18
Ball, 26
Config, 83
FuncList, 102
Kicker, 119
Master, 133
State, 169
pKicker
KnowledgeBase, 130

pProfileHandle
Config, 83
pVar
Config, 83
PacketID
UdpDefines.h, 249
pause
UDPsocket, 183
pfComputerTargetPos
AxisAgent, 17
prepareShoot
AxisKinetics.h, 204
prev
ObsListKnot, 139
profileDoc
Config, 83
Publish, 139
~Publish, 142
attach, 142
detach, 143
firstObs, 145
lastObs, 145
notify, 143, 144
Publish, 142
publish.h, 234
NULL, 235
ObsListKnot_t, 235

REQUEST
UdpDefines.h, 249
REQUEST_BALLPOS
UdpDefines.h, 250
recvData
UDPsocket, 183
request
requestPacket, 147
requestAxisPos
Kicker, 112
SpsClient, 151
SpsClientInterface, 156
requestBallPos
BallClient, 30
BallClientInterface, 36
requestPacket, 145
id, 146
request, 147
UdpDefines.h, 249
requestScore
Kicker, 113
SpsClient, 151
SpsClientInterface, 156
reserved_08
ubf16_request::bitfeld_16, 42
reserved_09
ubf16_request::bitfeld_16, 42
reserved_10
ubf16_request::bitfeld_16, 42
reserved_11
ubf16_request::bitfeld_16, 42
reserved_12
ubf16_request::bitfeld_16, 42
reserved_13
ubf16_request::bitfeld_16, 42
reserved_14
ubf16_request::bitfeld_16, 43
reserved_15
ubf16_request::bitfeld_16, 43
run
Master, 132
runAgent
AxisAgent, 13
runKB
KnowledgeBase, 127

SET_AXISPOS
UdpDefines.h, 249
SET_AXISRAMP
UdpDefines.h, 249
SET_SCORE
UdpDefines.h, 249
SET_STACKMODE
UdpDefines.h, 249
SHOOT_ALL
defines.h, 219
SHOOT_ANGLE
defines.h, 219
SHOOT_OFFSET
defines.h, 219
SPServerHost
ConfigDefines.h, 215
SPServerPort
ConfigDefines.h, 215
STATE_MAX
StateDefines.h, 244
STATE_MIN
StateDefines.h, 244
STATE_VALUE
StateDefines.h, 244
STATES_ALL
StateDefines.h, 244
STATES_AXIS1COMPUTER
StateDefines.h, 244
STATES_AXIS1HUMAN
StateDefines.h, 244
STATES_AXIS2COMPUTER
StateDefines.h, 244
STATES_AXIS2HUMAN
StateDefines.h, 244
STATES_AXIS3COMPUTER
StateDefines.h, 244
STATES_AXIS3HUMAN
StateDefines.h, 244
STATES_AXIS4COMPUTER
StateDefines.h, 244
STATES_AXIS4HUMAN
StateDefines.h, 244
STATES_BALL
StateDefines.h, 244

STATES_BALLPOS
StateDefines.h, 244
STATES_BALLPOSTOAXIS1COMPUTER
StateDefines.h, 244
STATES_BALLPOSTOAXIS1HUMAN
StateDefines.h, 244
STATES_BALLPOSTOAXIS2COMPUTER
StateDefines.h, 244
STATES_BALLPOSTOAXIS2HUMAN
StateDefines.h, 244
STATES_BALLPOSTOAXIS3COMPUTER
StateDefines.h, 244
STATES_BALLPOSTOAXIS3HUMAN
StateDefines.h, 244
STATES_BALLPOSTOAXIS4COMPUTER
StateDefines.h, 244
STATES_BALLPOSTOAXIS4HUMAN
StateDefines.h, 244
STATES_CONFIG
StateDefines.h, 243
STATES_GAME
StateDefines.h, 243
STATES_GLOBAL
StateDefines.h, 243
STATES_KICKER
StateDefines.h, 244
STATES_KNOWLEDGEBASE
StateDefines.h, 244
STOP
defines.h, 219
SUBSCRIBE_BALLPOS
UdpDefines.h, 250
SUBSCRIBE_BALLSQUEEZE
UdpDefines.h, 249
SUBSCRIBE_SAFETY
UdpDefines.h, 249
SUBSCRIBE_SCORE
UdpDefines.h, 249
Safety
defines.h, 219
safety_07
ubf16_request::bitfeld_16, 43
score_06
ubf16_request::bitfeld_16, 43
sendAxisPos
Kicker, 113
SpsClient, 151
SpsClientInterface, 157
sendData
UDPsocket, 184
sendScore
Kicker, 114
SpsClient, 152
SpsClientInterface, 157
set
State, 166
setAxisPos
Kicker, 114
setBallPos
Ball, 24
BallClientUserInterface, 38
setComputerAxisPos
Kicker, 115
SpsClientUserInterface, 160
setDesc
Exception, 91
setHumanAxisPos
Kicker, 116
SpsClientUserInterface, 160
setScore
Kicker, 116
SpsClientUserInterface, 160
setState
Config, 80
shoot
AxisAgent, 14
ShootAnglesByLin
ConfigDefines.h, 216
ShootAnglesByRot
ConfigDefines.h, 216
shootBall
AxisKinetics.h, 205
ShootOffset
defines.h, 219
SpsClient, 147
~SpsClient, 151
myRecvData, 153
myServerAddr, 153
mySocket, 154
myUser, 154
requestAxisPos, 151
requestScore, 151
sendAxisPos, 151
sendScore, 152
SpsClient, 150
SpsClient, 150
start, 152
subscribeScore, 152
update, 152, 153
SpsClient.cpp, 236
SpsClient.h, 236
SpsClientInterface, 154
~SpsClientInterface, 156
requestAxisPos, 156
requestScore, 156
sendAxisPos, 157
sendScore, 157
start, 157
subscribeScore, 158
SpsClientInterface.h, 238
SpsClientUserInterface, 159
~SpsClientUserInterface, 160
setComputerAxisPos, 160
setHumanAxisPos, 160
setScore, 160
start, 161

SpsClientUserInterface.h, 238
start
AxisAgent, 15
Ball, 24
BallClient, 31
BallClientInterface, 36
BallClientUserInterface, 38
Config, 81
Kicker, 116
KnowledgeBase, 128
SpsClient, 152
SpsClientInterface, 157
SpsClientUserInterface, 161
UDPsocket, 184
State, 161
dec, 165
diffToMax, 165
diffToMin, 165
get, 165
getMax, 166
getMin, 166
iGroup, 168
iState, 168
iStateMax, 168
iStateMin, 168
inc, 166
mutexHandle, 168
pConf, 169
pKB, 169
set, 166
State, 164
update, 167
StateDefines.h
AXIS_ALL, 241
AXIS_BLOCKING, 241
AXIS_MOVING, 241
AXIS_READY, 241
AXIS_SHOOTING, 241
AXIS_WOBBLING, 241
BALL_ALL, 242
BALL_READY, 242
BALLPOS_ALL, 242
BALLPOS_COMPUTER_POSSESSION, 242
BALLPOS_HUMAN_POSSESSION, 242
BALLPOS_SPEED, 242
BALLPOSTOAXIS_ALL, 242
BALLPOSTOAXIS_HANDABLE, 242
BALLPOSTOAXIS_HOLDABLE, 242
BALLPOSTOAXIS_PASSABLE, 242
BALLPOSTOAXIS_SHOOTABLE, 242
CONFIG_ALL, 242
CONFIG_READY, 242
GAME_ALL, 243
GAME_COMPUTERSCORE, 243
GAME_HUMANSCORE, 243
GAME_RUNNING, 243
GLOBAL_ALL, 243
GLOBAL_READY, 243
KICKER_ALL, 243
KICKER_READY, 243
KNOWLEDGEBASE_ALL, 243
KNOWLEDGEBASE_READY, 243
STATE_MAX, 244
STATE_MIN, 244
STATE_VALUE, 244
STATES_ALL, 244
STATES_AXIS1COMPUTER, 244
STATES_AXIS1HUMAN, 244
STATES_AXIS2COMPUTER, 244
STATES_AXIS2HUMAN, 244
STATES_AXIS3COMPUTER, 244
STATES_AXIS3HUMAN, 244
STATES_AXIS4COMPUTER, 244
STATES_AXIS4HUMAN, 244
STATES_BALL, 244
STATES_BALLPOS, 244
STATES_BALLPOSTOAXIS1COMPUTER, 244
STATES_BALLPOSTOAXIS1HUMAN, 244
STATES_BALLPOSTOAXIS2COMPUTER, 244
STATES_BALLPOSTOAXIS2HUMAN, 244
STATES_BALLPOSTOAXIS3COMPUTER, 244
STATES_BALLPOSTOAXIS3HUMAN, 244
STATES_BALLPOSTOAXIS4COMPUTER, 244
STATES_BALLPOSTOAXIS4HUMAN, 244
STATES_CONFIG, 243
STATES_GAME, 243
STATES_GLOBAL, 243
STATES_KICKER, 244
STATES_KNOWLEDGEBASE, 244
StateDefines.h, 239
AxisStates, 241
BallPosStates, 241
BallPosToAxisStates, 242
BallStates, 242
ConfigStates, 242
GameStates, 242
GlobalStates, 243
KickerStates, 243
KnowledgeBaseStates, 243
StateIDs, 243
StateVals, 244
StateException, 169
getState, 172
getStateCategory, 172
myState, 172
myStateCategory, 172
StateException, 172
StateException, 172
StateIDs
StateDefines.h, 243
StateVals
StateDefines.h, 244
States
ConfigDefines.h, 216
States.h, 244
StatesPerGroup

ConfigDefines.h, 216
stop
AxisAgent, 16
subscribeBallPos
BallClient, 31
BallClientInterface, 36
subscribeScore
Kicker, 117
SpsClient, 152
SpsClientInterface, 158

tObsListKnot, 175
TRAVERSE_DISTANCE_1
UdpDefines.h, 250
TRAVERSE_DISTANCE_2
UdpDefines.h, 250
TRAVERSE_DISTANCE_3
UdpDefines.h, 250
TRAVERSE_DISTANCE_5
UdpDefines.h, 250
ThreadException, 173
getThreadClassName, 175
myThreadClass, 175
ThreadException, 174
ThreadException, 174
timeout
funcElement, 98
timestamp
ballCoordinate_t, 39

UDPsocket, 177
~UDPsocket, 181
getAddrFromString, 181
getData, 181
getRecvDataInstance, 182
myAddr, 185
myBuffer, 185
myBufferSize, 185
myDataSize, 186
myPaused, 186
mySenderAddr, 186
mySock, 186
myStopped, 186
myThreadHandle, 187
myWsa, 187
myWsaErr, 187
pause, 183
recvData, 183
sendData, 184
start, 184
UDPsocket, 181
UDPsocket, 181
ubf16_request, 175
data, 176
UdpDefines.h, 249
value, 176
ubf16_request::bitfeld_16, 41
ballSqueeze_05, 42
bothAxisPos_02, 42
computerAxisPos_01, 42
computerAxisRamp_03, 42
computerAxisStackmode_04, 42
humanAxisPos_00, 42
reserved_08, 42
reserved_09, 42
reserved_10, 42
reserved_11, 42
reserved_12, 42
reserved_13, 42
reserved_14, 43
reserved_15, 43
safety_07, 43
score_06, 43
UdpDefines.h
GET_AXISRAMP, 249
GET_BALLPOS, 250
GET_BALLSQUEEZE, 249
GET_BOTH_AXISPOS, 249
GET_COMPUTER_AXISPOS, 249
GET_HUMAN_AXISPOS, 249
GET_SAFETY, 250
GET_SCORE, 250
GET_STACKMODE, 249
REQUEST, 249
REQUEST_BALLPOS, 250
SET_AXISPOS, 249
SET_AXISRAMP, 249
SET_SCORE, 249
SET_STACKMODE, 249
SUBSCRIBE_BALLPOS, 250
SUBSCRIBE_BALLSQUEEZE, 249
SUBSCRIBE_SAFETY, 249
SUBSCRIBE_SCORE, 249
UdpDefines.h, 246
ballCoordinate_t, 248
ballPacket, 248
bool_1, 248
bool_2, 248
float_1, 248
float_2_2_4, 248
float_2_4, 248
float_4, 249
MAX_VELOCITY_ROT, 250
MAX_VELOCITY_TRANS, 250
PacketID, 249
requestPacket, 249
TRAVERSE_DISTANCE_1, 250
TRAVERSE_DISTANCE_2, 250
TRAVERSE_DISTANCE_3, 250
TRAVERSE_DISTANCE_5, 250
ubf16_request, 249
unsignedShort_2, 249
UdpSocket.cpp, 251
UdpSocket.h, 251
ulFrameCnt
ballCoordinate_t, 39
unlinkStates

KnowledgeBase, 128
unsignedShort_2, 187
id, 188
UdpDefines.h, 249
us, 188
update
BallClient, 32, 33
Observer, 137, 138
SpsClient, 152, 153
State, 167
us
unsignedShort_2, 188
usBallPosIterator
Ball, 26

VALUE
defines.h, 218
VELOCITY
defines.h, 218
value
ubf16_request, 176
values
funcElement, 98
VelocityMaxRot
ConfigDefines.h, 216
VelocityMaxTrans
ConfigDefines.h, 216

writeConfToFile
Config, 83

x
ballCoordinate_t, 39

y
ballCoordinate_t, 40