

API 文档

June 30, 2026

Contents

1	AuboStudio SDK	23
2	专题索引	33
2.1	专题	33
3	目录结构	35
3.1	目录	35
4	命名空间索引	49
4.1	命名空间列表	49
5	继承关系索引	51
5.1	类继承关系	51
6	类索引	55
6.1	类列表	55
7	文件索引	61
7.1	文件列表	61
8	专题文档	65
8.1	ApplicationApi(应用程序)	65
8.1.1	详细描述	66
8.1.2	函数说明	66
8.2	InstallationApi(安装)	67
8.2.1	详细描述	68
8.2.2	函数说明	68
8.3	ProgramApi(程序)	68
8.3.1	详细描述	69
8.3.2	函数说明	69
8.4	SystemApi(系统信息)	69
8.4.1	详细描述	71
8.4.2	函数说明	71
8.5	UserInterfaceApi(用户界面)	76
8.5.1	详细描述	77
8.5.2	函数说明	77
9	目录说明	79
9.1	include/aubo_caps 目录参考	79
9.2	include/aubo_caps/domain/program/nodes/builtin 目录参考	80
9.3	include/aubo_caps/contribution/driver/gripper/capability 目录参考	81
9.4	include/aubo_caps/contribution 目录参考	81
9.5	include/aubo_caps/domain/data 目录参考	82
9.6	include/aubo_caps/domain/device 目录参考	82
9.7	doc 目录参考	83
9.8	include/aubo_caps/domain 目录参考	83
9.9	include/aubo_caps/contribution/driver 目录参考	84
9.10	include/aubo_caps/domain/value/expression 目录参考	84

9.11	include/aubo_caps/domain/feature 目录参考	85
9.12	include/aubo_caps/domain/function 目录参考	85
9.13	include/aubo_caps/contribution/driver/gripper 目录参考	86
9.14	include 目录参考	86
9.15	include/aubo_caps/contribution/installation 目录参考	87
9.16	include/aubo_caps/domain/installation 目录参考	87
9.17	include/aubo_caps/domain/io 目录参考	88
9.18	include/aubo_caps/domain/program/nodes 目录参考	88
9.19	include/aubo_caps/domain/payload 目录参考	89
9.20	include/aubo_caps/contribution/program 目录参考	89
9.21	include/aubo_caps/domain/program 目录参考	90
9.22	include/aubo_caps/contribution/driver/screw_driver 目录参考	90
9.23	include/aubo_caps/domain/script 目录参考	91
9.24	include/aubo_caps/domain/program/structure 目录参考	91
9.25	include/aubo_caps/domain/tcp 目录参考	92
9.26	include/aubo_caps/domain/undoredo 目录参考	92
9.27	include/aubo_caps/domain/user_interface 目录参考	93
9.28	include/aubo_caps/domain/value 目录参考	93
9.29	include/aubo_caps/domain/variable 目录参考	94
9.30	include/aubo_caps/domain/waypoint 目录参考	94
10	命名空间文档	97
10.1	arcs 命名空间参考	97
10.2	arcs::aubo_scope 命名空间参考	97
10.2.1	类型定义说明	104
10.2.2	枚举类型说明	105
10.2.3	函数说明	105
10.3	arcs::aubo_sdk 命名空间参考	116
10.3.1	函数说明	116
11	类说明	117
11.1	arcs::aubo_scope::AnalogIo 类参考	117
11.1.1	详细描述	119
11.1.2	构造及析构函数说明	119
11.1.3	成员函数说明	120
11.1.4	友元及相关符号说明	120
11.1.5	类成员变量说明	121
11.2	arcs::aubo_scope::AnalogIoFilter 结构体参考	121
11.2.1	详细描述	121
11.2.2	构造及析构函数说明	121
11.2.3	成员函数说明	121
11.2.4	类成员变量说明	121
11.3	arcs::aubo_scope::ApplicationApi 类参考	122
11.3.1	详细描述	123
11.3.2	构造及析构函数说明	123
11.3.3	成员函数说明	124
11.3.4	友元及相关符号说明	125
11.3.5	类成员变量说明	125
11.4	arcs::aubo_scope::AssemblyNode 类参考	125
11.4.1	详细描述	127
11.4.2	成员枚举类型说明	127
11.4.3	构造及析构函数说明	127
11.4.4	成员函数说明	128
11.4.5	友元及相关符号说明	128
11.4.6	类成员变量说明	129
11.5	arcs::aubo_scope::AssignmentNode 类参考	129
11.5.1	详细描述	130
11.5.2	成员枚举类型说明	130
11.5.3	构造及析构函数说明	131

11.5.4	成员函数说明	132
11.5.5	友元及相关符号说明	133
11.5.6	类成员变量说明	133
11.6	arcs::aubo_scope::AuboCapInfo 类参考	133
11.6.1	详细描述	133
11.6.2	构造及析构函数说明	133
11.6.3	成员函数说明	133
11.7	arcs::aubo_scope::BaseFeature 类参考	134
11.7.1	详细描述	135
11.7.2	构造及析构函数说明	135
11.7.3	友元及相关符号说明	136
11.7.4	类成员变量说明	137
11.8	arcs::aubo_scope::BoolRegister 类参考	137
11.8.1	详细描述	138
11.8.2	构造及析构函数说明	138
11.9	arcs::aubo_scope::BreakNode 类参考	139
11.9.1	详细描述	140
11.9.2	构造及析构函数说明	140
11.9.3	友元及相关符号说明	141
11.9.4	类成员变量说明	141
11.10	arcs::aubo_scope::CallNode 类参考	141
11.10.1	详细描述	142
11.10.2	构造及析构函数说明	143
11.10.3	友元及相关符号说明	144
11.10.4	类成员变量说明	144
11.11	arcs::aubo_scope::CaseNode 类参考	144
11.11.1	详细描述	145
11.11.2	构造及析构函数说明	145
11.11.3	友元及相关符号说明	146
11.11.4	类成员变量说明	146
11.12	arcs::aubo_scope::CircleMoveNode 类参考	147
11.12.1	详细描述	148
11.12.2	构造及析构函数说明	148
11.12.3	友元及相关符号说明	149
11.12.4	类成员变量说明	149
11.13	arcs::aubo_scope::CommentNode 类参考	149
11.13.1	详细描述	150
11.13.2	构造及析构函数说明	151
11.13.3	成员函数说明	152
11.13.4	友元及相关符号说明	152
11.13.5	类成员变量说明	152
11.14	arcs::aubo_scope::ContributionProvider< T > 模板类参考	152
11.14.1	详细描述	153
11.14.2	构造及析构函数说明	153
11.14.3	成员函数说明	153
11.15	arcs::aubo_scope::ConveyorTrackingNode 类参考	153
11.15.1	详细描述	154
11.15.2	成员枚举类型说明	155
11.15.3	构造及析构函数说明	155
11.15.4	友元及相关符号说明	156
11.15.5	类成员变量说明	157
11.16	arcs::aubo_scope::DataModel 类参考	157
11.16.1	详细描述	159
11.16.2	构造及析构函数说明	159
11.16.3	成员函数说明	160
11.16.4	友元及相关符号说明	167
11.16.5	类成员变量说明	167
11.17	arcs::aubo_scope::DefaultCaseNode 类参考	167

11.17.1 详细描述	168
11.17.2 构造及析构函数说明	169
11.17.3 友元及相关符号说明	170
11.17.4 类成员变量说明	170
11.18arcs::aubo_scope::DefaultSingletonTraits< Type > 模板结构体参考	170
11.18.1 详细描述	170
11.19arcs::aubo_scope::Device 类参考	170
11.19.1 详细描述	171
11.19.2 构造及析构函数说明	171
11.19.3 成员函数说明	171
11.20arcs::aubo_scope::DeviceManager 类参考	172
11.20.1 详细描述	172
11.20.2 构造及析构函数说明	172
11.20.3 成员函数说明	172
11.21arcs::aubo_scope::DeviceRegistrationManager 类参考	173
11.21.1 详细描述	173
11.21.2 构造及析构函数说明	173
11.21.3 成员函数说明	173
11.22arcs::aubo_scope::DigitalIo 类参考	174
11.22.1 详细描述	175
11.22.2 构造及析构函数说明	175
11.22.3 成员函数说明	177
11.22.4 友元及相关符号说明	177
11.22.5 类成员变量说明	177
11.23arcs::aubo_scope::DigitalIoFilter 结构体参考	177
11.23.1 详细描述	178
11.23.2 构造及析构函数说明	178
11.23.3 成员函数说明	178
11.23.4 类成员变量说明	178
11.24arcs::aubo_scope::DirectionNode 类参考	178
11.24.1 详细描述	180
11.24.2 成员枚举类型说明	180
11.24.3 构造及析构函数说明	181
11.24.4 成员函数说明	182
11.24.5 友元及相关符号说明	184
11.24.6 类成员变量说明	184
11.25arcs::aubo_scope::DoubleRangeValidator 结构体参考	184
11.25.1 详细描述	185
11.25.2 构造及析构函数说明	185
11.25.3 成员函数说明	186
11.25.4 类成员变量说明	186
11.26arcs::aubo_scope::DoubleRegister 类参考	187
11.26.1 详细描述	189
11.26.2 构造及析构函数说明	189
11.27arcs::aubo_scope::DriveScrewNotOKParameters 类参考	189
11.27.1 详细描述	189
11.27.2 构造及析构函数说明	189
11.28arcs::aubo_scope::DriveScrewOKParameters 类参考	189
11.28.1 详细描述	190
11.28.2 构造及析构函数说明	190
11.29arcs::aubo_scope::ElseIfNode 类参考	190
11.29.1 详细描述	191
11.29.2 构造及析构函数说明	191
11.29.3 成员函数说明	192
11.29.4 友元及相关符号说明	193
11.29.5 类成员变量说明	193
11.30arcs::aubo_scope::ElseNode 类参考	193
11.30.1 详细描述	194

11.30.2 构造及析构函数说明	195
11.30.3 友元及相关符号说明	196
11.30.4 类成员变量说明	196
11.31arcs::aubo_scope::EtherNetIP 类参考	196
11.31.1 详细描述	198
11.31.2 构造及析构函数说明	198
11.31.3 成员函数说明	199
11.31.4 友元及相关符号说明	200
11.31.5 类成员变量说明	200
11.32arcs::aubo_scope::EtherNetIPFilter 结构体参考	200
11.32.1 详细描述	201
11.32.2 构造及析构函数说明	201
11.32.3 成员函数说明	201
11.32.4 类成员变量说明	201
11.33arcs::aubo_scope::Expression 类参考	201
11.33.1 详细描述	202
11.33.2 构造及析构函数说明	202
11.33.3 成员函数说明	203
11.33.4 友元及相关符号说明	204
11.33.5 类成员变量说明	204
11.34arcs::aubo_scope::Feature 类参考	204
11.34.1 详细描述	205
11.34.2 构造及析构函数说明	205
11.34.3 成员函数说明	206
11.34.4 友元及相关符号说明	207
11.34.5 类成员变量说明	207
11.35arcs::aubo_scope::FeatureContributionModel 类参考	207
11.35.1 详细描述	208
11.35.2 构造及析构函数说明	208
11.35.3 成员函数说明	209
11.35.4 友元及相关符号说明	211
11.35.5 类成员变量说明	211
11.36arcs::aubo_scope::FeatureDefinedFilter 结构体参考	211
11.36.1 详细描述	211
11.36.2 成员函数说明	212
11.37arcs::aubo_scope::FeatureJoggableFilter 结构体参考	212
11.37.1 详细描述	212
11.37.2 成员函数说明	212
11.38arcs::aubo_scope::FeatureLine 类参考	212
11.38.1 详细描述	214
11.38.2 构造及析构函数说明	214
11.38.3 友元及相关符号说明	215
11.38.4 类成员变量说明	215
11.39arcs::aubo_scope::FeatureModel 类参考	215
11.39.1 详细描述	216
11.39.2 构造及析构函数说明	216
11.39.3 成员函数说明	217
11.39.4 友元及相关符号说明	218
11.39.5 类成员变量说明	218
11.40arcs::aubo_scope::FeaturePlane 类参考	218
11.40.1 详细描述	220
11.40.2 构造及析构函数说明	220
11.40.3 友元及相关符号说明	221
11.40.4 类成员变量说明	221
11.41arcs::aubo_scope::FeaturePoint 类参考	221
11.41.1 详细描述	223
11.41.2 构造及析构函数说明	223
11.41.3 友元及相关符号说明	224

11.41.4 类成员变量说明	224
11.42arcs::aubo_scope::FeatureTool 类参考	224
11.42.1 详细描述	226
11.42.2 构造及析构函数说明	226
11.42.3 友元及相关符号说明	227
11.42.4 类成员变量说明	227
11.43arcs::aubo_scope::FeatureVariable 类参考	227
11.43.1 详细描述	229
11.43.2 构造及析构函数说明	230
11.43.3 友元及相关符号说明	231
11.43.4 类成员变量说明	231
11.44arcs::aubo_scope::FeatureVariableFilter 结构体参考	231
11.44.1 详细描述	231
11.44.2 成员函数说明	231
11.45arcs::aubo_scope::FeedScrewParameters 类参考	231
11.45.1 详细描述	232
11.45.2 构造及析构函数说明	232
11.46arcs::aubo_scope::FolderNode 类参考	232
11.46.1 详细描述	233
11.46.2 构造及析构函数说明	233
11.46.3 成员函数说明	234
11.46.4 友元及相关符号说明	235
11.46.5 类成员变量说明	235
11.47arcs::aubo_scope::ForceNode 类参考	235
11.47.1 详细描述	236
11.47.2 成员枚举类型说明	236
11.47.3 构造及析构函数说明	237
11.47.4 友元及相关符号说明	238
11.47.5 类成员变量说明	238
11.48arcs::aubo_scope::ForceOptionNode 类参考	238
11.48.1 详细描述	239
11.48.2 构造及析构函数说明	239
11.48.3 友元及相关符号说明	240
11.48.4 类成员变量说明	240
11.49arcs::aubo_scope::Function 类参考	241
11.49.1 详细描述	241
11.49.2 构造及析构函数说明	241
11.49.3 成员函数说明	242
11.49.4 友元及相关符号说明	243
11.49.5 类成员变量说明	243
11.50arcs::aubo_scope::FunctionModel 类参考	243
11.50.1 详细描述	244
11.50.2 构造及析构函数说明	244
11.50.3 成员函数说明	245
11.50.4 友元及相关符号说明	246
11.50.5 类成员变量说明	246
11.51arcs::aubo_scope::GlobalVariable 类参考	247
11.51.1 详细描述	248
11.51.2 构造及析构函数说明	248
11.51.3 友元及相关符号说明	249
11.51.4 类成员变量说明	249
11.52arcs::aubo_scope::GlobalVariableFilter 结构体参考	250
11.52.1 详细描述	250
11.52.2 成员函数说明	250
11.53arcs::aubo_scope::GripActionParameters 类参考	250
11.53.1 详细描述	250
11.53.2 构造及析构函数说明	250
11.53.3 成员函数说明	250

11.54arcs::aubo_scope::GripForceCapability 类参考	252
11.54.1 详细描述	252
11.54.2 构造及析构函数说明	252
11.54.3 成员函数说明	253
11.55arcs::aubo_scope::GripperApiProvider 类参考	253
11.55.1 详细描述	253
11.55.2 构造及析构函数说明	253
11.55.3 成员函数说明	254
11.56arcs::aubo_scope::GripperConfiguration 类参考	254
11.56.1 详细描述	254
11.56.2 构造及析构函数说明	254
11.56.3 成员函数说明	255
11.57arcs::aubo_scope::GripperContribution 类参考	260
11.57.1 详细描述	260
11.57.2 构造及析构函数说明	260
11.57.3 成员函数说明	261
11.58arcs::aubo_scope::GripperDevice 类参考	262
11.58.1 详细描述	263
11.58.2 构造及析构函数说明	264
11.59arcs::aubo_scope::GripperNode 类参考	264
11.59.1 详细描述	265
11.59.2 成员枚举类型说明	265
11.59.3 构造及析构函数说明	266
11.59.4 成员函数说明	267
11.59.5 友元及相关符号说明	268
11.59.6 类成员变量说明	268
11.60arcs::aubo_scope::GripVacuumCapability 类参考	268
11.60.1 详细描述	268
11.60.2 构造及析构函数说明	268
11.60.3 成员函数说明	268
11.61arcs::aubo_scope::HaltNode 类参考	269
11.61.1 详细描述	270
11.61.2 构造及析构函数说明	270
11.61.3 友元及相关符号说明	271
11.61.4 类成员变量说明	271
11.62arcs::aubo_scope::IfNode 类参考	272
11.62.1 详细描述	273
11.62.2 构造及析构函数说明	273
11.62.3 成员函数说明	274
11.62.4 友元及相关符号说明	274
11.62.5 类成员变量说明	275
11.63arcs::aubo_scope::InputIoFilter 结构体参考	275
11.63.1 详细描述	275
11.63.2 成员函数说明	275
11.64arcs::aubo_scope::InputValidator 结构体参考	275
11.64.1 详细描述	276
11.64.2 成员函数说明	276
11.65arcs::aubo_scope::InstallationApi 类参考	277
11.65.1 详细描述	278
11.65.2 构造及析构函数说明	278
11.65.3 友元及相关符号说明	279
11.65.4 类成员变量说明	280
11.66arcs::aubo_scope::InstallationApiProvider 类参考	280
11.66.1 详细描述	280
11.66.2 构造及析构函数说明	280
11.66.3 成员函数说明	281
11.66.4 友元及相关符号说明	282
11.66.5 类成员变量说明	282

11.67arcs::aubo_scope::InstallationContributionConfiguration 类参考	282
11.67.1 详细描述	282
11.67.2 构造及析构函数说明	282
11.68arcs::aubo_scope::InstallationCreationContext 类参考	283
11.68.1 详细描述	283
11.68.2 成员枚举类型说明	283
11.68.3 构造及析构函数说明	283
11.68.4 成员函数说明	284
11.68.5 友元及相关符号说明	285
11.68.6 类成员变量说明	285
11.69arcs::aubo_scope::InstallationNodeContribution 类参考	285
11.69.1 详细描述	285
11.69.2 构造及析构函数说明	285
11.69.3 成员函数说明	285
11.70arcs::aubo_scope::InstallationNodeService 类参考	286
11.70.1 详细描述	286
11.70.2 构造及析构函数说明	286
11.70.3 成员函数说明	286
11.71arcs::aubo_scope::InstallationNodeView 类参考	288
11.71.1 详细描述	288
11.71.2 构造及析构函数说明	288
11.71.3 成员函数说明	288
11.72arcs::aubo_scope::IntegerRangeValidator 结构体参考	289
11.72.1 详细描述	290
11.72.2 构造及析构函数说明	290
11.72.3 成员函数说明	290
11.72.4 类成员变量说明	291
11.73arcs::aubo_scope::IntegerRegister 类参考	291
11.73.1 详细描述	293
11.73.2 构造及析构函数说明	293
11.74arcs::aubo_scope::Io 类参考	293
11.74.1 详细描述	294
11.74.2 成员枚举类型说明	294
11.74.3 构造及析构函数说明	295
11.74.4 成员函数说明	296
11.74.5 友元及相关符号说明	298
11.74.6 类成员变量说明	298
11.75arcs::aubo_scope::IoConfigure 结构体参考	298
11.75.1 详细描述	299
11.75.2 类成员变量说明	299
11.76arcs::aubo_scope::IoModel 类参考	300
11.76.1 详细描述	300
11.76.2 构造及析构函数说明	300
11.76.3 成员函数说明	301
11.76.4 友元及相关符号说明	302
11.76.5 类成员变量说明	302
11.77arcs::aubo_scope::KeyboardManager 类参考	302
11.77.1 详细描述	304
11.77.2 构造及析构函数说明	304
11.77.3 成员函数说明	305
11.77.4 友元及相关符号说明	307
11.77.5 类成员变量说明	307
11.78arcs::aubo_scope::LoopNode 类参考	307
11.78.1 详细描述	309
11.78.2 成员枚举类型说明	309
11.78.3 构造及析构函数说明	309
11.78.4 成员函数说明	310
11.78.5 友元及相关符号说明	312

11.78.6 类成员变量说明	312
11.79arcs::aubo_scope::ModbusIo 类参考	312
11.79.1 详细描述	314
11.79.2 构造及析构函数说明	314
11.79.3 成员函数说明	315
11.79.4 友元及相关符号说明	316
11.79.5 类成员变量说明	317
11.80arcs::aubo_scope::ModbusIoFilter 结构体参考	317
11.80.1 详细描述	317
11.80.2 构造及析构函数说明	317
11.80.3 成员函数说明	317
11.80.4 类成员变量说明	317
11.81arcs::aubo_scope::MoveNode 类参考	318
11.81.1 详细描述	319
11.81.2 成员枚举类型说明	319
11.81.3 构造及析构函数说明	320
11.81.4 成员函数说明	321
11.81.5 友元及相关符号说明	324
11.81.6 类成员变量说明	324
11.82arcs::aubo_scope::OfflineTrackNode 类参考	325
11.82.1 详细描述	326
11.82.2 构造及析构函数说明	326
11.82.3 友元及相关符号说明	327
11.82.4 类成员变量说明	327
11.83arcs::aubo_scope::OutputIoFilter 结构体参考	327
11.83.1 详细描述	327
11.83.2 成员函数说明	327
11.84arcs::aubo_scope::PalletNode 类参考	328
11.84.1 详细描述	329
11.84.2 构造及析构函数说明	329
11.84.3 友元及相关符号说明	330
11.84.4 类成员变量说明	330
11.85arcs::aubo_scope::Payload 类参考	330
11.85.1 详细描述	331
11.85.2 成员枚举类型说明	331
11.85.3 构造及析构函数说明	332
11.85.4 成员函数说明	333
11.85.5 友元及相关符号说明	334
11.85.6 类成员变量说明	335
11.86arcs::aubo_scope::PayloadContributionModel 类参考	335
11.86.1 详细描述	335
11.86.2 构造及析构函数说明	336
11.86.3 成员函数说明	337
11.86.4 友元及相关符号说明	340
11.86.5 类成员变量说明	340
11.87arcs::aubo_scope::PayloadModel 类参考	341
11.87.1 详细描述	341
11.87.2 构造及析构函数说明	341
11.87.3 成员函数说明	342
11.87.4 友元及相关符号说明	343
11.87.5 类成员变量说明	343
11.88arcs::aubo_scope::PersistedVariable 类参考	343
11.88.1 详细描述	345
11.88.2 构造及析构函数说明	345
11.88.3 成员函数说明	346
11.88.4 友元及相关符号说明	346
11.88.5 类成员变量说明	347
11.89arcs::aubo_scope::PersistedVariableFilter 结构体参考	347

11.89.1 详细描述	347
11.89.2 成员函数说明	347
11.90arcs::aubo_scope::PopupManager 类参考	347
11.90.1 详细描述	348
11.90.2 构造及析构函数说明	348
11.90.3 成员函数说明	349
11.90.4 友元及相关符号说明	349
11.90.5 类成员变量说明	349
11.91arcs::aubo_scope::PopupNode 类参考	350
11.91.1 详细描述	351
11.91.2 成员枚举类型说明	351
11.91.3 构造及析构函数说明	351
11.91.4 成员函数说明	352
11.91.5 友元及相关符号说明	353
11.91.6 类成员变量说明	353
11.92arcs::aubo_scope::PrepareToStartScrewdriverParameters 类参考	353
11.92.1 详细描述	354
11.92.2 构造及析构函数说明	354
11.93arcs::aubo_scope::Profinet 类参考	354
11.93.1 详细描述	356
11.93.2 构造及析构函数说明	356
11.93.3 成员函数说明	357
11.93.4 友元及相关符号说明	358
11.93.5 类成员变量说明	358
11.94arcs::aubo_scope::ProfinetFilter 结构体参考	358
11.94.1 详细描述	359
11.94.2 构造及析构函数说明	359
11.94.3 成员函数说明	359
11.94.4 类成员变量说明	359
11.95arcs::aubo_scope::ProgramApi 类参考	359
11.95.1 详细描述	361
11.95.2 构造及析构函数说明	361
11.95.3 友元及相关符号说明	362
11.95.4 类成员变量说明	362
11.96arcs::aubo_scope::ProgramApiProvider 类参考	363
11.96.1 详细描述	363
11.96.2 构造及析构函数说明	363
11.96.3 成员函数说明	364
11.96.4 友元及相关符号说明	365
11.96.5 类成员变量说明	365
11.97arcs::aubo_scope::ProgramContributionConfiguration 类参考	365
11.97.1 详细描述	366
11.97.2 构造及析构函数说明	366
11.97.3 成员函数说明	367
11.97.4 友元及相关符号说明	370
11.97.5 类成员变量说明	370
11.98arcs::aubo_scope::ProgramCreationContext 类参考	370
11.98.1 详细描述	370
11.98.2 成员枚举类型说明	370
11.98.3 构造及析构函数说明	371
11.98.4 成员函数说明	372
11.98.5 友元及相关符号说明	372
11.98.6 类成员变量说明	372
11.99arcs::aubo_scope::ProgramHomeNode 类参考	372
11.99.1 详细描述	374
11.99.2 构造及析构函数说明	374
11.99.3 友元及相关符号说明	375
11.99.4 类成员变量说明	375

11.100	arcs::aubo_scope::ProgramModel 类参考	375
11.100.1	详细描述	376
11.100.2	构造及析构函数说明	376
11.100.3	成员函数说明	377
11.100.4	友元及相关符号说明	378
11.100.5	类成员变量说明	378
11.101	arcs::aubo_scope::ProgramNode 类参考	378
11.101.1	详细描述	380
11.101.2	构造及析构函数说明	380
11.101.3	成员函数说明	381
11.101.4	友元及相关符号说明	381
11.101.5	类成员变量说明	381
11.102	arcs::aubo_scope::ProgramNodeContribution 类参考	381
11.102.1	详细描述	382
11.102.2	构造及析构函数说明	382
11.102.3	成员函数说明	382
11.103	arcs::aubo_scope::ProgramNodeFactory 类参考	384
11.103.1	详细描述	385
11.103.2	构造及析构函数说明	385
11.103.3	成员函数说明	386
11.103.4	友元及相关符号说明	390
11.103.5	类成员变量说明	391
11.104	arcs::aubo_scope::ProgramNodeService 类参考	391
11.104.1	详细描述	391
11.104.2	构造及析构函数说明	391
11.104.3	成员函数说明	391
11.105	arcs::aubo_scope::ProgramNodeView 类参考	393
11.105.1	详细描述	393
11.105.2	构造及析构函数说明	393
11.105.3	成员函数说明	393
11.106	arcs::aubo_scope::ProgramSelectionParameters 类参考	393
11.106.1	详细描述	394
11.106.2	构造及析构函数说明	394
11.106.3	成员函数说明	394
11.107	arcs::aubo_scope::RecordTrackNode 类参考	395
11.107.1	详细描述	396
11.107.2	构造及析构函数说明	396
11.107.3	友元及相关符号说明	397
11.107.4	类成员变量说明	397
11.108	arcs::aubo_scope::Register 类参考	397
11.108.1	详细描述	399
11.108.2	构造及析构函数说明	399
11.109	arcs::aubo_scope::RegisteredCapabilities 类参考	399
11.109.1	详细描述	399
11.109.2	构造及析构函数说明	399
11.109.3	成员函数说明	399
11.110	arcs::aubo_scope::ReleaseActionParameters 类参考	401
11.110.1	详细描述	401
11.110.2	构造及析构函数说明	401
11.110.3	成员函数说明	401
11.111	arcs::aubo_scope::RenderInterface 类参考	402
11.111.1	详细描述	404
11.111.2	成员枚举类型说明	404
11.111.3	构造及析构函数说明	405
11.111.4	成员函数说明	406
11.111.5	友元及相关符号说明	416
11.111.6	类成员变量说明	416
11.112	arcs::aubo_scope::ScrewdriverApiProvider 类参考	416

11.112.详细描述	417
11.112.构造及析构函数说明	417
11.112.成员函数说明	417
11.113arcs::aubo_scope::ScrewdriverConfiguration 类参考	417
11.113.详细描述	418
11.113.构造及析构函数说明	418
11.113.成员函数说明	418
11.114arcs::aubo_scope::ScrewdriverContribution 类参考	423
11.114.详细描述	424
11.114.构造及析构函数说明	424
11.114.成员函数说明	424
11.115arcs::aubo_scope::ScrewdriverParameters 类参考	425
11.115.详细描述	426
11.115.构造及析构函数说明	426
11.115.成员函数说明	426
11.116arcs::aubo_scope::ScrewdriverProgram 类参考	427
11.116.详细描述	427
11.116.构造及析构函数说明	427
11.116.成员函数说明	427
11.117arcs::aubo_scope::ScrewdriverReadyParameters 类参考	427
11.117.详细描述	428
11.117.构造及析构函数说明	428
11.118arcs::aubo_scope::ScrewdrivingNode 类参考	428
11.118.详细描述	429
11.118.构造及析构函数说明	429
11.118.友元及相关符号说明	430
11.118.类成员变量说明	430
11.119arcs::aubo_scope::ScriptNode 类参考	431
11.119.详细描述	432
11.119.构造及析构函数说明	432
11.119.友元及相关符号说明	433
11.119.类成员变量说明	433
11.120arcs::aubo_scope::ScriptWriter 类参考	433
11.120.详细描述	435
11.120.构造及析构函数说明	435
11.120.成员函数说明	436
11.120.友元及相关符号说明	441
11.120.类成员变量说明	442
11.121arcs::aubo_scope::SeekNode 类参考	442
11.121.详细描述	443
11.121.构造及析构函数说明	443
11.121.友元及相关符号说明	444
11.121.类成员变量说明	444
11.122arcs::aubo_scope::SetNode 类参考	444
11.122.详细描述	446
11.122.成员枚举类型说明	446
11.122.构造及析构函数说明	447
11.122.成员函数说明	448
11.122.友元及相关符号说明	451
11.122.类成员变量说明	451
11.123arcs::aubo_scope::SetPayloadNode 类参考	451
11.123.详细描述	452
11.123.成员枚举类型说明	452
11.123.构造及析构函数说明	453
11.123.成员函数说明	454
11.123.友元及相关符号说明	454
11.123.类成员变量说明	455
11.124arcs::aubo_scope::SpeedCapability 类参考	455

11.124.详细描述	455
11.124.构造及析构函数说明	455
11.124.成员函数说明	455
11.125rcs::aubo_scope::StringLengthValidator 结构体参考	456
11.125.详细描述	457
11.125.构造及析构函数说明	457
11.125.成员函数说明	457
11.125.类成员变量说明	458
11.126rcs::aubo_scope::SubProgNode 类参考	458
11.126.详细描述	459
11.126.构造及析构函数说明	459
11.126.友元及相关符号说明	460
11.126.类成员变量说明	460
11.127rcs::aubo_scope::SwitchNode 类参考	461
11.127.详细描述	462
11.127.构造及析构函数说明	462
11.127.友元及相关符号说明	463
11.127.类成员变量说明	463
11.128rcs::aubo_scope::SystemApi 类参考	463
11.128.详细描述	465
11.128.构造及析构函数说明	465
11.128.友元及相关符号说明	466
11.128.类成员变量说明	466
11.129rcs::aubo_scope::TCP 类参考	467
11.129.详细描述	467
11.129.构造及析构函数说明	467
11.129.成员函数说明	468
11.129.友元及相关符号说明	469
11.129.类成员变量说明	469
11.130rcs::aubo_scope::TcpContributionModel 类参考	469
11.130.详细描述	470
11.130.构造及析构函数说明	470
11.130.成员函数说明	471
11.130.友元及相关符号说明	473
11.130.类成员变量说明	473
11.131rcs::aubo_scope::TcpModel 类参考	473
11.131.详细描述	474
11.131.构造及析构函数说明	474
11.131.成员函数说明	475
11.131.友元及相关符号说明	476
11.131.类成员变量说明	477
11.132rcs::aubo_scope::ThreadNode 类参考	477
11.132.详细描述	478
11.132.构造及析构函数说明	478
11.132.友元及相关符号说明	479
11.132.类成员变量说明	479
11.133rcs::aubo_scope::TimerNode 类参考	479
11.133.详细描述	481
11.133.构造及析构函数说明	481
11.133.友元及相关符号说明	482
11.133.类成员变量说明	482
11.134rcs::aubo_scope::TreeNode 类参考	482
11.134.详细描述	483
11.134.构造及析构函数说明	483
11.134.成员函数说明	484
11.134.友元及相关符号说明	488
11.134.类成员变量说明	488
11.135rcs::aubo_scope::UIntegerRangeValidator 结构体参考	488

11.135. 详细描述	489
11.135. 构造及析构函数说明	490
11.135. 成员函数说明	490
11.135. 类成员变量说明	490
11.136rcs::aubo_scope::UndoRedoManager 类参考	491
11.136. 详细描述	491
11.136. 构造及析构函数说明	491
11.136. 成员函数说明	492
11.136. 友元及相关符号说明	493
11.136. 类成员变量说明	493
11.137rcs::aubo_scope::UntilNode 类参考	493
11.137. 详细描述	495
11.137. 成员枚举类型说明	496
11.137. 构造及析构函数说明	498
11.137. 成员函数说明	499
11.137. 友元及相关符号说明	503
11.137. 类成员变量说明	503
11.138rcs::aubo_scope::UserInteraction 类参考	503
11.138. 详细描述	504
11.138. 构造及析构函数说明	504
11.138. 成员函数说明	505
11.138. 友元及相关符号说明	509
11.138. 类成员变量说明	510
11.139rcs::aubo_scope::UserInterfaceApi 类参考	510
11.139. 详细描述	510
11.139. 构造及析构函数说明	510
11.139. 友元及相关符号说明	511
11.139. 类成员变量说明	512
11.140rcs::aubo_scope::ValueFactory 类参考	512
11.140. 详细描述	512
11.140. 构造及析构函数说明	512
11.140. 成员函数说明	513
11.140. 友元及相关符号说明	514
11.140. 类成员变量说明	514
11.141rcs::aubo_scope::Variable 类参考	514
11.141. 详细描述	515
11.141. 成员枚举类型说明	515
11.141. 构造及析构函数说明	516
11.141. 成员函数说明	517
11.141. 友元及相关符号说明	517
11.141. 类成员变量说明	518
11.142rcs::aubo_scope::VariableFactory 类参考	518
11.142. 详细描述	518
11.142. 构造及析构函数说明	518
11.142. 成员函数说明	520
11.142. 友元及相关符号说明	521
11.142. 类成员变量说明	521
11.143rcs::aubo_scope::VariableModel 类参考	521
11.143. 详细描述	522
11.143. 构造及析构函数说明	522
11.143. 成员函数说明	523
11.143. 友元及相关符号说明	525
11.143. 类成员变量说明	525
11.144rcs::aubo_scope::ViewApiProvider 类参考	525
11.144. 详细描述	526
11.144. 构造及析构函数说明	526
11.144. 成员函数说明	527
11.144. 友元及相关符号说明	527

11.144.5.类成员变量说明	528
11.145rcs::aubo_scope::VisualLocalizationProgramNode 类参考	528
11.145.1.详细描述	529
11.145.2.构造及析构函数说明	529
11.145.3.友元及相关符号说明	530
11.145.4.类成员变量说明	530
11.146rcs::aubo_scope::WaitNode 类参考	531
11.146.1.详细描述	532
11.146.2.成员枚举类型说明	533
11.146.3.构造及析构函数说明	533
11.146.4.成员函数说明	534
11.146.5.友元及相关符号说明	538
11.146.6.类成员变量说明	539
11.147rcs::aubo_scope::Waypoint 类参考	539
11.147.1.详细描述	539
11.147.2.构造及析构函数说明	540
11.147.3.成员函数说明	541
11.147.4.友元及相关符号说明	542
11.147.5.类成员变量说明	542
11.148rcs::aubo_scope::WaypointModel 类参考	542
11.148.1.详细描述	542
11.148.2.构造及析构函数说明	543
11.148.3.成员函数说明	544
11.148.4.友元及相关符号说明	546
11.148.5.类成员变量说明	546
11.149rcs::aubo_scope::WaypointNode 类参考	546
11.149.1.详细描述	548
11.149.2.成员枚举类型说明	548
11.149.3.构造及析构函数说明	550
11.149.4.成员函数说明	551
11.149.5.友元及相关符号说明	556
11.149.6.类成员变量说明	556
11.150rcs::aubo_scope::WidthCapability 类参考	556
11.150.1.详细描述	557
11.150.2.构造及析构函数说明	557
11.150.3.成员函数说明	557
12 文件说明	559
12.1 doc/SDK_overview.md 文件参考	559
12.2 include/aubo_caps/class_forward.h 文件参考	559
12.2.1 宏定义说明	560
12.3 class_forward.h	561
12.4 include/aubo_caps/contribution/contribution_provider.h 文件参考	562
12.5 contribution_provider.h	563
12.6 include/aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h 文件参考	564
12.7 grip_force_capability.h	565
12.8 include/aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h 文件参考	566
12.9 grip_vacuum_capability.h	567
12.10include/aubo_caps/contribution/driver/gripper/capability/registered_cpabilities.h 文件参考	568
12.11registered_cpabilities.h	568
12.12include/aubo_caps/contribution/driver/gripper/capability/speed_capability.h 文件参考	570
12.13speed_capability.h	571
12.14include/aubo_caps/contribution/driver/gripper/capability/width_capability.h 文件参考	572
12.15width_capability.h	573
12.16include/aubo_caps/contribution/driver/gripper/grip_action_parameters.h 文件参考	574

12.17grip_action_parameters.h	575
12.18include/aubo_caps/contribution/driver/gripper/gripper_api_provider.h 文件参考	577
12.19gripper_api_provider.h	578
12.20include/aubo_caps/contribution/driver/gripper/gripper_configuration.h 文件参考	579
12.21gripper_configuration.h	580
12.22include/aubo_caps/contribution/driver/gripper/gripper_contribution.h 文件参考	583
12.23gripper_contribution.h	584
12.24include/aubo_caps/contribution/driver/gripper/release_action_parameters.h 文件参考	587
12.25release_action_parameters.h	588
12.26include/aubo_caps/contribution/driver/screw_driver/screw_driver_api_provider.h 文件参考	590
12.27screw_driver_api_provider.h	591
12.28include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h 文件参考	591
12.29screw_driver_configuration.h	593
12.30include/aubo_caps/contribution/driver/screw_driver/screw_driver_contribution.h 文件参考	599
12.31screw_driver_contribution.h	600
12.32include/aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h 文件参考	602
12.33screw_driver_parameters.h	604
12.34include/aubo_caps/contribution/installation/installation_api_provider.h 文件参考	605
12.35installation_api_provider.h	606
12.36include/aubo_caps/contribution/installation/installation_contribution_configuration.h 文件参考	608
12.37installation_contribution_configuration.h	609
12.38include/aubo_caps/contribution/installation/installation_creation_context.h 文件参考	609
12.39installation_creation_context.h	610
12.40include/aubo_caps/contribution/installation/installation_node_contribution.h 文件参考	611
12.41installation_node_contribution.h	612
12.42include/aubo_caps/contribution/installation/installation_node_service.h 文件参考	613
12.43installation_node_service.h	614
12.44include/aubo_caps/contribution/installation/installation_node_view.h 文件参考	616
12.44.1 类型定义说明	617
12.45installation_node_view.h	617
12.46include/aubo_caps/contribution/program/program_api_provider.h 文件参考	618
12.47program_api_provider.h	619
12.48include/aubo_caps/contribution/program/program_contribution_configuration.h 文件参考	620
12.49program_contribution_configuration.h	621
12.50include/aubo_caps/contribution/program/program_creation_context.h 文件参考	623
12.51program_creation_context.h	624
12.52include/aubo_caps/contribution/program/program_node_contribution.h 文件参考	624
12.53program_node_contribution.h	625
12.54include/aubo_caps/contribution/program/program_node_service.h 文件参考	627
12.55program_node_service.h	628
12.56include/aubo_caps/contribution/program/program_node_view.h 文件参考	630
12.57program_node_view.h	631
12.58include/aubo_caps/contribution/view_api_provider.h 文件参考	631
12.59view_api_provider.h	632
12.60include/aubo_caps/domain/application_api.h 文件参考	633
12.61application_api.h	634
12.62include/aubo_caps/domain/data/data_model.h 文件参考	637
12.63data_model.h	638
12.64include/aubo_caps/domain/device/device.h 文件参考	647
12.65device.h	648
12.66include/aubo_caps/domain/device/device_manager.h 文件参考	650
12.67device_manager.h	651

12.68	<code>include/aubo_caps/domain/device/device_registration_manager.h</code> 文件参考	652
12.69	<code>device_registration_manager.h</code>	653
12.70	<code>include/aubo_caps/domain/device/gripper_device.h</code> 文件参考	654
12.71	<code>gripper_device.h</code>	655
12.72	<code>include/aubo_caps/domain/extension_info.h</code> 文件参考	656
12.73	<code>extension_info.h</code>	658
12.74	<code>include/aubo_caps/domain/feature/base_feature.h</code> 文件参考	659
12.75	<code>base_feature.h</code>	659
12.76	<code>include/aubo_caps/domain/feature/feature.h</code> 文件参考	660
12.77	<code>feature.h</code>	661
12.78	<code>include/aubo_caps/domain/feature/feature_contribution_model.h</code> 文件参考	663
12.79	<code>feature_contribution_model.h</code>	664
12.80	<code>include/aubo_caps/domain/feature/feature_line.h</code> 文件参考	666
12.81	<code>feature_line.h</code>	667
12.82	<code>include/aubo_caps/domain/feature/feature_model.h</code> 文件参考	668
12.83	<code>feature_model.h</code>	669
12.84	<code>include/aubo_caps/domain/feature/feature_plane.h</code> 文件参考	671
12.85	<code>feature_plane.h</code>	671
12.86	<code>include/aubo_caps/domain/feature/feature_point.h</code> 文件参考	672
12.87	<code>feature_point.h</code>	673
12.88	<code>include/aubo_caps/domain/feature/feature_tool.h</code> 文件参考	674
12.89	<code>feature_tool.h</code>	674
12.90	<code>include/aubo_caps/domain/function/function.h</code> 文件参考	675
12.91	<code>function.h</code>	676
12.92	<code>include/aubo_caps/domain/function/function_model.h</code> 文件参考	678
12.93	<code>function_model.h</code>	679
12.94	<code>include/aubo_caps/domain/installation/installation_api.h</code> 文件参考	681
12.95	<code>installation_api.h</code>	682
12.96	<code>include/aubo_caps/domain/io/analog_io.h</code> 文件参考	684
12.97	<code>analog_io.h</code>	686
12.98	<code>include/aubo_caps/domain/io/boolean_register.h</code> 文件参考	687
12.99	<code>boolean_register.h</code>	689
12.100	<code>include/aubo_caps/domain/io/digital_io.h</code> 文件参考	689
12.101	<code>digital_io.h</code>	691
12.102	<code>include/aubo_caps/domain/io/double_register.h</code> 文件参考	692
12.103	<code>double_register.h</code>	694
12.104	<code>include/aubo_caps/domain/io/ethernet_ip.h</code> 文件参考	694
12.105	<code>ethernet_ip.h</code>	695
12.106	<code>include/aubo_caps/domain/io/integer_register.h</code> 文件参考	697
12.107	<code>integer_register.h</code>	697
12.108	<code>include/aubo_caps/domain/io/io.h</code> 文件参考	698
12.109	<code>io.h</code>	699
12.110	<code>include/aubo_caps/domain/io/io_model.h</code> 文件参考	701
12.111	<code>io_model.h</code>	702
12.112	<code>include/aubo_caps/domain/io/modbus_io.h</code> 文件参考	703
12.113	<code>modbus_io.h</code>	705
12.114	<code>include/aubo_caps/domain/io/profinet.h</code> 文件参考	707
12.115	<code>profinet.h</code>	707
12.116	<code>include/aubo_caps/domain/io/register.h</code> 文件参考	709
12.117	<code>register.h</code>	710
12.118	<code>include/aubo_caps/domain/payload/payload.h</code> 文件参考	711
12.119	<code>payload.h</code>	712
12.120	<code>include/aubo_caps/domain/payload/payload_contribution_model.h</code> 文件参考	715
12.121	<code>payload_contribution_model.h</code>	716
12.122	<code>include/aubo_caps/domain/payload/payload_model.h</code> 文件参考	721
12.123	<code>payload_model.h</code>	723
12.124	<code>include/aubo_caps/domain/program/nodes/builtin/assembly_node.h</code> 文件参考	724
12.125	<code>assembly_node.h</code>	725

12.126	<code>include/aubo_caps/domain/program/nodes/builtin/assignment_node.h</code> 文件参考	726
12.127	<code>assignment_node.h</code>	727
12.128	<code>include/aubo_caps/domain/program/nodes/builtin/break_node.h</code> 文件参考	729
12.129	<code>break_node.h</code>	730
12.130	<code>include/aubo_caps/domain/program/nodes/builtin/call_node.h</code> 文件参考	731
12.131	<code>call_node.h</code>	732
12.132	<code>include/aubo_caps/domain/program/nodes/builtin/case_node.h</code> 文件参考	733
12.133	<code>case_node.h</code>	734
12.134	<code>include/aubo_caps/domain/program/nodes/builtin/circle_move_node.h</code> 文件参考	735
12.135	<code>circle_move_node.h</code>	736
12.136	<code>include/aubo_caps/domain/program/nodes/builtin/comment_node.h</code> 文件参考	737
12.137	<code>comment_node.h</code>	738
12.138	<code>include/aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h</code> 文件参考	739
12.139	<code>conveyor_tracking_node.h</code>	740
12.140	<code>include/aubo_caps/domain/program/nodes/builtin/default_case_node.h</code> 文件参考	741
12.141	<code>default_case_node.h</code>	742
12.142	<code>include/aubo_caps/domain/program/nodes/builtin/direction_node.h</code> 文件参考	743
12.143	<code>direction_node.h</code>	744
12.144	<code>include/aubo_caps/domain/program/nodes/builtin/else_if_node.h</code> 文件参考	746
12.145	<code>else_if_node.h</code>	748
12.146	<code>include/aubo_caps/domain/program/nodes/builtin/else_node.h</code> 文件参考	749
12.147	<code>else_node.h</code>	750
12.148	<code>include/aubo_caps/domain/program/nodes/builtin/folder_node.h</code> 文件参考	751
12.149	<code>folder_node.h</code>	752
12.150	<code>include/aubo_caps/domain/program/nodes/builtin/force_node.h</code> 文件参考	753
12.151	<code>force_node.h</code>	754
12.152	<code>include/aubo_caps/domain/program/nodes/builtin/force_option_node.h</code> 文件参考	756
12.153	<code>force_option_node.h</code>	757
12.154	<code>include/aubo_caps/domain/program/nodes/builtin/gripper_node.h</code> 文件参考	758
12.155	<code>gripper_node.h</code>	759
12.156	<code>include/aubo_caps/domain/program/nodes/builtin/halt_node.h</code> 文件参考	762
12.157	<code>halt_node.h</code>	763
12.158	<code>include/aubo_caps/domain/program/nodes/builtin/if_node.h</code> 文件参考	764
12.159	<code>if_node.h</code>	766
12.160	<code>include/aubo_caps/domain/program/nodes/builtin/loop_node.h</code> 文件参考	766
12.161	<code>loop_node.h</code>	768
12.162	<code>include/aubo_caps/domain/program/nodes/builtin/move_node.h</code> 文件参考	770
12.163	<code>move_node.h</code>	771
12.164	<code>include/aubo_caps/domain/program/nodes/builtin/offline_track_node.h</code> 文件参考	775
12.165	<code>offline_track_node.h</code>	776
12.166	<code>include/aubo_caps/domain/program/nodes/builtin/pallet_node.h</code> 文件参考	777
12.167	<code>pallet_node.h</code>	778
12.168	<code>include/aubo_caps/domain/program/nodes/builtin/popup_node.h</code> 文件参考	779
12.169	<code>popup_node.h</code>	780
12.170	<code>include/aubo_caps/domain/program/nodes/builtin/program_home_node.h</code> 文件参考	781
12.171	<code>program_home_node.h</code>	783
12.172	<code>include/aubo_caps/domain/program/nodes/builtin/record_track_node.h</code> 文件参考	784
12.173	<code>record_track_node.h</code>	785
12.174	<code>include/aubo_caps/domain/program/nodes/builtin/screw_driving_node.h</code> 文件参考	786
12.175	<code>screw_driving_node.h</code>	787
12.176	<code>include/aubo_caps/domain/program/nodes/builtin/script_node.h</code> 文件参考	788
12.177	<code>script_node.h</code>	789
12.178	<code>include/aubo_caps/domain/program/nodes/builtin/seek_node.h</code> 文件参考	790
12.179	<code>seek_node.h</code>	791
12.180	<code>include/aubo_caps/domain/program/nodes/builtin/set_node.h</code> 文件参考	792
12.181	<code>set_node.h</code>	793
12.182	<code>include/aubo_caps/domain/program/nodes/builtin/set_payload_node.h</code> 文件参考	796
12.183	<code>set_payload_node.h</code>	797

12.181	include/aubo_caps/domain/program/nodes/builtin/sub_prog_node.h 文件参考	799
12.185	sub_prog_node.h	800
12.186	include/aubo_caps/domain/program/nodes/builtin/switch_node.h 文件参考	801
12.187	switch_node.h	802
12.188	include/aubo_caps/domain/program/nodes/builtin/thread_node.h 文件参考	803
12.189	thread_node.h	804
12.190	include/aubo_caps/domain/program/nodes/builtin/timer_node.h 文件参考	805
12.191	timer_node.h	806
12.192	include/aubo_caps/domain/program/nodes/builtin/until_node.h 文件参考	807
12.193	until_node.h	808
12.194	include/aubo_caps/domain/program/nodes/builtin/visual_localization_program_node.h 文件参考	813
12.195	visual_localization_program_node.h	814
12.196	include/aubo_caps/domain/program/nodes/builtin/wait_node.h 文件参考	815
12.197	wait_node.h	816
12.198	include/aubo_caps/domain/program/nodes/builtin/waypoint_node.h 文件参考	819
12.199	waypoint_node.h	821
12.200	include/aubo_caps/domain/program/nodes/program_node.h 文件参考	826
12.201	program_node.h	827
12.202	include/aubo_caps/domain/program/nodes/program_node_factory.h 文件参考	828
12.203	program_node_factory.h	830
12.204	include/aubo_caps/domain/program/program_api.h 文件参考	833
12.205	program_api.h	834
12.206	include/aubo_caps/domain/program/program_model.h 文件参考	835
12.207	program_model.h	836
12.208	include/aubo_caps/domain/program/structure/tree_node.h 文件参考	837
12.209	tree_node.h	838
12.210	include/aubo_caps/domain/script/script_writer.h 文件参考	842
12.211	script_writer.h	843
12.212	include/aubo_caps/domain/system_api.h 文件参考	846
12.213	system_api.h	847
12.214	include/aubo_caps/domain/tcp/tcp.h 文件参考	854
12.215	tcp.h	855
12.216	include/aubo_caps/domain/tcp/tcp_contribution_model.h 文件参考	857
12.217	tcp_contribution_model.h	858
12.218	include/aubo_caps/domain/tcp/tcp_model.h 文件参考	860
12.219	tcp_model.h	861
12.220	include/aubo_caps/domain/undoredo/undo_redo_manager.h 文件参考	863
12.221	undo_redo_manager.h	865
12.222	include/aubo_caps/domain/user_interface/input_validator.h 文件参考	866
12.223	input_validator.h	867
12.224	include/aubo_caps/domain/user_interface/keyboard_manager.h 文件参考	869
12.225	keyboard_manager.h	870
12.226	include/aubo_caps/domain/user_interface/popup_manager.h 文件参考	871
12.227	popup_manager.h	873
12.228	include/aubo_caps/domain/user_interface/render_interface.h 文件参考	873
12.229	render_interface.h	875
12.230	include/aubo_caps/domain/user_interface/user_interaction.h 文件参考	883
12.231	user_interaction.h	884
12.232	include/aubo_caps/domain/user_interface/user_interface_api.h 文件参考	887
12.233	user_interface_api.h	888
12.234	include/aubo_caps/domain/value/expression/expression.h 文件参考	889
12.235	expression.h	890
12.236	include/aubo_caps/domain/value/value_factory_provider.h 文件参考	891
12.237	value_factory_provider.h	892
12.238	include/aubo_caps/domain/variable/feature_variable.h 文件参考	893
12.239	feature_variable.h	894
12.240	include/aubo_caps/domain/variable/global_variable.h 文件参考	896

12.241	global_variable.h	897
12.242	include/aubo_caps/domain/variable/persisted_variable.h 文件参考	898
12.243	persisted_variable.h	899
12.244	include/aubo_caps/domain/variable/variable.h 文件参考	900
12.245	variable.h	901
12.246	include/aubo_caps/domain/variable/variable_factory.h 文件参考	903
12.247	variable_factory.h	904
12.248	include/aubo_caps/domain/variable/variable_model.h 文件参考	906
12.249	variable_model.h	907
12.250	include/aubo_caps/domain/waypoint/waypoint.h 文件参考	909
12.251	waypoint.h	909
12.252	include/aubo_caps/domain/waypoint/waypoint_model.h 文件参考	911
12.253	waypoint_model.h	912
	索引	915

Chapter 1

AuboStudio SDK

概述

AuboStudio SDK 提供了一整套完整的接口供开发插件使用，为用户提供更加丰富的功能。

目录

- 在示教器中集成插件的原理
- SDK 架构
- 使用示例
- 命名空间

在示教器中集成插件的原理

插件是一个软件包，作为 `aubo_scope` 的子进程运行。`aubo_scope` 将注册插件，并以各种方式与之交互，例如在 `aubo_scope` 启动的过程中以及基于用户与 `aubo_scope` 的交互过程中。

每个插件可以为 `aubo_scope` 贡献多个新功能，这种功能通常称为服务。这种服务一般有以下两类：
服务：

1. 提供安装节点服务。
2. 提供程序节点服务。

示例 (夹爪 `aubo_caps`):

1. 提供安装节点服务：例如配置夹爪的安装和连接。
2. 提供程序节点服务：例如一个程序节点可以打开夹爪，另一个程序节点可以关闭夹爪。

将插件集成到 `aubo_scope` 的原理：

集成过程图如下：

- 当 `aubo_scope` 启动时，它将查找所有已安装的 `aubo_caps`。
- 然后 `aubo_scope` 在每个 `aubo_caps` 调用激活器，激活器有两个重写的函数 `start()`、`stop()`，在启动时首先调用 `start()` 利用 `RegisterService` 注册服务；而在 `aubo_scope` 停止时调用 `stop()` 函数。
- 注册后该 `Service` 提供了配置节点基本信息的函数、创建该节点的用户交互界面（`view` 类）的函数以及创建每个节点的 `Contribution` 类的函数。
- 在 `contribution` 类中可以为该节点提供特殊功能的 `Api` 以及生成相应的 `Script` 代码。
- 通常 `Service` 中的设置可以认为是静态的设置，因为它们不可用于运动时，只在启动时调用。

在 aubo_scope 中集成程序节点的原理:

集成过程图如下:

具体实现:

ProgramNodeService

ProgramNodeService 接口必须实现以下函数:

getId()

参数: void

返回值类型: String

返回值内容: myNodeType

调用时机: 服务注册后。

说明: getId() 函数返回此类节点的唯一 ID。当存在多个程序节点时, 该函数用于标识哪些程序节点属于同一类程序节点。

getTitle()

参数: Locale 区域

返回值类型: String

返回值内容: 程序节点的名称。

调用时机: 服务注册后。

说明: 它是一个静态名称, 应该能够告诉用户该节点的用途。

getIcon()

参数: void

返回值类型: String

返回值内容: 返回节点的图标地址。

调用时机: 服务注册后。

说明: 由于每一类的节点其图标都各不相同, 在动态创建节点的过程中, 需要为其匹配相对应的图标, 因此我们需要在创建节点时拿到这类节点的图标链接地址。

configureContribution()

参数: ProgramContributionConfigurationPtr configuration

返回值类型: void

调用时机: 服务注册后。

说明:

- ProgramContributionConfigurationPtr 代表此类节点的配置;
- ProgramContributionConfigurationPtr 对象可以通过调用相关联的函数来配置其值;
- 例如设置程序节点是否可以有子节点、节点是否可以为用户插入还是仅仅在 aubo_caps 内部使用。

```
configuration.setChildrenAllowed( false );  
configuration.setUserInsertable( false );
```

createView()

参数: ViewApiProviderPtr api_provider

返回值类型: ProgramNodeViewPtr

返回值内容: 返回 view 类的新实例。

调用时机: 仅一次, 此类型的节点由用户第一次插入或者加载到程序中。

说明:

- 仅在用户首次使用该节点时才调用 `createView`。
- 参数 `ViewApiProviderPtr` 对象可以获取 `SystemAPI`、`UserInterfaceAPI`。
- 参数 `ViewApiProviderPtr` 通常会被转发到 `view` 类，例如需要在文本域中提供键盘，而键盘由 `UserInterfaceAPI` 提供。

`createNode()`

参数:

- `ProgramApiProviderPtr api_provider`,
- `ProgramNodeViewPtr view`,
- `DataModelPtr model`,
- `ProgramCreationContextPtr context`

返回值类型: `ProgramNodeContributionPtr`

返回值内容:

- 此类节点 `contribution` 类的新实例;
- `MyProgramNodeContribution (apiProvider, view, model, context)`。

调用时机:

- 每当创建此类型的节点时;
- 要么由用户插入新的节点，然后加载包含此类型节点的程序;
- 要么由另一个 `aubo_caps` 插入该程序。

说明:

- `createNode` 返回此类节点 `contribution` 类的新实例。虽然只有一个 `view` 实例，但是每次创建一个新节点时，都会创建一个 `contribution` 类的新实例;
- 应该将参数转发给 `contribution` 类。`ProgramApiProviderPtr` 可以提供与程序节点相关的各类 API 的访问权限，`contribution` 对象可以使用 `DataModelPtr` 存储用户完成的节点的配置信息，如果新插入节点，则 `DataModelPtr` 将为空；如果加载了该节点，则 `DataModelPtr` 已经包含用户对该节点的配置信息，`contribution` 对象可以使用 `ProgramCreationContextPtr` 来检查自身的创建是否是新插入节点的结果，或者该节点是否已经被加载。

ProgramNodeView

`ProgramNodeView` 接口必须实现以下函数:

`buildUi()`

参数:

- [WidgetHandle](#) `parent`,
- `ContributionProviderPtr <ProgramNodeContribution> contribution`

返回值类型: `void`

调用时机: 仅一次，当用户第一次插入或者加载这种类型节点到程序中。

说明:

- `widgetHandle` 是 `aubo_scope` 中显示给用户的面板，在调用 `buildUi` 期间，我们将节点的界面嵌套在 `aubo_scope` 面板中;
- 另一个参数是 `ContributionProviderPtr`。由于可以有多个 `Contribution` 实例，但是只有一个 `view` 实例，因此当用户与节点交互时 `ContributionProviderPtr` 将提供对 `Contribution` 的访问。调用 `contribution.get()` 将返回活动的 `ProgramNodeContribution` 实例。

ProgramNodeContribution

ProgramNodeContribution 接口必须实现以下函数：

openView()

参数: void

返回值类型: void

调用时机: 每次用户进入特定的节点界面时。

说明:

- 由于同一种类型的节点其用户界面 view 只有一个实例，但是这种类型的节点可能有多 个，其 ProgramNodeContribution 可能也有多个实例，因此应该调用 openView() 函数 来更新 view 以显示各个节点的特定设置；
- 通常，此调用会读取 dataModel 中的设置，这些设置可用于动态更新 Ui 中的相关设置。 在这个函数中也可以用来启动一个定时器，只要节点在焦点中，就可以实现动态的更新 Ui 中的值。

closeView()

参数: void

返回值类型: void

调用时机: 用户每次离开此特定节点的用户界面时。

说明: 对 closeView() 的调用会通知 Contribution 告知用户现在不再查看此节点。如果在 openView() 中启动了定时器或其他正在运行的任务，那么现在应该停止。

getTitle()

参数: void

返回值类型: String

返回值内容: 返回程序树中所示的程序节点的名称。

调用时机: 调用频繁，通常在更改 dataModel 时。

说明:

- Contribution 中 getTitle() 和 Service 中的 getTitle() 的区别: Contribution 中 getTitle() 函数返回的时程序树中所示的程序节点的名称，是用来区分同一类型的不同节 点的；Service 中的 getTitle() 函数返回的是可用节点列表中节点的基本名称，是一个 静态名称，是用来区别哪些节点是同一类型的。由此也说明了程序树中的节点的名称， 通常根据节点的配置而发生变化。
- 通常，此调用将基于当前配置来更改节点的名称。它用于简化用户在程序树中的导航。 例如，如果这是可设置特定夹持宽度的自适应手爪的程序节点，则在设置为 35mm 的夹 持宽度时，它可能会返回 ** “Gripper Move: 35 mm” **。这有助于用户找到要配置的 适当节点。如果未设置夹持宽度，即未定义节点，则标题可能只是 “Gripper Move”。

isDefined()

参数: void

返回值类型: bool

返回值内容: 目前的节点配置能否生成脚本，如果缺少用于生成 Script 的数据，返回 false， 反之返回 true。

调用时机: 每当程序中任何地方发生任何的修改时。

说明:

- 几乎在程序中的任何地方发生任何修改时，都会调用 isDefined()。这意味着即使此节 点不在焦点上，isDefined() 也会被调用很多次，这就要求在回调中执行的代码必须非 常精简，并且依赖于 dataModel 的值。
- 此函数的作用不是发出是否已经连接外部硬件的信号，这应该在运行时来处理。isDefined() 仅用于提醒用户是否需要对此节点进行更多的配置。

generateScript()

参数: ScriptWriterPtr script_writer

返回值类型: void

调用时机: 通常在启动或者保存程序时。

说明:

- 程序节点和安装节点的目的是生成一些 LuaScript，这些脚本将在运行程序时被执行。而 **generateScript()** 函数就是为这些节点生成相应的脚本的，生成的脚本取决于 dataModel 中的配置或者创建节点时 service 中设置的静态配置。
- 该脚本应使用 ScriptWriter 对象创建。

在 aubo_scope 中集成安装节点的原理:

集成过程图如下:

具体实现:

InstallationNodeService

InstallationNodeService 接口必须实现以下函数:

getTitle()

参数: Locale 区域

返回值类型: String

返回值内容: 返回该安装节点的一个静态名称，能够提示用户该节点的用途。

说明: 系统 Locale 作为参数，可用于返回本地化为系统语言的节点名称。

getIcon()

参数: void

返回值类型: String

返回值内容: 返回节点的图标地址。

调用时机: 服务注册后。

说明: 由于每一类的节点其图标都各不相同，在动态创建节点的过程中，需要为其匹配相对应的图标，因此我们需要在创建节点时拿到这类节点的图标链接地址。

configureContribution()

参数: InstallationContributionConfigurationPtr configuration

返回值类型: void

调用时机: 服务注册后。

说明: InstallationContributionConfigurationPtr 对象存储着这类节点的配置，它已实现供将来使用，目前不提供任何选项，它应该留为空白。

createView()

参数: ViewApiProviderPtr api_provider

返回值类型: InstallationNodeViewPtr

返回值内容: 返回 view 类的新实例。

调用时机: 仅一次，此类型的节点由用户第一次插入或者加载到程序中。

说明:

- 仅在用户首次使用该类节点时才调用 createView。
- 参数 ViewApiProviderPtr 对象可以获取 SystemAPI、UserInterfaceAPI。
- 参数 ViewApiProviderPtr 通常会被转发到 view 类，例如需要在文本域中提供键盘，而键盘由 UserInterfaceAPI 提供。

createNode()

参数:

- InstallationApiProviderPtr api_provider,
- InstallationNodeViewPtr view,
- DataModelPtr model,
- InstallationCreationContextPtr context

返回值类型: InstallationNodeContributionPtr

返回值内容:

- 此类节点 contribution 类的新实例;
- MyInstallationNodeContribution (apiProvider, view, model, context)。

调用时机:

- 每当创建此类型的节点时;
- 要么由用户插入新的节点, 然后加载包含此类型节点的程序;
- 要么由另一个 aubo_caps 插入该程序。

说明:

- createNode 返回此类节点 contribution 类的新实例。虽然只有一个 view 实例, 但是每次创建一个新节点时, 都会创建一个 contribution 类的新实例;
- 应该将参数转发给 contribution 类。ProgramApiProviderPtr 可以提供与程序节点相关的各类API 的访问权限、contribution 对象可以使用 DataModelPtr 存储用户完成的节点的配置信息, 如果新插入节点, 则 DataModelPtr 将为空; 如果加载了该节点 (例如作为安装文件的一部分), 则 DataModelPtr 已经包含用户对该节点的配置信息、contribution 对象可以使用 ProgramCreationContextPtr 来检查自身的创建是否是新插入节点的结果, 或者该节点是否已经被加载。

InstallationNodeView

InstallationNodeView 接口必须实现以下函数:

buildUi()

参数:

- [WidgetHandle](#) parent,
- InstallationNodeContributionPtr contribution

返回值类型: void

调用时机: 仅一次, 当用户第一次插入或者加载这种类型节点到程序中。

说明:

- widgetHandle 是 aubo_scope 中显示给用户的面板, 在调用 buildUi 期间, 我们将节点的界面嵌套在 aubo_scope 面板中;
- 另一个参数是 InstallationNodeContributionPtr。由于可以有多个 Contribution 实例, 但是只有一个 view 实例, 因此当用户与节点交互时 ContributionProviderPtr 将提供对 Contribution 的访问。调用 contribution.get() 将返回活动的 ProgramNode← Contribution 实例。

InstallationNodeContribution

InstallationNodeContribution 接口必须实现以下函数:

openView()

参数: void

返回值类型: void

调用时机: 每次用户进入特定的节点界面时。

说明:

- 由于同一种类型的节点其用户界面 view 只有一个实例，但是这种类型的节点可能有多个，其 `InstallationNodeContribution` 可能也有多个实例，因此应该调用 `openView()` 函数来更新 view 以显示各个节点的特定设置；
- 通常，此调用会读取 `dataModel` 中的设置，这些设置可用于动态更新 `Ui` 中的相关设置。在这个函数中也可以用来启动一个定时器，只要节点在焦点中，就可以实现动态的更新 `Ui` 中的值。

closeView()

参数: void

返回值类型: void

调用时机: 用户每次离开此特定节点的用户界面时。

说明: 对 `closeView()` 的调用会通知 `Contribution` 告知用户现在不再查看此节点。如果在 `openView()` 中启动了定时器或其他正在运行的任务，那么现在应该停止。

generateScript()

参数: `ScriptWriterPtr script_writer`

返回值类型: void

调用时机: 通常在启动或者保存程序时。

说明: 程序节点和安装节点的目的是生成一些 `Script`，这些脚本将在运行程序时被执行。而 `generateScript()` 函数就是为这些节点生成相应的 `lua` 脚本的，生成的脚本取决于 `dataModel` 中的配置或者创建节点时 `service` 中设置的静态配置。

SDK 架构

AuboStudio SDK 为机器人安装的插件提供了一组功能。其“核心”就是 `Contribution`。`Contribution` 是可以添加到 `aubo_scope` 中的新服务，为用户提供更加丰富的功能。有两种一般类型的 `Contribution`:

1. 安装节点

- 安装节点是机器人安装域的扩展；
- 安装节点之间没有顺序；
- 安装节点可以提供用户交互界面并添加功能以配置“环境设置”。

1. 程序节点

- 程序节点是机器人程序域的扩展；
- 程序节点之间在程序树中按顺序排放，并以此决定了机器人在运行程序时被执行的顺序；
- 程序节点可以为机器人的程序流程提供新的编程思想，并提供用户交互界面。

插件提供的扩展通用类型（安装节点，程序节点）两个用户界面可以访问 SDK 中的许多功能。而提供给这些节点的 API 通常称为域 API，这类 API 提供对机器人的域详细信息和逻辑的访问。

由于程序节点和安装节点是在机器人程序的上下文中获得的，该上下文具有特殊的 API 功能。这些功能仅在 `Installation` 域或 `Program` 域中才有意义，这两类 API 分别为安装 API、程序 API。

域 API

提供给一般节点类型的通用 API 称为域 API。但是，并非在任何给定时间，此 API 中的所有功能都对所有节点类型有意义。因此，将域 API 以合理的 API 子类型提供给使用 `aubo_caps` 的节点。

有三种基于域的 API。

1. 应用程序 API

提供对与机器人或应用程序特定功能相关的 API 的访问。即机器人的 I/O，程序变量或坐标系（特征）。

1. 系统 API

提供对包含有关机器人的系统信息的 API 的访问。即序列号，软件版本或系统语言。

1. 用户界面 API

提供与 UI 中与用户进行交互时相关的 API 的访问。即创建键盘，或要求用户定义位置。

安装 API

1. 提供仅在 Program 域中才有意义的功能。

2. 安装 API 是对应用程序 API 的扩展，包括了可用于安装节点的特殊功能，例如创建 Tcp、创建 Feature 等。

程序 API

1. 提供仅在 Installation 域中才有意义的功能。

2. 程序 API 是对系统 API 和用户界面 API 的扩展，包括了可用于程序节点的功能，例如构建程序节点模板。

在特定于节点类型的 Provider 对象中提供了对安装 API 和程序 API 的访问。

层次结构

```
域 API
  system_api (系统信息)
  user_interface_api (用户界面)
  application_api (应用程序)
安装 API
  installation_api (安装)
程序 API
  program_api (程序)
```

使用示例

开发 Service 类

目的: service 类提供了配置节点基本信息的函数、创建该节点的用户交互界面（view 类）的函数以及创建每个节点的 Contribution 类的函数。

例如开发一个名为 hello 的程序节点：

```
helloProgramNodeService::helloProgramNodeService(){}
helloProgramNodeService::~helloProgramNodeService(){}

std::string helloProgramNodeService::getId()
{
    return "hello";
}

std::string helloProgramNodeService::getTitle()
{
    return "hello";
}

std::string helloProgramNodeService::getIcon()
{
    return "img:node-comment.png";
}
```

```

}

void helloProgramNodeService::configureContribution(
    ProgramContributionConfigurationPtr configuration)
{
}

ProgramNodeViewPtr helloProgramNodeService::createView(
    ViewApiProviderPtr api_provider)
{
    return std::make_shared<helloProgramNodeView>(api_provider);
}

ProgramNodeContributionPtr helloProgramNodeService::createNode(
    ProgramApiProviderPtr api_provider, ProgramNodeViewPtr view,
    DataModelPtr model, ProgramCreationContextPtr context)
{
    return std::make_shared<helloProgramNodeContribution>(
        api_provider, std::dynamic_pointer_cast<helloProgramNodeView>(view),
        model);
}

```

在激活器中注册服务

当 service 类完成时，需要在激活器中注册此服务。激活器注册节点应该提供给 aubo_scope 的所有服务。当 aubo_scope 启动时，将调用激活器 start() 方法，并且应在 bundle context 中注册相关服务。

```

#include "cppmicroservices/Bundle.h"
#include "cppmicroservices/BundleActivator.h"
#include "cppmicroservices/BundleImport.h"
#include "cppmicroservices/BundleContext.h"
#include "cppmicroservices/BundleResource.h"
#include "cppmicroservices/BundleResourceStream.h"
#include "cppmicroservices/Constants.h"
#include "cppmicroservices/ServiceEvent.h"

#include "hello_program_node_service.h"

using namespace cppmicroservices;

/**
 * This class implements a simple bundle that utilizes the CppMicroServices's
 * event mechanism to listen for service events. Upon receiving a service event,
 * it prints out the event's details.
 */
class Activator : public BundleActivator
{
private:
    /**
     * Implements BundleActivator::Start(). Prints a message and adds a member
     * function to the bundle context as a service listener.
     *
     * @param context the framework context for the bundle.
     */
    void Start(BundleContext context)
    {
        auto bundle = context.GetBundle();
        auto headers = bundle.GetHeaders();

        cppmicroservices::ServiceProperties props;
        props["Name"] = std::string("MyFirstAuboCap");
        props["VendorName"] = std::string("aubo-robotics");
        props["Version"] = 0;

        //注册节点
        context.RegisterService<arcs::aubo_scope::ProgramNodeService>(
            std::make_shared<helloProgramNodeService>(), props);
    }

    /**
     * Implements BundleActivator::Stop(). Prints a message and removes the
     * member function from the bundle context as a service listener.
     *
     * @param context the framework context for the bundle.
     */
    void Stop(BundleContext /*context*/) {}
};

CPPMICROSERVICES_EXPORT_BUNDLE_ACTIVATOR(Activator)

```

开发 Contribution 类

目的: 为节点提供需要的特殊 API, 以及生成 Script 脚本用以执行程序节点的功能, 该代码基于 dataModel 中存储的配置信息。

例如开发一个名为 hello 的程序节点:

```
helloProgramNodeContribution::helloProgramNodeContribution(
    ProgramApiProviderPtr api_provider, helloProgramNodeViewPtr view,
    DataModelPtr model)
: api_provider_(api_provider), view_(view), model_(model)
{
}

helloProgramNodeContribution::~helloProgramNodeContribution()
{
}

void helloProgramNodeContribution::openView()
{
}

void helloProgramNodeContribution::closeView()
{
}

std::string helloProgramNodeContribution::getTitle()
{
    return "hello";
}

bool helloProgramNodeContribution::isDefined()
{
    return true;
}

void helloProgramNodeContribution::generateScript(ScriptWriterPtr script_writer)
{
}
```

开发 View 类

目的: 为节点提供用户交互界面。

例如开发一个名为 hello 的程序节点:

```
helloProgramNodeView::helloProgramNodeView(ViewApiProviderPtr api_provider,
                                             QWidget *parent)
: QWidget(parent), ui(new Ui::helloProgramNodeView), view_api_(api_provider)
{
}

helloProgramNodeView::~helloProgramNodeView()
{
    delete ui;
}

void helloProgramNodeView::buildUi(
    WidgetHandle parent,
    ContributionProviderPtr<ProgramNodeContribution> contribution)
{
    setParent((QWidget *)parent);
    ui->setupUi(this);
}
```

命名空间

所有接口都定义在以下命名空间中:

```
namespace arcs {
namespace aubo_scope {
    // 所有接口定义
}
}
```


Chapter 2

专题索引

2.1 专题

这里是所有专题及其简介:

ApplicationApi(应用程序)	65
InstallationApi(安装)	67
ProgramApi(程序)	68
SystemApi(系统信息)	69
UserInterfaceApi(用户界面)	76

Chapter 3

目录结构

3.1 目录

aubo_caps	79
contribution	81
driver	84
gripper	86
capability	81
grip_force_capability.h	564
grip_vacuum_capability.h	566
registered_capabilities.h	568
speed_capability.h	570
width_capability.h	572
grip_action_parameters.h	574
gripper_api_provider.h	577
gripper_configuration.h	579
gripper_contribution.h	583
release_action_parameters.h	587
screw_driver	90
screw_driver_api_provider.h	590
screw_driver_configuration.h	591
screw_driver_contribution.h	599
screw_driver_parameters.h	602
installation	87
installation_api_provider.h	605
installation_contribution_configuration.h	608
installation_creation_context.h	609
installation_node_contribution.h	611
installation_node_service.h	613
installation_node_view.h	616
program	89
program_api_provider.h	618
program_contribution_configuration.h	620
program_creation_context.h	623
program_node_contribution.h	624
program_node_service.h	627
program_node_view.h	630
contribution_provider.h	562
view_api_provider.h	631
domain	83
data	82
data_model.h	637
device	82

device.h	647
device_manager.h	650
device_registration_manager.h	652
gripper_device.h	654
feature	85
base_feature.h	659
feature.h	660
feature_contribution_model.h	663
feature_line.h	666
feature_model.h	668
feature_plane.h	671
feature_point.h	672
feature_tool.h	674
function	85
function.h	675
function_model.h	678
installation	87
installation_api.h	681
io	88
analog_io.h	684
boolean_register.h	687
digital_io.h	689
double_register.h	692
ethernet_ip.h	694
integer_register.h	697
io.h	698
io_model.h	701
modbus_io.h	703
profinet.h	707
register.h	709
payload	89
payload.h	711
payload_contribution_model.h	715
payload_model.h	721
program	90
nodes	88
builtin	80
assembly_node.h	724
assignment_node.h	726
break_node.h	729
call_node.h	731
case_node.h	733
circle_move_node.h	735
comment_node.h	737
conveyor_tracking_node.h	739
default_case_node.h	741
direction_node.h	743
else_if_node.h	746
else_node.h	749
folder_node.h	751
force_node.h	753
force_option_node.h	756
gripper_node.h	758
halt_node.h	762
if_node.h	764
loop_node.h	766
move_node.h	770
offline_track_node.h	775

pallet_node.h	777
popup_node.h	779
program_home_node.h	781
record_track_node.h	784
screw_driving_node.h	786
script_node.h	788
seek_node.h	790
set_node.h	792
set_payload_node.h	796
sub_prog_node.h	799
switch_node.h	801
thread_node.h	803
timer_node.h	805
until_node.h	807
visual_localization_program_node.h	813
wait_node.h	815
waypoint_node.h	819
program_node.h	826
program_node_factory.h	828
structure	91
tree_node.h	837
program_api.h	833
program_model.h	835
script	91
script_writer.h	842
tcp	92
tcp.h	854
tcp_contribution_model.h	857
tcp_model.h	860
undoredo	92
undo_redo_manager.h	863
user_interface	93
input_validator.h	866
keyboard_manager.h	869
popup_manager.h	871
render_interface.h	873
user_interaction.h	883
user_interface_api.h	887
value	93
expression	84
expression.h	889
value_factory_provider.h	891
variable	94
feature_variable.h	893
global_variable.h	896
persisted_variable.h	898
variable.h	900
variable_factory.h	903
variable_model.h	906
waypoint	94
waypoint.h	909
waypoint_model.h	911
application_api.h	633
extension_info.h	656
system_api.h	846
class_forward.h	559
builtin	80

assembly_node.h	724
assignment_node.h	726
break_node.h	729
call_node.h	731
case_node.h	733
circle_move_node.h	735
comment_node.h	737
conveyor_tracking_node.h	739
default_case_node.h	741
direction_node.h	743
else_if_node.h	746
else_node.h	749
folder_node.h	751
force_node.h	753
force_option_node.h	756
gripper_node.h	758
halt_node.h	762
if_node.h	764
loop_node.h	766
move_node.h	770
offline_track_node.h	775
pallet_node.h	777
popup_node.h	779
program_home_node.h	781
record_track_node.h	784
screw_driving_node.h	786
script_node.h	788
seek_node.h	790
set_node.h	792
set_payload_node.h	796
sub_prog_node.h	799
switch_node.h	801
thread_node.h	803
timer_node.h	805
until_node.h	807
visual_localization_program_node.h	813
wait_node.h	815
waypoint_node.h	819
capability	81
grip_force_capability.h	564
grip_vacuum_capability.h	566
registered_cpabilities.h	568
speed_capability.h	570
width_capability.h	572
contribution	81
driver	84
gripper	86
capability	81
grip_force_capability.h	564
grip_vacuum_capability.h	566
registered_cpabilities.h	568
speed_capability.h	570
width_capability.h	572
grip_action_parameters.h	574
gripper_api_provider.h	577
gripper_configuration.h	579
gripper_contribution.h	583
release_action_parameters.h	587

screw_driver	90
screw_driver_api_provider.h	590
screw_driver_configuration.h	591
screw_driver_contribution.h	599
screw_driver_parameters.h	602
installation	87
installation_api_provider.h	605
installation_contribution_configuration.h	608
installation_creation_context.h	609
installation_node_contribution.h	611
installation_node_service.h	613
installation_node_view.h	616
program	89
program_api_provider.h	618
program_contribution_configuration.h	620
program_creation_context.h	623
program_node_contribution.h	624
program_node_service.h	627
program_node_view.h	630
contribution_provider.h	562
view_api_provider.h	631
data	82
data_model.h	637
device	82
device.h	647
device_manager.h	650
device_registration_manager.h	652
gripper_device.h	654
doc	83
domain	83
data	82
data_model.h	637
device	82
device.h	647
device_manager.h	650
device_registration_manager.h	652
gripper_device.h	654
feature	85
base_feature.h	659
feature.h	660
feature_contribution_model.h	663
feature_line.h	666
feature_model.h	668
feature_plane.h	671
feature_point.h	672
feature_tool.h	674
function	85
function.h	675
function_model.h	678
installation	87
installation_api.h	681
io	88
analog_io.h	684
boolean_register.h	687
digital_io.h	689
double_register.h	692

ethernet_ip.h	694
integer_register.h	697
io.h	698
io_model.h	701
modbus_io.h	703
profinet.h	707
register.h	709
payload	89
payload.h	711
payload_contribution_model.h	715
payload_model.h	721
program	90
nodes	88
builtin	80
assembly_node.h	724
assignment_node.h	726
break_node.h	729
call_node.h	731
case_node.h	733
circle_move_node.h	735
comment_node.h	737
conveyor_tracking_node.h	739
default_case_node.h	741
direction_node.h	743
else_if_node.h	746
else_node.h	749
folder_node.h	751
force_node.h	753
force_option_node.h	756
gripper_node.h	758
halt_node.h	762
if_node.h	764
loop_node.h	766
move_node.h	770
offline_track_node.h	775
pallet_node.h	777
popup_node.h	779
program_home_node.h	781
record_track_node.h	784
screw_driving_node.h	786
script_node.h	788
seek_node.h	790
set_node.h	792
set_payload_node.h	796
sub_prog_node.h	799
switch_node.h	801
thread_node.h	803
timer_node.h	805
until_node.h	807
visual_localization_program_node.h	813
wait_node.h	815
waypoint_node.h	819
program_node.h	826
program_node_factory.h	828
structure	91
tree_node.h	837
program_api.h	833
program_model.h	835

script	91
script_writer.h	842
tcp	92
tcp.h	854
tcp_contribution_model.h	857
tcp_model.h	860
undoredo	92
undo_redo_manager.h	863
user_interface	93
input_validator.h	866
keyboard_manager.h	869
popup_manager.h	871
render_interface.h	873
user_interaction.h	883
user_interface_api.h	887
value	93
expression	84
expression.h	889
value_factory_provider.h	891
variable	94
feature_variable.h	893
global_variable.h	896
persisted_variable.h	898
variable.h	900
variable_factory.h	903
variable_model.h	906
waypoint	94
waypoint.h	909
waypoint_model.h	911
application_api.h	633
extension_info.h	656
system_api.h	846
driver	84
gripper	86
capability	81
grip_force_capability.h	564
grip_vacuum_capability.h	566
registered_capabilities.h	568
speed_capability.h	570
width_capability.h	572
grip_action_parameters.h	574
gripper_api_provider.h	577
gripper_configuration.h	579
gripper_contribution.h	583
release_action_parameters.h	587
screw_driver	90
screw_driver_api_provider.h	590
screw_driver_configuration.h	591
screw_driver_contribution.h	599
screw_driver_parameters.h	602
expression	84
expression.h	889
feature	85
base_feature.h	659
feature.h	660
feature_contribution_model.h	663

feature_line.h	666
feature_model.h	668
feature_plane.h	671
feature_point.h	672
feature_tool.h	674
function	85
function.h	675
function_model.h	678
gripper	86
capability	81
grip_force_capability.h	564
grip_vacuum_capability.h	566
registered_cpabilities.h	568
speed_capability.h	570
width_capability.h	572
grip_action_parameters.h	574
gripper_api_provider.h	577
gripper_configuration.h	579
gripper_contribution.h	583
release_action_parameters.h	587
include	86
aubo_caps	79
contribution	81
driver	84
gripper	86
capability	81
grip_force_capability.h	564
grip_vacuum_capability.h	566
registered_cpabilities.h	568
speed_capability.h	570
width_capability.h	572
grip_action_parameters.h	574
gripper_api_provider.h	577
gripper_configuration.h	579
gripper_contribution.h	583
release_action_parameters.h	587
screw_driver	90
screw_driver_api_provider.h	590
screw_driver_configuration.h	591
screw_driver_contribution.h	599
screw_driver_parameters.h	602
installation	87
installation_api_provider.h	605
installation_contribution_configuration.h	608
installation_creation_context.h	609
installation_node_contribution.h	611
installation_node_service.h	613
installation_node_view.h	616
program	89
program_api_provider.h	618
program_contribution_configuration.h	620
program_creation_context.h	623
program_node_contribution.h	624
program_node_service.h	627
program_node_view.h	630
contribution_provider.h	562
view_api_provider.h	631

domain	83
data	82
data_model.h	637
device	82
device.h	647
device_manager.h	650
device_registration_manager.h	652
gripper_device.h	654
feature	85
base_feature.h	659
feature.h	660
feature_contribution_model.h	663
feature_line.h	666
feature_model.h	668
feature_plane.h	671
feature_point.h	672
feature_tool.h	674
function	85
function.h	675
function_model.h	678
installation	87
installation_api.h	681
io	88
analog_io.h	684
boolean_register.h	687
digital_io.h	689
double_register.h	692
ethernet_ip.h	694
integer_register.h	697
io.h	698
io_model.h	701
modbus_io.h	703
profinet.h	707
register.h	709
payload	89
payload.h	711
payload_contribution_model.h	715
payload_model.h	721
program	90
nodes	88
builtin	80
assembly_node.h	724
assignment_node.h	726
break_node.h	729
call_node.h	731
case_node.h	733
circle_move_node.h	735
comment_node.h	737
conveyor_tracking_node.h	739
default_case_node.h	741
direction_node.h	743
else_if_node.h	746
else_node.h	749
folder_node.h	751
force_node.h	753
force_option_node.h	756
gripper_node.h	758

halt_node.h	762
if_node.h	764
loop_node.h	766
move_node.h	770
offline_track_node.h	775
pallet_node.h	777
popup_node.h	779
program_home_node.h	781
record_track_node.h	784
screw_driving_node.h	786
script_node.h	788
seek_node.h	790
set_node.h	792
set_payload_node.h	796
sub_prog_node.h	799
switch_node.h	801
thread_node.h	803
timer_node.h	805
until_node.h	807
visual_localization_program_node.h	813
wait_node.h	815
waypoint_node.h	819
program_node.h	826
program_node_factory.h	828
structure	91
tree_node.h	837
program_api.h	833
program_model.h	835
script	91
script_writer.h	842
tcp	92
tcp.h	854
tcp_contribution_model.h	857
tcp_model.h	860
undoredo	92
undo_redo_manager.h	863
user_interface	93
input_validator.h	866
keyboard_manager.h	869
popup_manager.h	871
render_interface.h	873
user_interaction.h	883
user_interface_api.h	887
value	93
expression	84
expression.h	889
value_factory_provider.h	891
variable	94
feature_variable.h	893
global_variable.h	896
persisted_variable.h	898
variable.h	900
variable_factory.h	903
variable_model.h	906
waypoint	94
waypoint.h	909
waypoint_model.h	911

application_api.h	633
extension_info.h	656
system_api.h	846
class_forward.h	559
installation	87
installation_api_provider.h	605
installation_contribution_configuration.h	608
installation_creation_context.h	609
installation_node_contribution.h	611
installation_node_service.h	613
installation_node_view.h	616
installation	87
installation_api.h	681
io	88
analog_io.h	684
boolean_register.h	687
digital_io.h	689
double_register.h	692
ethernet_ip.h	694
integer_register.h	697
io.h	698
io_model.h	701
modbus_io.h	703
profinet.h	707
register.h	709
nodes	88
builtin	80
assembly_node.h	724
assignment_node.h	726
break_node.h	729
call_node.h	731
case_node.h	733
circle_move_node.h	735
comment_node.h	737
conveyor_tracking_node.h	739
default_case_node.h	741
direction_node.h	743
else_if_node.h	746
else_node.h	749
folder_node.h	751
force_node.h	753
force_option_node.h	756
gripper_node.h	758
halt_node.h	762
if_node.h	764
loop_node.h	766
move_node.h	770
offline_track_node.h	775
pallet_node.h	777
popup_node.h	779
program_home_node.h	781
record_track_node.h	784
screw_driving_node.h	786
script_node.h	788
seek_node.h	790
set_node.h	792
set_payload_node.h	796

sub_prog_node.h	799
switch_node.h	801
thread_node.h	803
timer_node.h	805
until_node.h	807
visual_localization_program_node.h	813
wait_node.h	815
waypoint_node.h	819
program_node.h	826
program_node_factory.h	828
payload	89
payload.h	711
payload_contribution_model.h	715
payload_model.h	721
program	89
program_api_provider.h	618
program_contribution_configuration.h	620
program_creation_context.h	623
program_node_contribution.h	624
program_node_service.h	627
program_node_view.h	630
program	90
nodes	88
builtin	80
assembly_node.h	724
assignment_node.h	726
break_node.h	729
call_node.h	731
case_node.h	733
circle_move_node.h	735
comment_node.h	737
conveyor_tracking_node.h	739
default_case_node.h	741
direction_node.h	743
else_if_node.h	746
else_node.h	749
folder_node.h	751
force_node.h	753
force_option_node.h	756
gripper_node.h	758
halt_node.h	762
if_node.h	764
loop_node.h	766
move_node.h	770
offline_track_node.h	775
pallet_node.h	777
popup_node.h	779
program_home_node.h	781
record_track_node.h	784
screw_driving_node.h	786
script_node.h	788
seek_node.h	790
set_node.h	792
set_payload_node.h	796
sub_prog_node.h	799
switch_node.h	801
thread_node.h	803

timer_node.h	805
until_node.h	807
visual_localization_program_node.h	813
wait_node.h	815
waypoint_node.h	819
program_node.h	826
program_node_factory.h	828
structure	91
tree_node.h	837
program_api.h	833
program_model.h	835
screw_driver	90
screw_driver_api_provider.h	590
screw_driver_configuration.h	591
screw_driver_contribution.h	599
screw_driver_parameters.h	602
script	91
script_writer.h	842
structure	91
tree_node.h	837
tcp	92
tcp.h	854
tcp_contribution_model.h	857
tcp_model.h	860
undoredo	92
undo_redo_manager.h	863
user_interface	93
input_validator.h	866
keyboard_manager.h	869
popup_manager.h	871
render_interface.h	873
user_interaction.h	883
user_interface_api.h	887
value	93
expression	84
expression.h	889
value_factory_provider.h	891
variable	94
feature_variable.h	893
global_variable.h	896
persisted_variable.h	898
variable.h	900
variable_factory.h	903
variable_model.h	906
waypoint	94
waypoint.h	909
waypoint_model.h	911

Chapter 4

命名空间索引

4.1 命名空间列表

这里列出了所有命名空间定义，附带简要说明:

arcs	97
arcs::aubo_scope	97
arcs::aubo_sdk	116

Chapter 5

继承关系索引

5.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

arcs::aubo_scope::AnalogIoFilter	121
arcs::aubo_scope::ApplicationApi	122
arcs::aubo_scope::InstallationApi	277
arcs::aubo_scope::ProgramApi	359
arcs::aubo_scope::AuboCapInfo	133
arcs::aubo_scope::ContributionProvider< T >	152
arcs::aubo_scope::DataModel	157
arcs::aubo_scope::DefaultSingletonTraits< Type >	170
arcs::aubo_scope::Device	170
arcs::aubo_scope::GripperDevice	262
arcs::aubo_scope::DeviceManager	172
arcs::aubo_scope::DeviceRegistrationManager	173
arcs::aubo_scope::DigitalIoFilter	177
arcs::aubo_scope::DriveScrewNotOKParameters	189
arcs::aubo_scope::DriveScrewOKParameters	189
arcs::aubo_scope::EtherNetIPFilter	200
arcs::aubo_scope::Expression	201
arcs::aubo_scope::Feature	204
arcs::aubo_scope::BaseFeature	134
arcs::aubo_scope::FeatureLine	212
arcs::aubo_scope::FeaturePlane	218
arcs::aubo_scope::FeaturePoint	221
arcs::aubo_scope::FeatureTool	224
arcs::aubo_scope::FeatureContributionModel	207
arcs::aubo_scope::FeatureDefinedFilter	211
arcs::aubo_scope::FeatureJoggableFilter	212
arcs::aubo_scope::FeatureModel	215
arcs::aubo_scope::FeatureVariableFilter	231
arcs::aubo_scope::FeedScrewParameters	231
arcs::aubo_scope::Function	241
arcs::aubo_scope::FunctionModel	243
arcs::aubo_scope::GlobalVariableFilter	250
arcs::aubo_scope::GripActionParameters	250
arcs::aubo_scope::GripForceCapability	252
arcs::aubo_scope::GripperApiProvider	253
arcs::aubo_scope::GripperConfiguration	254
arcs::aubo_scope::GripperContribution	260
arcs::aubo_scope::GripVacuumCapability	268
arcs::aubo_scope::InputIoFilter	275

arcs::aubo_scope::InputValidator	275
arcs::aubo_scope::DoubleRangeValidator	184
arcs::aubo_scope::IntegerRangeValidator	289
arcs::aubo_scope::StringLengthValidator	456
arcs::aubo_scope::UIntegerRangeValidator	488
arcs::aubo_scope::InstallationApiProvider	280
arcs::aubo_scope::InstallationContributionConfiguration	282
arcs::aubo_scope::InstallationCreationContext	283
arcs::aubo_scope::InstallationNodeContribution	285
arcs::aubo_scope::InstallationNodeService	286
arcs::aubo_scope::InstallationNodeView	288
arcs::aubo_scope::Io	293
arcs::aubo_scope::AnalogIo	117
arcs::aubo_scope::DigitalIo	174
arcs::aubo_scope::EtherNetIP	196
arcs::aubo_scope::ModbusIo	312
arcs::aubo_scope::Profinet	354
arcs::aubo_scope::Register	397
arcs::aubo_scope::BoolRegister	137
arcs::aubo_scope::DoubleRegister	187
arcs::aubo_scope::IntegerRegister	291
arcs::aubo_scope::IoConfigure	298
arcs::aubo_scope::IoModel	300
arcs::aubo_scope::KeyboardManager	302
arcs::aubo_scope::ModbusIoFilter	317
arcs::aubo_scope::OutputIoFilter	327
arcs::aubo_scope::Payload	330
arcs::aubo_scope::PayloadContributionModel	335
arcs::aubo_scope::PayloadModel	341
arcs::aubo_scope::PersistedVariableFilter	347
arcs::aubo_scope::PopupManager	347
arcs::aubo_scope::PrepareToStartScrewdriverParameters	353
arcs::aubo_scope::ProfinetFilter	358
arcs::aubo_scope::ProgramApiProvider	363
arcs::aubo_scope::ProgramContributionConfiguration	365
arcs::aubo_scope::ProgramCreationContext	370
arcs::aubo_scope::ProgramModel	375
arcs::aubo_scope::ProgramNode	378
arcs::aubo_scope::AssemblyNode	125
arcs::aubo_scope::AssignmentNode	129
arcs::aubo_scope::BreakNode	139
arcs::aubo_scope::CallNode	141
arcs::aubo_scope::CaseNode	144
arcs::aubo_scope::CircleMoveNode	147
arcs::aubo_scope::CommentNode	149
arcs::aubo_scope::ConveyorTrackingNode	153
arcs::aubo_scope::DefaultCaseNode	167
arcs::aubo_scope::DirectionNode	178
arcs::aubo_scope::ElseIfNode	190
arcs::aubo_scope::ElseNode	193
arcs::aubo_scope::FolderNode	232
arcs::aubo_scope::ForceNode	235
arcs::aubo_scope::ForceOptionNode	238
arcs::aubo_scope::GripperNode	264
arcs::aubo_scope::HaltNode	269
arcs::aubo_scope::IfNode	272
arcs::aubo_scope::LoopNode	307
arcs::aubo_scope::MoveNode	318

arcs::aubo_scope::OfflineTrackNode	325
arcs::aubo_scope::PalletNode	328
arcs::aubo_scope::PopupNode	350
arcs::aubo_scope::ProgramHomeNode	372
arcs::aubo_scope::RecordTrackNode	395
arcs::aubo_scope::ScrewdrivingNode	428
arcs::aubo_scope::ScriptNode	431
arcs::aubo_scope::SeekNode	442
arcs::aubo_scope::SetNode	444
arcs::aubo_scope::SetPayloadNode	451
arcs::aubo_scope::SubProgNode	458
arcs::aubo_scope::SwitchNode	461
arcs::aubo_scope::ThreadNode	477
arcs::aubo_scope::TimerNode	479
arcs::aubo_scope::UntilNode	493
arcs::aubo_scope::VisualLocalizationProgramNode	528
arcs::aubo_scope::WaitNode	531
arcs::aubo_scope::WaypointNode	546
arcs::aubo_scope::ProgramNodeContribution	381
arcs::aubo_scope::ProgramNodeFactory	384
arcs::aubo_scope::ProgramNodeService	391
arcs::aubo_scope::ProgramNodeView	393
arcs::aubo_scope::ProgramSelectionParameters	393
arcs::aubo_scope::RegisteredCapabilities	399
arcs::aubo_scope::ReleaseActionParameters	401
arcs::aubo_scope::RenderInterface	402
arcs::aubo_scope::ScrewdriverApiProvider	416
arcs::aubo_scope::ScrewdriverConfiguration	417
arcs::aubo_scope::ScrewdriverContribution	423
arcs::aubo_scope::ScrewdriverParameters	425
arcs::aubo_scope::ScrewdriverProgram	427
arcs::aubo_scope::ScrewdriverReadyParameters	427
arcs::aubo_scope::ScriptWriter	433
arcs::aubo_scope::SpeedCapability	455
arcs::aubo_scope::SystemApi	463
arcs::aubo_scope::TCP	467
arcs::aubo_scope::TcpContributionModel	469
arcs::aubo_scope::TcpModel	473
arcs::aubo_scope::TreeNode	482
arcs::aubo_scope::UndoRedoManager	491
arcs::aubo_scope::UserInteraction	503
arcs::aubo_scope::UserInterfaceApi	510
arcs::aubo_scope::ValueFactory	512
arcs::aubo_scope::Variable	514
arcs::aubo_scope::GlobalVariable	247
arcs::aubo_scope::PersistedVariable	343
arcs::aubo_scope::FeatureVariable	227
arcs::aubo_scope::VariableFactory	518
arcs::aubo_scope::VariableModel	521
arcs::aubo_scope::ViewApiProvider	525
arcs::aubo_scope::Waypoint	539
arcs::aubo_scope::WaypointModel	542
arcs::aubo_scope::WidthCapability	556

Chapter 6

类索引

6.1 类列表

这里列出了所有类、结构、联合以及接口定义等，并附带简要说明：

arcs::aubo_scope::AnalogIo	117
模拟 IO 接口此接口提供对模拟 I/O 的支持。	
arcs::aubo_scope::AnalogIoFilter	121
arcs::aubo_scope::ApplicationApi	122
arcs::aubo_scope::AssemblyNode	125
arcs::aubo_scope::AssignmentNode	129
arcs::aubo_scope::AuboCapInfo	
This interface provides various metadata information about an installed AuboCap, such as the name of the AuboCap	133
arcs::aubo_scope::BaseFeature	
This interface represents the Base feature in AuboScope	134
arcs::aubo_scope::BoolRegister	
This interface provides support for bool registers	137
arcs::aubo_scope::BreakNode	139
arcs::aubo_scope::CallNode	141
arcs::aubo_scope::CaseNode	144
arcs::aubo_scope::CircleMoveNode	147
arcs::aubo_scope::CommentNode	149
arcs::aubo_scope::ContributionProvider< T >	
贡献提供者此接口提供对贡献的访问。	152
arcs::aubo_scope::ConveyorTrackingNode	153
arcs::aubo_scope::DataModel	
数据模型此接口用于存储和管理表示当前配置的数据，例如 ProgramNodeContribution 或 InstallationNodeContribution 的配置。提供了在字典中添加、删除、获取和修改值的方法。	157
arcs::aubo_scope::DefaultCaseNode	167
arcs::aubo_scope::DefaultSingletonTraits< Type >	170
arcs::aubo_scope::Device	
设备接口此基础接口表示 AuboScope 支持的设备，例如夹爪设备。	170
arcs::aubo_scope::DeviceManager	
设备管理器设备管理器的基础接口。	172
arcs::aubo_scope::DeviceRegistrationManager	
设备注册管理器设备注册管理器的基础接口。	173
arcs::aubo_scope::DigitalIo	
数字 IO 接口此接口提供对数字 I/O 的支持。	174
arcs::aubo_scope::DigitalIoFilter	177
arcs::aubo_scope::DirectionNode	178
arcs::aubo_scope::DoubleRangeValidator	184
arcs::aubo_scope::DoubleRegister	
This interface provides support for float registers	187

<code>arcs::aubo_scope::DriveScrewNotOKParameters</code>	189
<code>arcs::aubo_scope::DriveScrewOKParameters</code>	189
<code>arcs::aubo_scope::ElseIfNode</code>	190
<code>arcs::aubo_scope::ElseNode</code>	193
<code>arcs::aubo_scope::EtherNetIP</code>	
EtherNet/IP IO 接口此接口提供对 EtherNet/IP I/O 的支持。	196
<code>arcs::aubo_scope::EtherNetIPFilter</code>	200
<code>arcs::aubo_scope::Expression</code>	
表达式表达式可用于配置各种内置程序节点, 例如 If 和 Wait 节点。表达式通常是条件表达式 (计算结果为 true 或 false)。	201
<code>arcs::aubo_scope::Feature</code>	
特征接口此基础接口表示 AuboScope 中所有可用的特征类型。	204
<code>arcs::aubo_scope::FeatureContributionModel</code>	
特征贡献模型此接口提供在 AuboScope 中添加、更新和删除特征的功能。	207
<code>arcs::aubo_scope::FeatureDefinedFilter</code>	211
<code>arcs::aubo_scope::FeatureJoggableFilter</code>	212
<code>arcs::aubo_scope::FeatureLine</code>	212
<code>arcs::aubo_scope::FeatureModel</code>	
特征模型提供从当前机器人安装中获取特征的方法。	215
<code>arcs::aubo_scope::FeaturePlane</code>	218
<code>arcs::aubo_scope::FeaturePoint</code>	
This interface represents a Point feature in AuboScope	221
<code>arcs::aubo_scope::FeatureTool</code>	
This interface represents the Tool feature in AuboScope	224
<code>arcs::aubo_scope::FeatureVariable</code>	
特征变量在安装中被标记为变量的特征, 即使在机器人关闭和重新启动后也会存在。	227
<code>arcs::aubo_scope::FeatureVariableFilter</code>	231
<code>arcs::aubo_scope::FeedScrewParameters</code>	231
<code>arcs::aubo_scope::FolderNode</code>	232
<code>arcs::aubo_scope::ForceNode</code>	235
<code>arcs::aubo_scope::ForceOptionNode</code>	238
<code>arcs::aubo_scope::Function</code>	
函数接口表示在表达式编辑器中显示的函数。	241
<code>arcs::aubo_scope::FunctionModel</code>	
函数模型此接口提供在表达式编辑器中添加和删除脚本函数的方法, 该编辑器可从赋值和 If 程序节点等位置访问。	243
<code>arcs::aubo_scope::GlobalVariable</code>	
Global variables are available everywhere in a program	247
<code>arcs::aubo_scope::GlobalVariableFilter</code>	250
<code>arcs::aubo_scope::GripActionParameters</code>	250
<code>arcs::aubo_scope::GripForceCapability</code>	
夹持力能力此接口表示夹爪已注册的力度能力, 支持使用用户可配置的力度进行夹取。	252
<code>arcs::aubo_scope::GripperApiProvider</code>	
夹爪 API 提供者此接口提供对 AuboScope 中与夹爪相关的功能和服务的访问。	253
<code>arcs::aubo_scope::GripperConfiguration</code>	
This interface provides access to register or setup various properties and capabilities of a gripper	254
<code>arcs::aubo_scope::GripperContribution</code>	
夹爪贡献此接口定义了向 AuboScope 添加夹爪设备功能的 API。	260
<code>arcs::aubo_scope::GripperDevice</code>	
夹爪设备此接口表示 AuboScope 中的夹爪设备。	262
<code>arcs::aubo_scope::GripperNode</code>	264
<code>arcs::aubo_scope::GripVacuumCapability</code>	
真空夹持能力此接口表示夹爪已注册的真空能力, 支持使用用户可配置的真空度进行夹取。	268
<code>arcs::aubo_scope::HaltNode</code>	269
<code>arcs::aubo_scope::IfNode</code>	272

arcs::aubo_scope::InputIoFilter	275
arcs::aubo_scope::InputValidator	
输入验证器表示由 InputValidationFactory 创建的输入验证器接口。该工厂提供一组标准输入验证器，可用于验证用户使用虚拟键盘/小键盘输入的内容。	275
arcs::aubo_scope::InstallationApi	277
arcs::aubo_scope::InstallationApiProvider	
安装 API 提供者提供与安装节点相关的功能和服务访问。这包括 AuboScope 内部已有的功能，以及在 AuboScope 中创建要使用的额外模型元素。	280
arcs::aubo_scope::InstallationContributionConfiguration	282
arcs::aubo_scope::InstallationCreationContext	
安装创建上下文提供安装节点创建时所在上下文的相关信息。	283
arcs::aubo_scope::InstallationNodeContribution	
安装节点贡献安装节点贡献的 API。	285
arcs::aubo_scope::InstallationNodeService	
安装节点服务定义了 AuboScope 中定义和添加安装节点及对应界面（用户界面基于 Qt）所需的 API。	286
arcs::aubo_scope::InstallationNodeView	
安装节点视图安装节点视图的 API，定义了对应 InstallationNodeContribution 的 UI。	288
arcs::aubo_scope::IntegerRangeValidator	289
arcs::aubo_scope::IntegerRegister	
This interface provides support for integer registers	291
arcs::aubo_scope::Io	
IO 基础接口这是表示 AuboScope 中所有可用 I/O 类型的基础接口。	293
arcs::aubo_scope::IoConfigure	298
arcs::aubo_scope::IoModel	
IO 模型提供从当前机器人安装中获取 I/O 的方法。	300
arcs::aubo_scope::KeyboardManager	302
arcs::aubo_scope::LoopNode	307
arcs::aubo_scope::ModbusIo	
Modbus IO 接口此接口提供对 MODBUS I/O 的支持。	312
arcs::aubo_scope::ModbusIoFilter	317
arcs::aubo_scope::MoveNode	318
arcs::aubo_scope::OfflineTrackNode	325
arcs::aubo_scope::OutputIoFilter	327
arcs::aubo_scope::PalletNode	328
arcs::aubo_scope::Payload	
获取负载参数API 接口	330
arcs::aubo_scope::PayloadContributionModel	
动态负载管理	335
arcs::aubo_scope::PayloadModel	
获取负载参数API 接口	341
arcs::aubo_scope::PersistedVariable	
Ingroup ApplicationApi This interface represents variables stored with the installation	343
arcs::aubo_scope::PersistedVariableFilter	347
arcs::aubo_scope::PopupManager	347
arcs::aubo_scope::PopupNode	350
arcs::aubo_scope::PrepareToStartScrewdriverParameters	353
arcs::aubo_scope::Profinet	
PROFINET IO 接口此接口提供对 PROFINET I/O 的支持。	354
arcs::aubo_scope::ProfinetFilter	358
arcs::aubo_scope::ProgramApi	359
arcs::aubo_scope::ProgramApiProvider	
程序 API 提供者提供与程序节点相关的功能和服务访问。这包括 AuboScope 内部已有的功能，以及在 AuboScope 中创建要使用的额外模型元素。	363
arcs::aubo_scope::ProgramContributionConfiguration	
程序贡献配置提供对安装节点贡献属性进行配置的访问。	365

<code>arcs::aubo_scope::ProgramCreationContext</code>	程序创建上下文提供安装节点创建时所在上下文的相关信息。	370
<code>arcs::aubo_scope::ProgramHomeNode</code>		372
<code>arcs::aubo_scope::ProgramModel</code>	程序模型	375
<code>arcs::aubo_scope::ProgramNode</code>	程序节点此接口表示 AuboScope 中的程序节点。	378
<code>arcs::aubo_scope::ProgramNodeContribution</code>	程序节点贡献定义了向 AuboScope 中指定 AuboCap 程序节点所需的 API。	381
<code>arcs::aubo_scope::ProgramNodeFactory</code>	程序节点工厂此接口提供创建不同类型程序节点的方法。	384
<code>arcs::aubo_scope::ProgramNodeService</code>	程序节点服务定义了向 AuboScope 添加新类型程序节点所需的 API。	391
<code>arcs::aubo_scope::ProgramNodeView</code>	程序节点视图程序节点视图的 API, 定义了对应 <code>ProgramNodeContribution</code> 的 UI。 <code>ProgramNodeView</code> 实例是多个贡献之间的共享实体。	393
<code>arcs::aubo_scope::ProgramSelectionParameters</code>		393
<code>arcs::aubo_scope::RecordTrackNode</code>		395
<code>arcs::aubo_scope::Register</code>	This interface provides support for registers	397
<code>arcs::aubo_scope::RegisteredCapabilities</code>		399
<code>arcs::aubo_scope::ReleaseActionParameters</code>		401
<code>arcs::aubo_scope::RenderInterface</code>	渲染接口提供 3D 渲染相关的显示选项控制, 包括地面、机器人模型、坐标系、轨迹和点等元素的显示/隐藏。	402
<code>arcs::aubo_scope::ScrewdriverApiProvider</code>	螺丝刀 API 提供者此接口提供对 AuboScope 中与螺丝刀贡献相关的功能和服务的访问。	416
<code>arcs::aubo_scope::ScrewdriverConfiguration</code>	This interface provides access to register or setup various properties and capabilities of a screwdriver	417
<code>arcs::aubo_scope::ScrewdriverContribution</code>	螺丝刀贡献此接口定义了向 AuboScope 添加螺丝刀功能的 API。	423
<code>arcs::aubo_scope::ScrewdriverParameters</code>	This interface represents parameters for a screwdriver operation defined/configured by the end user	425
<code>arcs::aubo_scope::ScrewdriverProgram</code>	This abstract class represents a screwdriver program, typically defined on an external control box for the screwdriver	427
<code>arcs::aubo_scope::ScrewdriverReadyParameters</code>	This interface provides the parameters relevant for generating script for the Screwdriver Ready feedback operation capability (registered using <code>ScrewdriverFeedbackCapabilities#registerScrewdriverReadyCapability(ScriptCodeGenerator)</code>)	427
<code>arcs::aubo_scope::ScrewdrivingNode</code>		428
<code>arcs::aubo_scope::ScriptNode</code>		431
<code>arcs::aubo_scope::ScriptWriter</code>	脚本写入器此接口提供生成脚本代码的支持。	433
<code>arcs::aubo_scope::SeekNode</code>		442
<code>arcs::aubo_scope::SetNode</code>		444
<code>arcs::aubo_scope::SetPayloadNode</code>		451
<code>arcs::aubo_scope::SpeedCapability</code>	速度能力此接口表示夹爪已注册的速度能力, 支持使用用户可配置的速度进行操作。	455
<code>arcs::aubo_scope::StringLengthValidator</code>		456
<code>arcs::aubo_scope::SubProgNode</code>		458
<code>arcs::aubo_scope::SwitchNode</code>		461
<code>arcs::aubo_scope::SystemApi</code>		463

arcs::aubo_scope::TCP	
TCP 接口此接口表示 AuboScope 中的一个 TCP（工具中心点）。可以是用户定义的 TCP，也可以是由 AuboCap 添加的 TCP。	467
arcs::aubo_scope::TcpContributionModel	
TCP 贡献模型此接口提供在 AuboScope 中添加、更新和删除 TCP 的功能。	469
arcs::aubo_scope::TcpModel	
TCP 模型此接口提供对 AuboScope 中当前可用的 TCP 的访问。	473
arcs::aubo_scope::ThreadNode	477
arcs::aubo_scope::TimerNode	479
arcs::aubo_scope::TreeNode	
树节点此接口表示程序树中的一个节点，可用于构建以 AuboCap 程序节点为根的子树。	482
arcs::aubo_scope::UIntegerRangeValidator	488
arcs::aubo_scope::UndoRedoManager	
撤销/重做管理器此接口用于记录终端用户在 AuboCap 程序节点贡献画面中触发的更改（例如，点击按钮时）。应在 AuboCap 程序节点中使用，以支持 AuboScope 中的撤销/重做功能。	491
arcs::aubo_scope::UntilNode	
493	
arcs::aubo_scope::UserInteraction	
用户交互接口此接口提供向终端用户请求输入或操作的功能。	503
arcs::aubo_scope::UserInterfaceApi	510
arcs::aubo_scope::ValueFactory	
值工厂接口	512
arcs::aubo_scope::Variable	
变量接口表示 AuboScope 中的变量。变量可用于在程序中存储和传递数据。	514
arcs::aubo_scope::VariableFactory	518
arcs::aubo_scope::VariableModel	
变量模型提供返回系统中当前已注册变量的方法。变量在存储到 DataModel 实例中或用于配置内置 AuboScope 程序节点时即被注册。	521
arcs::aubo_scope::ViewApiProvider	
视图 API 提供者提供与安装或程序节点贡献的视图（UI）相关的功能访问。	525
arcs::aubo_scope::VisualLocalizationProgramNode	528
arcs::aubo_scope::WaitNode	531
arcs::aubo_scope::Waypoint	
路点接口此接口组合了定义机器人位置的参数。	539
arcs::aubo_scope::WaypointModel	
路点模型提供从当前机器人获取路点的方法。	542
arcs::aubo_scope::WaypointNode	546
arcs::aubo_scope::WidthCapability	
宽度能力此接口表示夹爪已注册的宽度/位置能力，支持移动到用户可配置的位置（打开/关闭到可配置的宽度）。	556

Chapter 7

文件索引

7.1 文件列表

这里列出了所有文件，并附带简要说明:

include/aubo_caps/class_forward.h	559
include/aubo_caps/contribution/contribution_provider.h	562
include/aubo_caps/contribution/view_api_provider.h	631
include/aubo_caps/contribution/driver/gripper/grip_action_parameters.h	574
include/aubo_caps/contribution/driver/gripper/gripper_api_provider.h	577
include/aubo_caps/contribution/driver/gripper/gripper_configuration.h	579
include/aubo_caps/contribution/driver/gripper/gripper_contribution.h	583
include/aubo_caps/contribution/driver/gripper/release_action_parameters.h	587
include/aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h	564
include/aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h	566
include/aubo_caps/contribution/driver/gripper/capability/registered_capabilities.h	568
include/aubo_caps/contribution/driver/gripper/capability/speed_capability.h	570
include/aubo_caps/contribution/driver/gripper/capability/width_capability.h	572
include/aubo_caps/contribution/driver/screw_driver/screw_driver_api_provider.h	590
include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h	591
include/aubo_caps/contribution/driver/screw_driver/screw_driver_contribution.h	599
include/aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h	602
include/aubo_caps/contribution/installation/installation_api_provider.h	605
include/aubo_caps/contribution/installation/installation_contribution_configuration.h	608
include/aubo_caps/contribution/installation/installation_creation_context.h	609
include/aubo_caps/contribution/installation/installation_node_contribution.h	611
include/aubo_caps/contribution/installation/installation_node_service.h	613
include/aubo_caps/contribution/installation/installation_node_view.h	616
include/aubo_caps/contribution/program/program_api_provider.h	618
include/aubo_caps/contribution/program/program_contribution_configuration.h	620
include/aubo_caps/contribution/program/program_creation_context.h	623
include/aubo_caps/contribution/program/program_node_contribution.h	624
include/aubo_caps/contribution/program/program_node_service.h	627
include/aubo_caps/contribution/program/program_node_view.h	630
include/aubo_caps/domain/application_api.h	633
include/aubo_caps/domain/extension_info.h	656
include/aubo_caps/domain/system_api.h	846
include/aubo_caps/domain/data/data_model.h	637
include/aubo_caps/domain/device/device.h	647
include/aubo_caps/domain/device/device_manager.h	650
include/aubo_caps/domain/device/device_registration_manager.h	652
include/aubo_caps/domain/device/gripper_device.h	654
include/aubo_caps/domain/feature/base_feature.h	659
include/aubo_caps/domain/feature/feature.h	660
include/aubo_caps/domain/feature/feature_contribution_model.h	663

include/aubo_caps/domain/feature/feature_line.h	666
include/aubo_caps/domain/feature/feature_model.h	668
include/aubo_caps/domain/feature/feature_plane.h	671
include/aubo_caps/domain/feature/feature_point.h	672
include/aubo_caps/domain/feature/feature_tool.h	674
include/aubo_caps/domain/function/function.h	675
include/aubo_caps/domain/function/function_model.h	678
include/aubo_caps/domain/installation/installation_api.h	681
include/aubo_caps/domain/io/analog_io.h	684
include/aubo_caps/domain/io/boolean_register.h	687
include/aubo_caps/domain/io/digital_io.h	689
include/aubo_caps/domain/io/double_register.h	692
include/aubo_caps/domain/io/ethernet_ip.h	694
include/aubo_caps/domain/io/integer_register.h	697
include/aubo_caps/domain/io/io.h	698
include/aubo_caps/domain/io/io_model.h	701
include/aubo_caps/domain/io/modbus_io.h	703
include/aubo_caps/domain/io/profinet.h	707
include/aubo_caps/domain/io/register.h	709
include/aubo_caps/domain/payload/payload.h	711
include/aubo_caps/domain/payload/payload_contribution_model.h	715
include/aubo_caps/domain/payload/payload_model.h	721
include/aubo_caps/domain/program/program_api.h	833
include/aubo_caps/domain/program/program_model.h	835
include/aubo_caps/domain/program/nodes/program_node.h	826
include/aubo_caps/domain/program/nodes/program_node_factory.h	828
include/aubo_caps/domain/program/nodes/builtin/assembly_node.h	724
include/aubo_caps/domain/program/nodes/builtin/assignment_node.h	726
include/aubo_caps/domain/program/nodes/builtin/break_node.h	729
include/aubo_caps/domain/program/nodes/builtin/call_node.h	731
include/aubo_caps/domain/program/nodes/builtin/case_node.h	733
include/aubo_caps/domain/program/nodes/builtin/circle_move_node.h	735
include/aubo_caps/domain/program/nodes/builtin/comment_node.h	737
include/aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h	739
include/aubo_caps/domain/program/nodes/builtin/default_case_node.h	741
include/aubo_caps/domain/program/nodes/builtin/direction_node.h	743
include/aubo_caps/domain/program/nodes/builtin/else_if_node.h	746
include/aubo_caps/domain/program/nodes/builtin/else_node.h	749
include/aubo_caps/domain/program/nodes/builtin/folder_node.h	751
include/aubo_caps/domain/program/nodes/builtin/force_node.h	753
include/aubo_caps/domain/program/nodes/builtin/force_option_node.h	756
include/aubo_caps/domain/program/nodes/builtin/gripper_node.h	758
include/aubo_caps/domain/program/nodes/builtin/halt_node.h	762
include/aubo_caps/domain/program/nodes/builtin/if_node.h	764
include/aubo_caps/domain/program/nodes/builtin/loop_node.h	766
include/aubo_caps/domain/program/nodes/builtin/move_node.h	770
include/aubo_caps/domain/program/nodes/builtin/offline_track_node.h	775
include/aubo_caps/domain/program/nodes/builtin/pallet_node.h	777
include/aubo_caps/domain/program/nodes/builtin/popup_node.h	779
include/aubo_caps/domain/program/nodes/builtin/program_home_node.h	781
include/aubo_caps/domain/program/nodes/builtin/record_track_node.h	784
include/aubo_caps/domain/program/nodes/builtin/screw_driving_node.h	786
include/aubo_caps/domain/program/nodes/builtin/script_node.h	788
include/aubo_caps/domain/program/nodes/builtin/seek_node.h	790
include/aubo_caps/domain/program/nodes/builtin/set_node.h	792
include/aubo_caps/domain/program/nodes/builtin/set_payload_node.h	796
include/aubo_caps/domain/program/nodes/builtin/sub_prog_node.h	799
include/aubo_caps/domain/program/nodes/builtin/switch_node.h	801

include/aubo_caps/domain/program/nodes/builtin/thread_node.h	803
include/aubo_caps/domain/program/nodes/builtin/timer_node.h	805
include/aubo_caps/domain/program/nodes/builtin/until_node.h	807
include/aubo_caps/domain/program/nodes/builtin/visual_localization_program_node.h	813
include/aubo_caps/domain/program/nodes/builtin/wait_node.h	815
include/aubo_caps/domain/program/nodes/builtin/waypoint_node.h	819
include/aubo_caps/domain/program/structure/tree_node.h	837
include/aubo_caps/domain/script/script_writer.h	842
include/aubo_caps/domain/tcp/tcp.h	854
include/aubo_caps/domain/tcp/tcp_contribution_model.h	857
include/aubo_caps/domain/tcp/tcp_model.h	860
include/aubo_caps/domain/undoredo/undo_redo_manager.h	863
include/aubo_caps/domain/user_interface/input_validator.h	866
include/aubo_caps/domain/user_interface/keyboard_manager.h	869
include/aubo_caps/domain/user_interface/popup_manager.h	871
include/aubo_caps/domain/user_interface/render_interface.h	873
include/aubo_caps/domain/user_interface/user_interaction.h	883
include/aubo_caps/domain/user_interface/user_interface_api.h	887
include/aubo_caps/domain/value/value_factory_provider.h	891
include/aubo_caps/domain/value/expression/expression.h	889
include/aubo_caps/domain/variable/feature_variable.h	893
include/aubo_caps/domain/variable/global_variable.h	896
include/aubo_caps/domain/variable/persisted_variable.h	898
include/aubo_caps/domain/variable/variable.h	900
include/aubo_caps/domain/variable/variable_factory.h	903
include/aubo_caps/domain/variable/variable_model.h	906
include/aubo_caps/domain/waypoint/waypoint.h	909
include/aubo_caps/domain/waypoint/waypoint_model.h	911

Chapter 8

专题文档

8.1 ApplicationApi(应用程序)

应用程序接口提供对各种不同应用程序和上下文相关的功能与服务的访问。这包括 AuboScope 内部已有的功能，以及在 AuboScope 中创建要使用的额外模型元素。

类

- class `arcs::aubo_scope::DeviceManager`
设备管理器设备管理器的基础接口。
- class `arcs::aubo_scope::FeatureModel`
特征模型提供从当前机器人安装中获取特征的方法。
- class `arcs::aubo_scope::IoModel`
IO 模型提供从当前机器人安装中获取 *I/O* 的方法。
- class `arcs::aubo_scope::ModbusIo`
Modbus IO 接口此接口提供对 *MODBUS I/O* 的支持。
- class `arcs::aubo_scope::Profinet`
PROFINET IO 接口此接口提供对 *PROFINET I/O* 的支持。
- class `arcs::aubo_scope::PayloadModel`
获取负载参数API 接口
- class `arcs::aubo_scope::TcpModel`
TCP 模型此接口提供对 *AuboScope* 中当前可用的 *TCP* 的访问。
- class `arcs::aubo_scope::ValueFactory`
值工厂接口
- class `arcs::aubo_scope::Variable`
变量接口表示 *AuboScope* 中的变量。变量可用于在程序中存储和传递数据。
- class `arcs::aubo_scope::VariableFactory`
- class `arcs::aubo_scope::VariableModel`
变量模型提供返回系统中当前已注册变量的方法。变量在存储到 *DataModel* 实例中或用于配置内置 *AuboScope* 程序节点时即被注册。

函数

- `IoModelPtr arcs::aubo_scope::ApplicationApi::getIoModel ()`
IoModel 获取 *IO* 模型接口，用于操作机器人的输入和输出。
- `FeatureModelPtr arcs::aubo_scope::ApplicationApi::getFeatureModel ()`
FeatureModel 获取特征模型接口，用于操作当前安装中的特征。
- `TcpModelPtr arcs::aubo_scope::ApplicationApi::getTcpModel ()`
TcpModel 获取 *TCP* 模型接口，用于访问当前安装中的 *TCP*。
- `PayloadModelPtr arcs::aubo_scope::ApplicationApi::getPayloadModel ()`
PayloadModel 获取负载模型接口，用于访问当前 *AuboScope* 安装中的负载。

- VariableModelPtr `arcs::aubo_scope::ApplicationApi::getVariableModel ()`
VariableModel 获取变量模型接口，用于操作变量。
- ValueFactoryPtr `arcs::aubo_scope::ApplicationApi::getValueFactory ()`
ValueFactory 获取值工厂接口，用于访问能够创建值对象的各种工厂。
- DeviceManagerPtr `arcs::aubo_scope::ApplicationApi::getDeviceManager ()`
DeviceManager 此方法用于获取指定的*DeviceManager*实例。

8.1.1 详细描述

应用程序接口提供对各种不同应用程序和上下文相关的功能与服务的访问。这包括 AuboScope 内部已有的功能，以及在 AuboScope 中创建要使用的额外模型元素。

8.1.2 函数说明

getDeviceManager()

DeviceManagerPtr `arcs::aubo_scope::ApplicationApi::getDeviceManager ()`
DeviceManager 此方法用于获取指定的*DeviceManager*实例。

参数

<i>deviceManagerClass</i>	要返回的设备管理器类。
<i>< T ></i>	要返回的类的泛型类型。

返回

用于注册和操作特定设备的接口。

getFeatureModel()

FeatureModelPtr `arcs::aubo_scope::ApplicationApi::getFeatureModel ()`
FeatureModel 获取特征模型接口，用于操作当前安装中的特征。

返回

用于操作当前安装中特征的接口。

getIoModel()

IoModelPtr `arcs::aubo_scope::ApplicationApi::getIoModel ()`
IoModel 获取IO 模型接口，用于操作机器人的输入和输出。

返回

用于操作机器人输入和输出的接口。

getPayloadModel()

PayloadModelPtr `arcs::aubo_scope::ApplicationApi::getPayloadModel ()`
PayloadModel 获取负载模型接口，用于访问当前AuboScope 安装中的负载。

返回

用于访问当前AuboScope 安装中负载的接口。

getTcpModel()

TcpModelPtr arcs::aubo_scope::ApplicationApi::getTcpModel ()
[TcpModel](#) 获取TCP 模型接口，用于访问当前安装中的TCP。

返回

用于访问当前安装中TCP 的接口。

getValueFactory()

ValueFactoryPtr arcs::aubo_scope::ApplicationApi::getValueFactory ()
[ValueFactory](#) 获取值工厂接口，用于访问能够创建值对象的各种工厂。

返回

用于访问能够创建值对象的各种工厂的接口。

getVariableModel()

VariableModelPtr arcs::aubo_scope::ApplicationApi::getVariableModel ()
[VariableModel](#) 获取变量模型接口，用于操作变量。

返回

用于操作变量的接口。

8.2 InstallationApi(安装)

安装接口提供与安装相关的功能和服务访问。这包括 AuboScope 内部已有的功能，以及在 AuboScope 中创建要使用的额外模型元素。

类

- class [arcs::aubo_scope::FeatureContributionModel](#)
特征贡献模型此接口提供在 *AuboScope* 中添加、更新和删除特征的功能。
- class [arcs::aubo_scope::FunctionModel](#)
函数模型此接口提供在表达式编辑器中添加和删除脚本函数的方法，该编辑器可从赋值和 *If* 程序节点等位置访问。
- class [arcs::aubo_scope::Payload](#)
获取负载参数API 接口
- class [arcs::aubo_scope::PayloadContributionModel](#)
动态负载管理
- class [arcs::aubo_scope::TCP](#)
[TCP](#) 接口此接口表示 *AuboScope* 中的一个 *TCP*（工具中心点）。可以是用户定义的 *TCP*，也可以是由 *AuboCap* 添加的 *TCP*。
- class [arcs::aubo_scope::TcpContributionModel](#)
[TCP](#) 贡献模型此接口提供在 *AuboScope* 中添加、更新和删除 [TCP](#) 的功能。

函数

- FunctionModelPtr arcs::aubo_scope::InstallationApi::getFunctionModel ()
获取向 *AuboScope* 添加脚本功能的接口
- TcpContributionModelPtr arcs::aubo_scope::InstallationApi::getTcpContributionModel ()
获取向 *AuboScope* 贡献 *TCP* 的接口
- FeatureContributionModelPtr arcs::aubo_scope::InstallationApi::getFeatureContributionModel ()
获取向 *AuboScope* 贡献特征的接口
- PayloadContributionModelPtr arcs::aubo_scope::InstallationApi::getPayloadContributionModel ()

8.2.1 详细描述

安装接口提供与安装相关的功能和服务访问。这包括 AuboScope 内部已有的功能, 以及在 AuboScope 中创建要使用的额外模型元素。

8.2.2 函数说明

getFeatureContributionModel()

```
FeatureContributionModelPtr arcs::aubo_scope::InstallationApi::getFeatureContributionModel ()
```

获取向AuboScope 贡献特征的接口

返回

用于贡献特征的接口。

getFunctionModel()

```
FunctionModelPtr arcs::aubo_scope::InstallationApi::getFunctionModel ()
```

获取向AuboScope 添加脚本功能的接口

返回

用于添加脚本功能的接口。

getPayloadContributionModel()

```
PayloadContributionModelPtr arcs::aubo_scope::InstallationApi::getPayloadContributionModel ()
```

获取向AuboScope 安装贡献负载的接口。

返回

用于贡献负载的接口。

getTcpContributionModel()

```
TcpContributionModelPtr arcs::aubo_scope::InstallationApi::getTcpContributionModel ()
```

获取向AuboScope 贡献TCP 的接口

返回

用于贡献TCP 的接口。

8.3 ProgramApi(程序)

程序接口提供与程序相关的功能和服务访问。这包括 AuboScope 内部已有的功能, 以及在 AuboScope 中创建要使用的额外模型元素。

类

- class [arcs::aubo_scope::ProgramModel](#)
程序模型
- class [arcs::aubo_scope::UndoRedoManager](#)
撤销/重做管理器此接口用于记录终端用户在 *AuboCap* 程序节点贡献画面中触发的更改 (例如, 点击按钮时)。应在 *AuboCap* 程序节点中使用, 以支持 *AuboScope* 中的撤销/重做功能。
- class [arcs::aubo_scope::Waypoint](#)
路点接口此接口组合了定义机器人位置的参数。
- class [arcs::aubo_scope::WaypointModel](#)
路点模型提供从当前机器人获取路点的方法。

函数

- ProgramModelPtr [arcs::aubo_scope::ProgramApi::getProgramModel \(\)](#)
[ProgramModel](#)
- WaypointModelPtr [arcs::aubo_scope::ProgramApi::getWaypointModel \(\)](#)
[WaypointModel](#)
- UndoRedoManagerPtr [arcs::aubo_scope::ProgramApi::getUndoRedoManager \(\)](#)
[UndoRedoManager](#)

8.3.1 详细描述

程序接口提供与程序相关的功能和服务访问。这包括 AuboScope 内部已有的功能, 以及在 AuboScope 中创建要使用的额外模型元素。

8.3.2 函数说明

getProgramModel()

ProgramModelPtr [arcs::aubo_scope::ProgramApi::getProgramModel \(\)](#)
[ProgramModel](#)

返回

用于操作程序节点子树的接口

getUndoRedoManager()

UndoRedoManagerPtr [arcs::aubo_scope::ProgramApi::getUndoRedoManager \(\)](#)
[UndoRedoManager](#)

返回

用于记录程序节点贡献可撤销更改的接口

getWaypointModel()

WaypointModelPtr [arcs::aubo_scope::ProgramApi::getWaypointModel \(\)](#)
[WaypointModel](#)

返回

用于操作路点的接口

8.4 SystemApi(系统信息)

系统信息接口提供对 AuboScope 软件系统信息的访问, 包括软件版本号、系统语言区域、单位制、机器人序列号, 以及机器人的关节位置范围、速度、加速度和负载等参数信息。

函数

- int [arcs::aubo_scope::SystemApi::getMajorVersion \(\)](#)
获取软件版本号
- int [arcs::aubo_scope::SystemApi::getMinorVersion \(\)](#)
获取软件次版本号
- int [arcs::aubo_scope::SystemApi::getBugfixVersion \(\)](#)
获取软件 *Bug* 修复版本号
- int [arcs::aubo_scope::SystemApi::getBuildNumber \(\)](#)
获取软件构建号

- `std::string arcs::aubo_scope::SystemApi::getLocale ()`
返回系统的`Locale`语言区域设置。
- `std::string arcs::aubo_scope::SystemApi::getLocaleForProgrammingLanguage ()`
返回系统中编程语言使用的`Locale`语言区域设置。
- `UnitType arcs::aubo_scope::SystemApi::getUnitType ()`
以枚举类型返回系统的单位制。
- `std::string arcs::aubo_scope::SystemApi::getSerialNumber ()`
获取机器人的序列号
- `bool arcs::aubo_scope::SystemApi::isRealRobot ()`
检查是否控制真实机器人
- `std::vector< double > arcs::aubo_scope::SystemApi::getJointMaxPositions ()`
获取机器人各关节的最大位置
- `std::vector< double > arcs::aubo_scope::SystemApi::getJointMinPositions ()`
获取机器人各关节的最小位置
- `double arcs::aubo_scope::SystemApi::getJointDefaultVelocity ()`
获取关节默认速度
- `double arcs::aubo_scope::SystemApi::getJointDefaultAcceleration ()`
获取关节默认加速度
- `std::vector< double > arcs::aubo_scope::SystemApi::getJointMaxVelocity ()`
获取各关节最大速度
- `std::vector< double > arcs::aubo_scope::SystemApi::getJointMaxAcceleration ()`
获取各关节最大加速度
- `double arcs::aubo_scope::SystemApi::getTcpDefaultVelocity ()`
获取`TCP` 默认速度
- `double arcs::aubo_scope::SystemApi::getTcpDefaultAcceleration ()`
获取`TCP` 默认加速度
- `double arcs::aubo_scope::SystemApi::getTcpMaxVelocity ()`
获取`TCP` 最大速度
- `double arcs::aubo_scope::SystemApi::getTcpMaxAcceleration ()`
获取`TCP` 最大加速度
- `double arcs::aubo_scope::SystemApi::getTcpMaxDistance ()`
获取`TCP` 最大移动距离
- `double arcs::aubo_scope::SystemApi::getMaxPayloadMass ()`
获取最大负载质量
- `double arcs::aubo_scope::SystemApi::getMaxPayloadDistance ()`
获取最大负载距离
- `std::vector< double > arcs::aubo_scope::SystemApi::getHomeDefaultPosition ()`
获取默认`Home` 位置
- `std::vector< double > arcs::aubo_scope::SystemApi::getHomePackagePosition ()`
获取包装`Home` 位置
- `std::unordered_map< std::string, std::vector< double > > arcs::aubo_scope::SystemApi::getRobotDhParams (bool is_real)`
获取机器人`DH` 参数
- `IoConfigure arcs::aubo_scope::SystemApi::getToolDigitalIOConfigure ()`
获取工具端数字`IO` 配置
- `IoConfigure arcs::aubo_scope::SystemApi::getStandardDigitalInputConfigure ()`
获取标准数字输入配置
- `IoConfigure arcs::aubo_scope::SystemApi::getConfigurableDigitalInputConfigure ()`
获取可配置数字输入配置
- `IoConfigure arcs::aubo_scope::SystemApi::getStandardDigitalOutputConfigure ()`
获取标准数字输出配置

- `IoConfigure arcs::aubo_scope::SystemApi::getConfigurableDigitalOutputConfigure ()`
获取可配置数字输出配置
- `IoConfigure arcs::aubo_scope::SystemApi::getStandardAnalogInputConfigure ()`
获取标准模拟输入配置
- `IoConfigure arcs::aubo_scope::SystemApi::getToolAnalogInputConfigure ()`
获取工具端模拟输入配置
- `IoConfigure arcs::aubo_scope::SystemApi::getStandardAnalogOutputConfigure ()`
获取标准模拟输出配置
- `IoConfigure arcs::aubo_scope::SystemApi::getToolAnalogOutputConfigure ()`
获取工具端模拟输出配置
- `IoConfigure arcs::aubo_scope::SystemApi::getStaticLinkInputConfigure ()`
获取静态链路输入配置
- `IoConfigure arcs::aubo_scope::SystemApi::getStaticLinkOutputConfigure ()`
获取静态链路输出配置
- `IoConfigure arcs::aubo_scope::SystemApi::getENCConfigure ()`
获取编码器配置

8.4.1 详细描述

系统信息接口提供对 AuboScope 软件系统信息的访问，包括软件版本号、系统语言区域、单位制、机器人序列号，以及机器人的关节位置范围、速度、加速度和负载等参数信息。

8.4.2 函数说明

getBugfixVersion()

```
int arcs::aubo_scope::SystemApi::getBugfixVersion ()
    获取软件Bug 修复版本号
```

返回

Bug 修复版本号。

getBuildNumber()

```
int arcs::aubo_scope::SystemApi::getBuildNumber ()
    获取软件构建号
```

返回

构建号。

getConfigurableDigitalInputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getConfigurableDigitalInputConfigure ()
    获取可配置数字输入配置
```

返回

可配置数字输入配置。

getConfigurableDigitalOutputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getConfigurableDigitalOutputConfigure ()
    获取可配置数字输出配置
```

返回

可配置数字输出配置。

getENCConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getENCConfigure ()
```

获取编码器配置

返回

编码器配置。

getHomeDefaultPosition()

```
std::vector< double > arcs::aubo_scope::SystemApi::getHomeDefaultPosition ()
```

获取默认Home 位置

返回

默认Home 位置关节角度。

getHomePackagePosition()

```
std::vector< double > arcs::aubo_scope::SystemApi::getHomePackagePosition ()
```

获取包装Home 位置

返回

包装Home 位置关节角度。

getJointDefaultAcceleration()

```
double arcs::aubo_scope::SystemApi::getJointDefaultAcceleration ()
```

获取关节默认加速度

返回

关节默认加速度。

getJointDefaultVelocity()

```
double arcs::aubo_scope::SystemApi::getJointDefaultVelocity ()
```

获取关节默认速度

返回

关节默认速度。

getJointMaxAcceleration()

```
std::vector< double > arcs::aubo_scope::SystemApi::getJointMaxAcceleration ()
```

获取各关节最大加速度

返回

各关节最大加速度。

getJointMaxPositions()

```
std::vector< double > arcs::aubo_scope::SystemApi::getJointMaxPositions ()
```

获取机器人各关节的最大位置

返回

各关节最大位置，单位为弧度。

getJointMaxVelocity()

```
std::vector< double > arcs::aubo_scope::SystemApi::getJointMaxVelocity ()
```

获取各关节最大速度

返回

各关节最大速度。

getJointMinPositions()

```
std::vector< double > arcs::aubo_scope::SystemApi::getJointMinPositions ()
```

获取机器人各关节的最小位置

返回

各关节最小位置，单位为弧度。

getLocale()

```
std::string arcs::aubo_scope::SystemApi::getLocale ()
```

返回系统的Locale语言区域设置。

返回

语言区域。

getLocaleForProgrammingLanguage()

```
std::string arcs::aubo_scope::SystemApi::getLocaleForProgrammingLanguage ()
```

返回系统中编程语言使用的Locale语言区域设置。

返回

语言区域。

getMajorVersion()

```
int arcs::aubo_scope::SystemApi::getMajorVersion ()
```

获取软件版本号

返回

版本号。

getMaxPayloadDistance()

```
double arcs::aubo_scope::SystemApi::getMaxPayloadDistance ()
```

获取最大负载距离

返回

最大负载距离。

getMaxPayloadMass()

```
double arcs::aubo_scope::SystemApi::getMaxPayloadMass ()
```

获取最大负载质量

返回

最大负载质量。

getMinorVersion()

```
int arcs::aubo_scope::SystemApi::getMinorVersion ()
```

获取软件次版本号

返回

次版本号。

getRobotDhParams()

```
std::unordered_map< std::string, std::vector< double > > arcs::aubo_scope::SystemApi::getRobotDhParams (
    bool is_real)
    获取机器人DH 参数
```

参数

<i>is_real</i>	是否获取真实参数。

返回

DH 参数表。

getSerialNumber()

```
std::string arcs::aubo_scope::SystemApi::getSerialNumber ()
```

获取机器人的序列号

返回

序列号。

getStandardAnalogInputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getStandardAnalogInputConfigure ()
```

获取标准模拟输入配置

返回

标准模拟输入配置。

getStandardAnalogOutputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getStandardAnalogOutputConfigure ()
```

获取标准模拟输出配置

返回

标准模拟输出配置。

getStandardDigitalInputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getStandardDigitalInputConfigure ()
```

获取标准数字输入配置

返回

标准数字输入配置。

getStandardDigitalOutputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getStandardDigitalOutputConfigure ()
```

获取标准数字输出配置

返回

标准数字输出配置。

getStaticLinkInputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getStaticLinkInputConfigure ()
```

获取静态链路输入配置

返回

静态链路输入配置。

getStaticLinkOutputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getStaticLinkOutputConfigure ()
```

获取静态链路输出配置

返回

静态链路输出配置。

getTcpDefaultAcceleration()

```
double arcs::aubo_scope::SystemApi::getTcpDefaultAcceleration ()
```

获取TCP 默认加速度

返回

TCP 默认加速度。

getTcpDefaultVelocity()

```
double arcs::aubo_scope::SystemApi::getTcpDefaultVelocity ()
```

获取TCP 默认速度

返回

TCP 默认速度。

getTcpMaxAcceleration()

```
double arcs::aubo_scope::SystemApi::getTcpMaxAcceleration ()
```

获取TCP 最大加速度

返回

TCP 最大加速度。

getTcpMaxDistance()

```
double arcs::aubo_scope::SystemApi::getTcpMaxDistance ()
```

获取TCP 最大移动距离

返回

TCP 最大距离。

getTcpMaxVelocity()

```
double arcs::aubo_scope::SystemApi::getTcpMaxVelocity ()  
获取TCP 最大速度
```

返回

TCP 最大速度。

getToolAnalogInputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getToolAnalogInputConfigure ()  
获取工具端模拟输入配置
```

返回

工具端模拟输入配置。

getToolAnalogOutputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getToolAnalogOutputConfigure ()  
获取工具端模拟输出配置
```

返回

工具端模拟输出配置。

getToolDigitalIOConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getToolDigitalIOConfigure ()  
获取工具端数字IO 配置
```

返回

工具端数字IO 配置。

getUnitType()

```
UnitType arcs::aubo_scope::SystemApi::getUnitType ()  
以枚举类型返回系统的单位制。
```

返回

单位制枚举。

isRealRobot()

```
bool arcs::aubo_scope::SystemApi::isRealRobot ()  
检查是否控制真实机器人
```

返回

true 表示控制真实物理机器人，false 表示处于仿真模式。

8.5 UserInterfaceApi(用户界面)

用户界面接口提供对 AuboScope 中与用户界面和终端用户交互相关的功能和服务访问。

类

- class [arcs::aubo_scope::RenderInterface](#)
渲染接口提供 3D 渲染相关的显示选项控制，包括地面、机器人模型、坐标系、轨迹和点等元素的显示/隐藏。
- class [arcs::aubo_scope::UserInteraction](#)
用户交互接口此接口提供向终端用户请求输入或操作的功能。

函数

- `UserInteractionPtr` [arcs::aubo_scope::UserInterfaceApi::getUserInteraction \(\)](#)
[UserInteraction](#)

8.5.1 详细描述

用户界面接口提供对 AuboScope 中与用户界面和终端用户交互相关的功能和服务访问。

8.5.2 函数说明

`getUserInteraction()`

```
UserInteractionPtr arcs::aubo_scope::UserInterfaceApi::getUserInteraction ()  
UserInteraction
```

返回

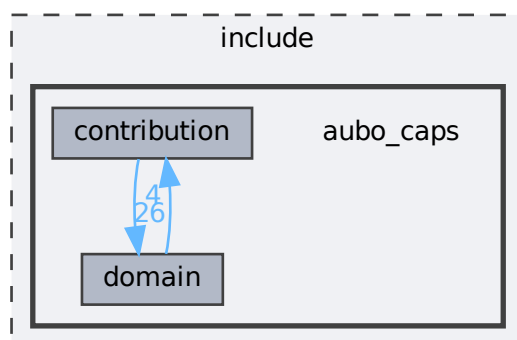
提供向终端用户请求输入和操作功能的接口

Chapter 9

目录说明

9.1 include/aubo_caps 目录参考

aubo_caps 的目录依赖关系图



目录

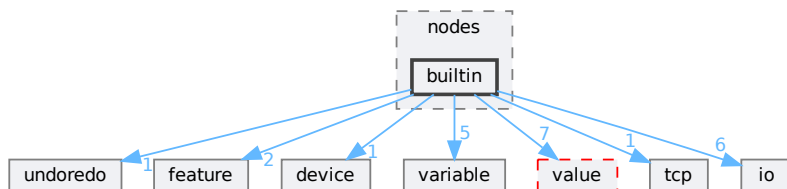
- 目录 [contribution](#)
- 目录 [domain](#)

文件

- 文件 [class_forward.h](#)

9.2 include/aubo__caps/domain/program/nodes/builtin 目录参考

builtin 的目录依赖关系图

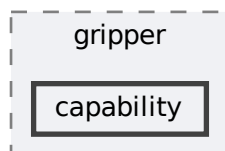


文件

- 文件 [assembly_node.h](#)
- 文件 [assignment_node.h](#)
- 文件 [break_node.h](#)
- 文件 [call_node.h](#)
- 文件 [case_node.h](#)
- 文件 [circle_move_node.h](#)
- 文件 [comment_node.h](#)
- 文件 [conveyor_tracking_node.h](#)
- 文件 [default_case_node.h](#)
- 文件 [direction_node.h](#)
- 文件 [else_if_node.h](#)
- 文件 [else_node.h](#)
- 文件 [folder_node.h](#)
- 文件 [force_node.h](#)
- 文件 [force_option_node.h](#)
- 文件 [gripper_node.h](#)
- 文件 [halt_node.h](#)
- 文件 [if_node.h](#)
- 文件 [loop_node.h](#)
- 文件 [move_node.h](#)
- 文件 [offline_track_node.h](#)
- 文件 [pallet_node.h](#)
- 文件 [popup_node.h](#)
- 文件 [program_home_node.h](#)
- 文件 [record_track_node.h](#)
- 文件 [screw_driving_node.h](#)
- 文件 [script_node.h](#)
- 文件 [seek_node.h](#)
- 文件 [set_node.h](#)
- 文件 [set_payload_node.h](#)
- 文件 [sub_prog_node.h](#)
- 文件 [switch_node.h](#)
- 文件 [thread_node.h](#)
- 文件 [timer_node.h](#)
- 文件 [until_node.h](#)
- 文件 [visual_localization_program_node.h](#)
- 文件 [wait_node.h](#)
- 文件 [waypoint_node.h](#)

9.3 include/aubo_caps/contribution/driver/gripper/capability 目录参考

capability 的目录依赖关系图

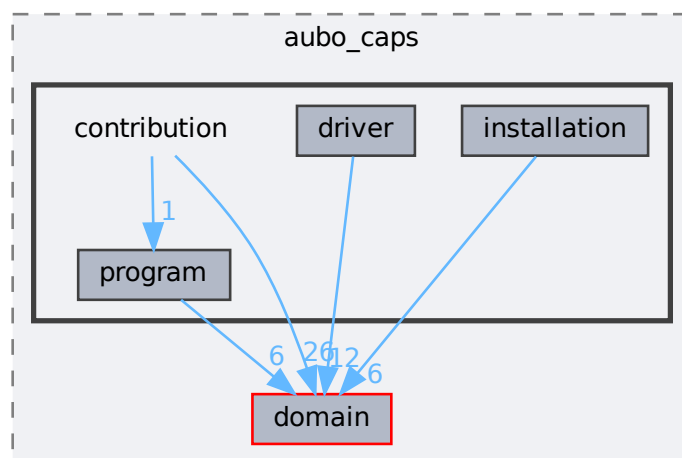


文件

- 文件 [grip_force_capability.h](#)
- 文件 [grip_vacuum_capability.h](#)
- 文件 [registered_capabilities.h](#)
- 文件 [speed_capability.h](#)
- 文件 [width_capability.h](#)

9.4 include/aubo_caps/contribution 目录参考

contribution 的目录依赖关系图



目录

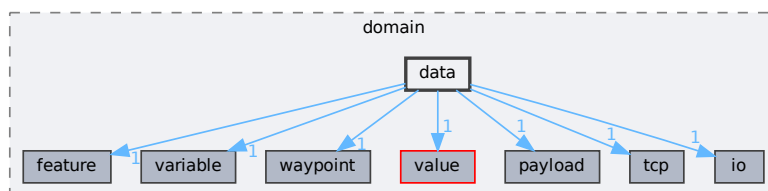
- 目录 [driver](#)
- 目录 [installation](#)
- 目录 [program](#)

文件

- 文件 [contribution_provider.h](#)
- 文件 [view_api_provider.h](#)

9.5 include/aubo_caps/domain/data 目录参考

data 的目录依赖关系图

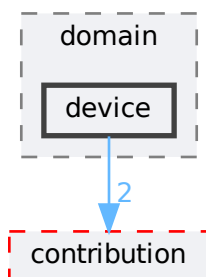


文件

- 文件 [data_model.h](#)

9.6 include/aubo_caps/domain/device 目录参考

device 的目录依赖关系图



文件

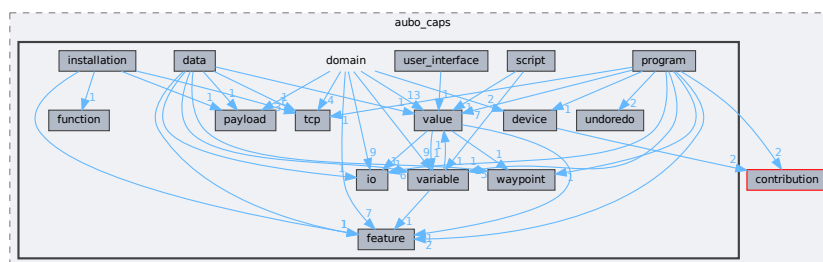
- 文件 [device.h](#)
- 文件 [device_manager.h](#)

- 文件 [device_registration_manager.h](#)
- 文件 [gripper_device.h](#)

9.7 doc 目录参考

9.8 include/aubo_caps/domain 目录参考

domain 的目录依赖关系图



目录

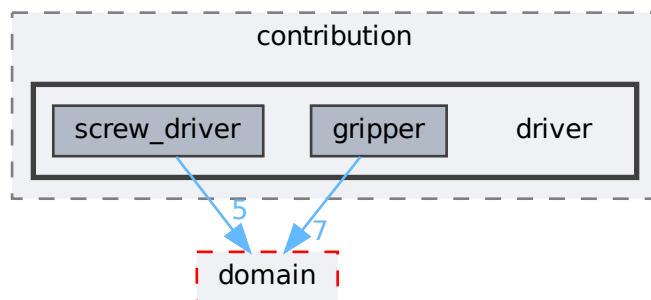
- 目录 [data](#)
- 目录 [device](#)
- 目录 [feature](#)
- 目录 [function](#)
- 目录 [installation](#)
- 目录 [io](#)
- 目录 [payload](#)
- 目录 [program](#)
- 目录 [script](#)
- 目录 [tcp](#)
- 目录 [undoredo](#)
- 目录 [user_interface](#)
- 目录 [value](#)
- 目录 [variable](#)
- 目录 [waypoint](#)

文件

- 文件 [application_api.h](#)
- 文件 [extension_info.h](#)
- 文件 [system_api.h](#)

9.9 include/aubo_caps/contribution/driver 目录参考

driver 的目录依赖关系图

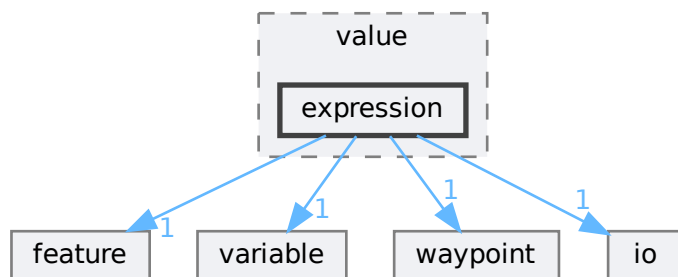


目录

- 目录 [gripper](#)
- 目录 [screw_driver](#)

9.10 include/aubo_caps/domain/value/expression 目录参考

expression 的目录依赖关系图

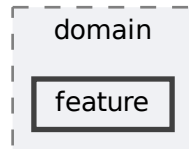


文件

- 文件 [expression.h](#)

9.11 include/aubo_caps/domain/feature 目录参考

feature 的目录依赖关系图

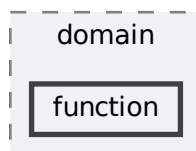


文件

- 文件 [base_feature.h](#)
- 文件 [feature.h](#)
- 文件 [feature_contribution_model.h](#)
- 文件 [feature_line.h](#)
- 文件 [feature_model.h](#)
- 文件 [feature_plane.h](#)
- 文件 [feature_point.h](#)
- 文件 [feature_tool.h](#)

9.12 include/aubo_caps/domain/function 目录参考

function 的目录依赖关系图

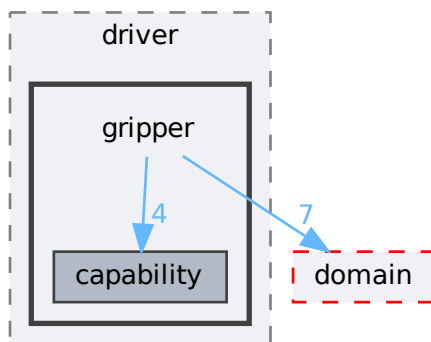


文件

- 文件 [function.h](#)
- 文件 [function_model.h](#)

9.13 include/aubo_caps/contribution/driver/gripper 目录参考

gripper 的目录依赖关系图



目录

- 目录 [capability](#)

文件

- 文件 [grip_action_parameters.h](#)
- 文件 [gripper_api_provider.h](#)
- 文件 [gripper_configuration.h](#)
- 文件 [gripper_contribution.h](#)
- 文件 [release_action_parameters.h](#)

9.14 include 目录参考

include 的目录依赖关系图

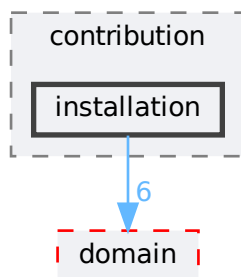


目录

- 目录 [aubo_caps](#)

9.15 include/aubo_caps/contribution/installation 目录参考

installation 的目录依赖关系图

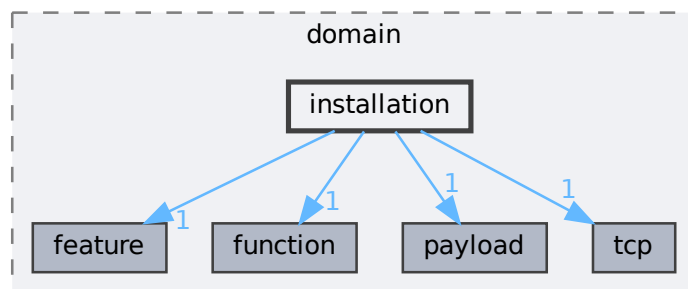


文件

- 文件 [installation_api_provider.h](#)
- 文件 [installation_contribution_configuration.h](#)
- 文件 [installation_creation_context.h](#)
- 文件 [installation_node_contribution.h](#)
- 文件 [installation_node_service.h](#)
- 文件 [installation_node_view.h](#)

9.16 include/aubo_caps/domain/installation 目录参考

installation 的目录依赖关系图

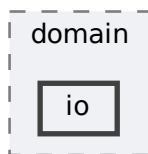


文件

- 文件 [installation_api.h](#)

9.17 include/aubo_caps/domain/io 目录参考

io 的目录依赖关系图

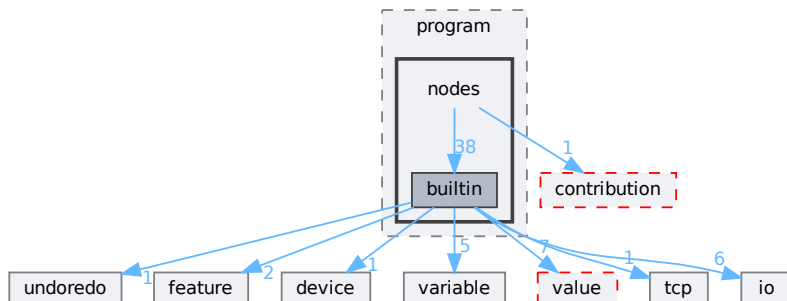


文件

- 文件 [analog_io.h](#)
- 文件 [boolean_register.h](#)
- 文件 [digital_io.h](#)
- 文件 [double_register.h](#)
- 文件 [ethernet_ip.h](#)
- 文件 [integer_register.h](#)
- 文件 [io.h](#)
- 文件 [io_model.h](#)
- 文件 [modbus_io.h](#)
- 文件 [profinet.h](#)
- 文件 [register.h](#)

9.18 include/aubo_caps/domain/program/nodes 目录参考

nodes 的目录依赖关系图



目录

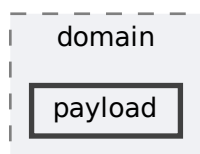
- 目录 [builtin](#)

文件

- 文件 [program_node.h](#)
- 文件 [program_node_factory.h](#)

9.19 include/aubo_caps/domain/payload 目录参考

payload 的目录依赖关系图

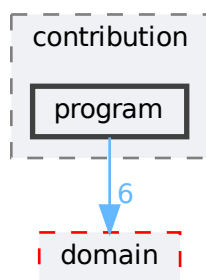


文件

- 文件 [payload.h](#)
- 文件 [payload_contribution_model.h](#)
- 文件 [payload_model.h](#)

9.20 include/aubo_caps/contribution/program 目录参考

program 的目录依赖关系图



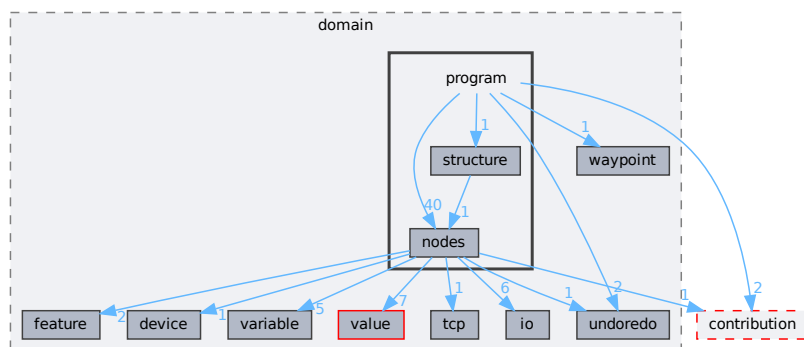
文件

- 文件 [program_api_provider.h](#)
- 文件 [program_contribution_configuration.h](#)
- 文件 [program_creation_context.h](#)
- 文件 [program_node_contribution.h](#)

- 文件 [program_node_service.h](#)
- 文件 [program_node_view.h](#)

9.21 include/aubo_caps/domain/program 目录参考

program 的目录依赖关系图



目录

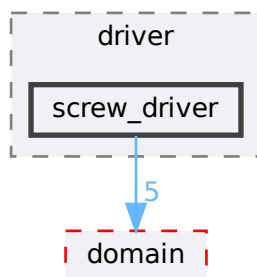
- 目录 [nodes](#)
- 目录 [structure](#)

文件

- 文件 [program_api.h](#)
- 文件 [program_model.h](#)

9.22 include/aubo_caps/contribution/driver/screw_driver 目录参考

screw_driver 的目录依赖关系图

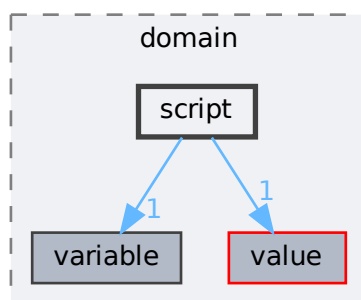


文件

- 文件 [screw_driver_api_provider.h](#)
- 文件 [screw_driver_configuration.h](#)
- 文件 [screw_driver_contribution.h](#)
- 文件 [screw_driver_parameters.h](#)

9.23 include/aubo_caps/domain/script 目录参考

script 的目录依赖关系图

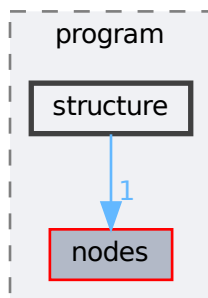


文件

- 文件 [script_writer.h](#)

9.24 include/aubo_caps/domain/program/structure 目录参考

structure 的目录依赖关系图

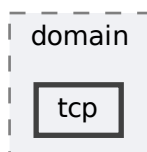


文件

- 文件 [tree_node.h](#)

9.25 include/aubo__caps/domain/tcp 目录参考

tcp 的目录依赖关系图

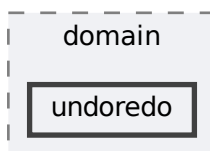


文件

- 文件 [tcp.h](#)
- 文件 [tcp_contribution_model.h](#)
- 文件 [tcp_model.h](#)

9.26 include/aubo__caps/domain/undoredo 目录参考

undoredo 的目录依赖关系图

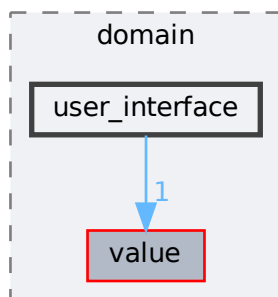


文件

- 文件 [undo_redo_manager.h](#)

9.27 include/aubo_caps/domain/user_interface 目录参考

user_interface 的目录依赖关系图

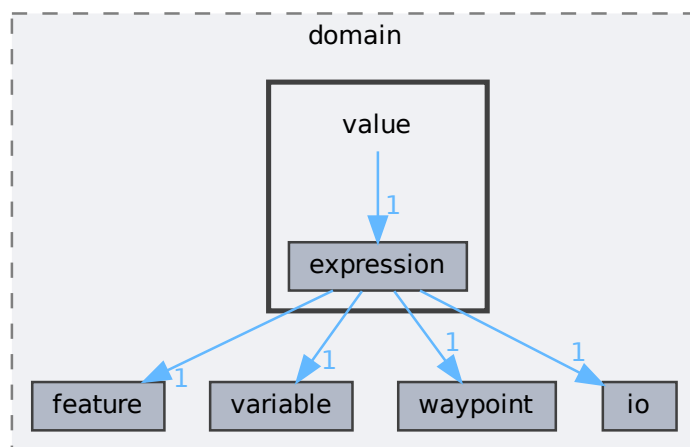


文件

- 文件 [input_validator.h](#)
- 文件 [keyboard_manager.h](#)
- 文件 [popup_manager.h](#)
- 文件 [render_interface.h](#)
- 文件 [user_interaction.h](#)
- 文件 [user_interface_api.h](#)

9.28 include/aubo_caps/domain/value 目录参考

value 的目录依赖关系图



目录

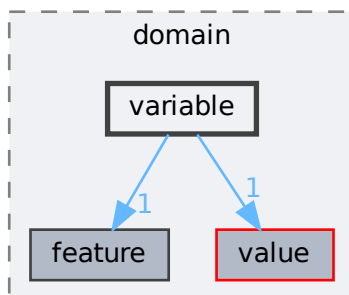
- 目录 [expression](#)

文件

- 文件 [value_factory_provider.h](#)

9.29 include/aubo_caps/domain/variable 目录参考

variable 的目录依赖关系图

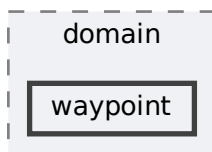


文件

- 文件 [feature_variable.h](#)
- 文件 [global_variable.h](#)
- 文件 [persisted_variable.h](#)
- 文件 [variable.h](#)
- 文件 [variable_factory.h](#)
- 文件 [variable_model.h](#)

9.30 include/aubo_caps/domain/waypoint 目录参考

waypoint 的目录依赖关系图



文件

- 文件 [waypoint.h](#)
- 文件 [waypoint_model.h](#)

Chapter 10

命名空间文档

10.1 arcs 命名空间参考

命名空间

- namespace [aubo_scope](#)
- namespace [aubo_sdk](#)

10.2 arcs::aubo_scope 命名空间参考

类

- class [ContributionProvider](#)
贡献提供者此接口提供对贡献的访问。
- class [GripForceCapability](#)
夹持力能力此接口表示夹爪已注册的力度能力，支持使用用户可配置的力度进行夹取。
- class [GripVacuumCapability](#)
真空夹持能力此接口表示夹爪已注册的真空能力，支持使用用户可配置的真空度进行夹取。
- class [RegisteredCapabilities](#)
- class [SpeedCapability](#)
速度能力此接口表示夹爪已注册的速度能力，支持使用用户可配置的速度进行操作。
- class [WidthCapability](#)
宽度能力此接口表示夹爪已注册的宽度/位置能力，支持移动到用户可配置的位置（打开/关闭到可配置的宽度）。
- class [GripActionParameters](#)
- class [GripperApiProvider](#)
夹爪 API 提供者此接口提供对 *AuboScope* 中与夹爪相关的功能和服务的访问。
- class [GripperConfiguration](#)
This interface provides access to register or setup various properties and capabilities of a gripper.
- class [GripperContribution](#)
夹爪贡献此接口定义了向 *AuboScope* 添加夹爪设备功能的 API。
- class [ReleaseActionParameters](#)
- class [ScrewdriverApiProvider](#)
螺丝刀 API 提供者此接口提供对 *AuboScope* 中与螺丝刀贡献相关的功能和服务的访问。
- class [ProgramSelectionParameters](#)
- class [PrepareToStartScrewdriverParameters](#)
- class [FeedScrewParameters](#)
- class [DriveScrewNotOKParameters](#)
- class [DriveScrewOKParameters](#)
- class [ScrewdriverReadyParameters](#)

This interface provides the parameters relevant for generating script for the Screwdriver Ready feedback operation capability (registered using `ScrewdriverFeedbackCapabilities#registerScrewdriverReadyCapability` (Scripting))

- class `ScrewdriverConfiguration`

This interface provides access to register or setup various properties and capabilities of a screwdriver.

- class `ScrewdriverContribution`

螺丝刀贡献此接口定义了向 `AuboScope` 添加螺丝刀功能的 API。

- class `ScrewdriverProgram`

This abstract class represents a screwdriver program, typically defined on an external control box for the screwdriver.

- class `ScrewdriverParameters`

This interface represents parameters for a screwdriver operation defined/configured by the end user.

- class `InstallationApiProvider`

安装 API 提供者提供与安装节点相关的功能和服务访问。这包括 `AuboScope` 内部已有的功能，以及在 `AuboScope` 中创建要使用的额外模型元素。

- class `InstallationContributionConfiguration`

- class `InstallationCreationContext`

安装创建上下文提供安装节点创建时所在上下文的相关信息。

- class `InstallationNodeContribution`

安装节点贡献安装节点贡献的 API。

- class `InstallationNodeService`

安装节点服务定义了向 `AuboScope` 中定义和添加安装节点及对应界面（用户界面基于 Qt）所需的 API。

- class `InstallationNodeView`

安装节点视图安装节点视图的 API，定义了对应 `InstallationNodeContribution` 的 UI。

- class `ProgramApiProvider`

程序 API 提供者提供与程序节点相关的功能和服务访问。这包括 `AuboScope` 内部已有的功能，以及在 `AuboScope` 中创建要使用的额外模型元素。

- class `ProgramContributionConfiguration`

程序贡献配置提供对安装节点贡献属性进行配置访问。

- class `ProgramCreationContext`

程序创建上下文提供安装节点创建时所在上下文的相关信息。

- class `ProgramNodeContribution`

程序节点贡献定义了向 `AuboScope` 中指定 `AuboCap` 程序节点所需的 API。

- class `ProgramNodeService`

程序节点服务定义了向 `AuboScope` 添加新类型程序节点所需的 API。

- class `ProgramNodeView`

程序节点视图程序节点视图的 API，定义了对应 `ProgramNodeContribution` 的 UI。 `ProgramNodeView` 实例是多个贡献之间的共享实体。

- class `ViewApiProvider`

视图 API 提供者提供与安装或程序节点贡献的视图（UI）相关的功能访问。

- class `ApplicationApi`

- class `DataModel`

数据模型此接口用于存储和管理表示当前配置的数据，例如 `ProgramNodeContribution` 或 `InstallationNodeContribution` 的配置。提供了在字典中添加、删除、获取和修改值的方法。

- class `Device`

设备接口此基础接口表示 `AuboScope` 支持的设备，例如夹爪设备。

- class `DeviceManager`

设备管理器设备管理器的基础接口。

- class `DeviceRegistrationManager`

设备注册管理器设备注册管理器的基础接口。

- class `GripperDevice`

夹爪设备此接口表示 `AuboScope` 中的夹爪设备。

- class `AuboCapInfo`

This interface provides various metadata information about an installed AuboCap, such as the name of the AuboCap.

- class [BaseFeature](#)

This interface represents the Base feature in AuboScope.

- class [Feature](#)

特征接口此基础接口表示 *AuboScope* 中所有可用的特征类型。

- struct [FeatureDefinedFilter](#)
- struct [FeatureJoggableFilter](#)
- class [FeatureContributionModel](#)

特征贡献模型此接口提供在 *AuboScope* 中添加、更新和删除特征的功能。

- class [FeatureLine](#)
- class [FeatureModel](#)

特征模型提供从当前机器人安装中获取特征的方法。

- class [FeaturePlane](#)
- class [FeaturePoint](#)

This interface represents a Point feature in AuboScope

- class [FeatureTool](#)

This interface represents the Tool feature in AuboScope.

- class [Function](#)

函数接口表示在表达式编辑器中显示的函数。

- class [FunctionModel](#)

函数模型此接口提供在表达式编辑器中添加和删除脚本函数的方法，该编辑器可从赋值和 *If* 程序节点等位置访问。

- class [InstallationApi](#)
- class [AnalogIo](#)

模拟 *IO* 接口此接口提供对模拟 *I/O* 的支持。

- struct [AnalogIoFilter](#)
- class [BoolRegister](#)

This interface provides support for bool registers.

- class [DigitalIo](#)

数字 *IO* 接口此接口提供对数字 *I/O* 的支持。

- struct [DigitalIoFilter](#)
- struct [InputIoFilter](#)
- struct [OutputIoFilter](#)
- class [DoubleRegister](#)

This interface provides support for float registers.

- class [EtherNetIP](#)

EtherNet/IP IO 接口此接口提供对 *EtherNet/IP I/O* 的支持。

- struct [EtherNetIPFilter](#)
- class [IntegerRegister](#)

This interface provides support for integer registers.

- class [Io](#)

IO 基础接口这是表示 *AuboScope* 中所有可用 *I/O* 类型的基础接口。

- class [IoModel](#)

IO 模型提供从当前机器人安装中获取 *I/O* 的方法。

- class [ModbusIo](#)

Modbus IO 接口此接口提供对 *MODBUS I/O* 的支持。

- struct [ModbusIoFilter](#)
- class [Profinet](#)

PROFINET IO 接口此接口提供对 *PROFINET I/O* 的支持。

- struct [ProfinetFilter](#)
- class [Register](#)

This interface provides support for registers.

- class [Payload](#)
获取负载参数API 接口
- class [PayloadContributionModel](#)
动态负载管理
- class [PayloadModel](#)
获取负载参数API 接口
- class [AssemblyNode](#)
- class [AssignmentNode](#)
- class [BreakNode](#)
- class [CallNode](#)
- class [CaseNode](#)
- class [CircleMoveNode](#)
- class [CommentNode](#)
- class [ConveyorTrackingNode](#)
- class [DefaultCaseNode](#)
- class [DirectionNode](#)
- class [ElseIfNode](#)
- class [ElseNode](#)
- class [FolderNode](#)
- class [ForceNode](#)
- class [ForceOptionNode](#)
- class [GripperNode](#)
- class [HaltNode](#)
- class [IfNode](#)
- class [LoopNode](#)
- class [MoveNode](#)
- class [OfflineTrackNode](#)
- class [PalletNode](#)
- class [PopupNode](#)
- class [ProgramHomeNode](#)
- class [RecordTrackNode](#)
- class [ScrewdrivingNode](#)
- class [ScriptNode](#)
- class [SeekNode](#)
- class [SetNode](#)
- class [SetPayloadNode](#)
- class [SubProgNode](#)
- class [SwitchNode](#)
- class [ThreadNode](#)
- class [TimerNode](#)
- class [UntilNode](#)
- class [VisualLocalizationProgramNode](#)
- class [WaitNode](#)
- class [WaypointNode](#)
- class [ProgramNode](#)
程序节点此接口表示 *AuboScope* 中的程序节点。
- class [ProgramNodeFactory](#)
程序节点工厂此接口提供创建不同类型程序节点的方法。
- class [ProgramApi](#)
- class [ProgramModel](#)
程序模型
- class [TreeNode](#)

- 树节点此接口表示程序树中的一个节点，可用于构建以 *AuboCap* 程序节点为根的子树。
- class [ScriptWriter](#)
脚本写入器此接口提供生成脚本代码的支持。
 - struct [IoConfigure](#)
 - class [SystemApi](#)
 - class [TCP](#)
TCP 接口此接口表示 *AuboScope* 中的一个 *TCP* (工具中心点)。可以是用户定义的 *TCP*，也可以是由 *AuboCap* 添加的 *TCP*。
 - class [TcpContributionModel](#)
TCP 贡献模型此接口提供在 *AuboScope* 中添加、更新和删除 *TCP* 的功能。
 - class [TcpModel](#)
TCP 模型此接口提供对 *AuboScope* 中当前可用的 *TCP* 的访问。
 - class [UndoRedoManager](#)
撤销/重做管理器此接口用于记录终端用户在 *AuboCap* 程序节点贡献画面中触发的更改（例如，点击按钮时）。应在 *AuboCap* 程序节点中使用，以支持 *AuboScope* 中的撤销/重做功能。
 - struct [InputValidator](#)
输入验证器表示由 *InputValidationFactory* 创建的输入验证器接口。该工厂提供一组标准输入验证器，可用于验证用户使用虚拟键盘/小键盘输入的内容。
 - struct [IntegerRangeValidator](#)
 - struct [UIntegerRangeValidator](#)
 - struct [DoubleRangeValidator](#)
 - struct [StringLengthValidator](#)
 - class [KeyboardManager](#)
 - struct [DefaultSingletonTraits](#)
 - class [PopupManager](#)
 - class [RenderInterface](#)
渲染接口提供 3D 渲染相关的显示选项控制，包括地面、机器人模型、坐标系、轨迹和点等元素的显示/隐藏。
 - class [UserInteraction](#)
用户交互接口此接口提供向终端用户请求输入或操作的功能。
 - class [UserInterfaceApi](#)
 - class [Expression](#)
表达式表达式可用于配置各种内置程序节点，例如 *If* 和 *Wait* 节点。表达式通常是条件表达式（计算结果为 *true* 或 *false*）。
 - class [ValueFactory](#)
值工厂接口
 - class [FeatureVariable](#)
特征变量在安装中被标记为变量的特征，即使在机器人关闭和重新启动后也会存在。
 - struct [FeatureVariableFilter](#)
 - class [GlobalVariable](#)
Global variables are available everywhere in a program.
 - struct [GlobalVariableFilter](#)
 - class [PersistedVariable](#)
ingroup ApplicationApi This interface represents variables stored with the installation.
 - struct [PersistedVariableFilter](#)
 - class [Variable](#)
变量接口表示 *AuboScope* 中的变量。变量可用于在程序中存储和传递数据。
 - class [VariableFactory](#)
 - class [VariableModel](#)
变量模型提供返回系统中当前已注册变量的方法。变量在存储到 *DataModel* 实例中或用于配置内置 *AuboScope* 程序节点时即被注册。
 - class [Waypoint](#)
路点接口此接口组合了定义机器人位置的参数。
 - class [WaypointModel](#)
路点模型提供从当前机器人获取路点的方法。

类型定义

- `template<class T>`
using `ContributionProviderPtr` = `std::shared_ptr<ContributionProvider<T>>`
- `typedef long WidgetHandle`
- using `RobotPositionCallback`
- using `RobotMovementCallback` = `std::function<void(bool accepted, int error)>`

枚举

- enum `ScrewdriverOperationType` { `TIGHTEN` , `LOOSEN` }
This interface represents the type of screwing operation performed by the screwdriver.
- enum class `UnitType` { `METRIC` , `IMPERIAL` }

函数

- `ARCS_CLASS_FORWARD` (`GripForceCapability`)
- `ARCS_CLASS_FORWARD` (`GripVacuumCapability`)
- `ARCS_CLASS_FORWARD` (`RegisteredCapabilities`)
- `ARCS_CLASS_FORWARD` (`SpeedCapability`)
- `ARCS_CLASS_FORWARD` (`WidthCapability`)
- `ARCS_CLASS_FORWARD` (`GripActionParameters`)
- `ARCS_CLASS_FORWARD` (`GripperApiProvider`)
- `ARCS_CLASS_FORWARD` (`GripperConfiguration`)
- `ARCS_CLASS_FORWARD` (`GripperContribution`)
- `ARCS_CLASS_FORWARD` (`ReleaseActionParameters`)
- `ARCS_CLASS_FORWARD` (`ScrewdriverApiProvider`)
- `ARCS_CLASS_FORWARD` (`ProgramSelectionParameters`)
- `ARCS_CLASS_FORWARD` (`PrepareToStartScrewdriverParameters`)
- `ARCS_CLASS_FORWARD` (`FeedScrewParameters`)
- `ARCS_CLASS_FORWARD` (`DriveScrewNotOKParameters`)
- `ARCS_CLASS_FORWARD` (`DriveScrewOKParameters`)
- `ARCS_CLASS_FORWARD` (`ScrewdriverReadyParameters`)
- `ARCS_CLASS_FORWARD` (`ScrewdriverConfiguration`)
- `ARCS_CLASS_FORWARD` (`ScrewdriverContribution`)
- `ARCS_CLASS_FORWARD` (`ScrewdriverProgram`)
- `ARCS_CLASS_FORWARD` (`ScrewdriverParameters`)
- `ARCS_CLASS_FORWARD` (`InstallationApiProvider`)
- `ARCS_CLASS_FORWARD` (`InstallationContributionConfiguration`)
- `ARCS_CLASS_FORWARD` (`InstallationCreationContext`)
- `ARCS_CLASS_FORWARD` (`InstallationNodeContribution`)
- `ARCS_CLASS_FORWARD` (`InstallationNodeService`)
- `ARCS_CLASS_FORWARD` (`InstallationNodeView`)
- `ARCS_CLASS_FORWARD` (`ProgramApiProvider`)
- `ARCS_CLASS_FORWARD` (`ProgramContributionConfiguration`)
- `ARCS_CLASS_FORWARD` (`ProgramCreationContext`)
- `ARCS_CLASS_FORWARD` (`ProgramNodeContribution`)
- `ARCS_CLASS_FORWARD` (`ProgramNodeService`)
- `ARCS_CLASS_FORWARD` (`ProgramNodeView`)
- `ARCS_CLASS_FORWARD` (`ViewApiProvider`)
- `ARCS_CLASS_FORWARD` (`ApplicationApi`)
- `ARCS_CLASS_FORWARD` (`DataModel`)
- `ARCS_CLASS_FORWARD` (`Device`)
- `ARCS_CLASS_FORWARD` (`DeviceManager`)
- `ARCS_CLASS_FORWARD` (`DeviceRegistrationManager`)
- `ARCS_CLASS_FORWARD` (`GripperDevice`)

- [ARCS_CLASS_FORWARD \(AuboCapInfo\)](#)
- [ARCS_CLASS_FORWARD \(BaseFeature\)](#)
- [ARCS_CLASS_FORWARD \(Feature\)](#)
- [ARCS_CLASS_FORWARD \(FeatureContributionModel\)](#)
- [ARCS_CLASS_FORWARD \(FeatureLine\)](#)
- [ARCS_CLASS_FORWARD \(FeatureModel\)](#)
- [ARCS_CLASS_FORWARD \(FeaturePlane\)](#)
- [ARCS_CLASS_FORWARD \(FeaturePoint\)](#)
- [ARCS_CLASS_FORWARD \(FeatureTool\)](#)
- [ARCS_CLASS_FORWARD \(Function\)](#)
- [ARCS_CLASS_FORWARD \(FunctionModel\)](#)
- [ARCS_CLASS_FORWARD \(InstallationApi\)](#)
- [ARCS_CLASS_FORWARD \(AnalogIo\)](#)
- [ARCS_CLASS_FORWARD \(BoolRegister\)](#)
- [ARCS_CLASS_FORWARD \(DigitalIo\)](#)
- [ARCS_CLASS_FORWARD \(DoubleRegister\)](#)
- [ARCS_CLASS_FORWARD \(EtherNetIP\)](#)
- [ARCS_CLASS_FORWARD \(IntegerRegister\)](#)
- [ARCS_CLASS_FORWARD \(Io\)](#)
- [ARCS_CLASS_FORWARD \(IoModel\)](#)
- [ARCS_CLASS_FORWARD \(ModbusIo\)](#)
- [ARCS_CLASS_FORWARD \(Profinet\)](#)
- [ARCS_CLASS_FORWARD \(Register\)](#)
- [ARCS_CLASS_FORWARD \(Payload\)](#)
- [ARCS_CLASS_FORWARD \(PayloadContributionModel\)](#)
- [ARCS_CLASS_FORWARD \(PayloadModel\)](#)
- [ARCS_CLASS_FORWARD \(AssemblyNode\)](#)
- [ARCS_CLASS_FORWARD \(AssignmentNode\)](#)
- [ARCS_CLASS_FORWARD \(BreakNode\)](#)
- [ARCS_CLASS_FORWARD \(CallNode\)](#)
- [ARCS_CLASS_FORWARD \(CaseNode\)](#)
- [ARCS_CLASS_FORWARD \(CircleMoveNode\)](#)
- [ARCS_CLASS_FORWARD \(CommentNode\)](#)
- [ARCS_CLASS_FORWARD \(ConveyorTrackingNode\)](#)
- [ARCS_CLASS_FORWARD \(DefaultCaseNode\)](#)
- [ARCS_CLASS_FORWARD \(DirectionNode\)](#)
- [ARCS_CLASS_FORWARD \(ElseIfNode\)](#)
- [ARCS_CLASS_FORWARD \(ElseNode\)](#)
- [ARCS_CLASS_FORWARD \(FolderNode\)](#)
- [ARCS_CLASS_FORWARD \(ForceNode\)](#)
- [ARCS_CLASS_FORWARD \(ForceOptionNode\)](#)
- [ARCS_CLASS_FORWARD \(GripperNode\)](#)
- [ARCS_CLASS_FORWARD \(HaltNode\)](#)
- [ARCS_CLASS_FORWARD \(Expression\)](#)
- [ARCS_CLASS_FORWARD \(IfNode\)](#)
- [ARCS_CLASS_FORWARD \(LoopNode\)](#)
- [ARCS_CLASS_FORWARD \(MoveNode\)](#)
- [ARCS_CLASS_FORWARD \(OfflineTrackNode\)](#)
- [ARCS_CLASS_FORWARD \(PalletNode\)](#)
- [ARCS_CLASS_FORWARD \(PopupNode\)](#)
- [ARCS_CLASS_FORWARD \(ProgramHomeNode\)](#)
- [ARCS_CLASS_FORWARD \(RecordTrackNode\)](#)
- [ARCS_CLASS_FORWARD \(ScrewdrivingNode\)](#)
- [ARCS_CLASS_FORWARD \(ScriptNode\)](#)
- [ARCS_CLASS_FORWARD \(SeekNode\)](#)

- [ARCS_CLASS_FORWARD \(SetNode\)](#)
- [ARCS_CLASS_FORWARD \(SetPayloadNode\)](#)
- [ARCS_CLASS_FORWARD \(SubProgNode\)](#)
- [ARCS_CLASS_FORWARD \(SwitchNode\)](#)
- [ARCS_CLASS_FORWARD \(ThreadNode\)](#)
- [ARCS_CLASS_FORWARD \(TimerNode\)](#)
- [ARCS_CLASS_FORWARD \(UntilNode\)](#)
- [ARCS_CLASS_FORWARD \(VisualLocalizationProgramNode\)](#)
- [ARCS_CLASS_FORWARD \(WaitNode\)](#)
- [ARCS_CLASS_FORWARD \(WaypointNode\)](#)
- [ARCS_CLASS_FORWARD \(ProgramNode\)](#)
- [ARCS_CLASS_FORWARD \(ProgramNodeFactory\)](#)
- [ARCS_CLASS_FORWARD \(ProgramApi\)](#)
- [ARCS_CLASS_FORWARD \(ProgramModel\)](#)
- [ARCS_CLASS_FORWARD \(TreeNode\)](#)
- [ARCS_CLASS_FORWARD \(ScriptWriter\)](#)
- [ARCS_CLASS_FORWARD \(SystemApi\)](#)
- [ARCS_CLASS_FORWARD \(TCP\)](#)
- [ARCS_CLASS_FORWARD \(TcpContributionModel\)](#)
- [ARCS_CLASS_FORWARD \(TcpModel\)](#)
- [ARCS_CLASS_FORWARD \(UndoRedoManager\)](#)
- [ARCS_CLASS_FORWARD \(KeyboardManager\)](#)
- [ARCS_CLASS_FORWARD \(PopupManager\)](#)
- [ARCS_CLASS_FORWARD \(RenderInterface\)](#)
- [ARCS_CLASS_FORWARD \(UserInteraction\)](#)
- [ARCS_CLASS_FORWARD \(UserInterfaceApi\)](#)
- [ARCS_CLASS_FORWARD \(ValueFactory\)](#)
- [ARCS_CLASS_FORWARD \(FeatureVariable\)](#)
- [ARCS_CLASS_FORWARD \(GlobalVariable\)](#)
- [ARCS_CLASS_FORWARD \(PersistedVariable\)](#)
- [ARCS_CLASS_FORWARD \(Variable\)](#)
- [ARCS_CLASS_FORWARD \(VariableFactory\)](#)
- [ARCS_CLASS_FORWARD \(VariableModel\)](#)
- [ARCS_CLASS_FORWARD \(Waypoint\)](#)
- [ARCS_CLASS_FORWARD \(WaypointModel\)](#)

10.2.1 类型定义说明

ContributionProviderPtr

```
template<class T>
using arcs::aubo_scope::ContributionProviderPtr = std::shared_ptr<ContributionProvider<T>>
```

在文件 [contribution_provider.h](#) 第 53 行定义.

RobotMovementCallback

```
using arcs::aubo_scope::RobotMovementCallback = std::function<void(bool accepted, int error)>
```

在文件 [user_interaction.h](#) 第 21 行定义.

RobotPositionCallback

```
using arcs::aubo_scope::RobotPositionCallback
```

初始值:

```
std::function<void(
    bool accepted, const std::vector<double> &pose,
    const std::vector<double> &q, const std::vector<double> &offset)>
```

在文件 [user_interaction.h](#) 第 18 行定义.

WidgetHandle

typedef long `arcs::aubo_scope::WidgetHandle`
 在文件 `program_node_view.h` 第 9 行定义.

10.2.2 枚举类型说明

ScrewdriverOperationType

enum `arcs::aubo_scope::ScrewdriverOperationType`
 This interface represents the type of screwing operation performed by the screwdriver.

枚举值

TIGHTEN	The screwdriver operation will follow the screw in a tightening direction (in).
LOOSEN	The screwdriver operation will follow the screw in a loosening direction (out).

在文件 `screw_driver_parameters.h` 第 55 行定义.

UnitType

enum class `arcs::aubo_scope::UnitType` [strong]

枚举值

METRIC	公制 mm Kg
IMPERIAL	英制

在文件 `system_api.h` 第 12 行定义.

10.2.3 函数说明

ARCS_CLASS_FORWARD() [1/130]

`arcs::aubo_scope::ARCS_CLASS_FORWARD (`
 `AnalogIo )`

ARCS_CLASS_FORWARD() [2/130]

`arcs::aubo_scope::ARCS_CLASS_FORWARD (`
 `ApplicationApi )`

ARCS_CLASS_FORWARD() [3/130]

`arcs::aubo_scope::ARCS_CLASS_FORWARD (`
 `AssemblyNode )`

ARCS_CLASS_FORWARD() [4/130]

`arcs::aubo_scope::ARCS_CLASS_FORWARD (`
 `AssignmentNode )`

ARCS_CLASS_FORWARD() [5/130]

`arcs::aubo_scope::ARCS_CLASS_FORWARD (`
 `AuboCapInfo )`

ARCS__CLASS__FORWARD() [6/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    BaseFeature )
```

ARCS__CLASS__FORWARD() [7/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    BoolRegister )
```

ARCS__CLASS__FORWARD() [8/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    BreakNode )
```

ARCS__CLASS__FORWARD() [9/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    CallNode )
```

ARCS__CLASS__FORWARD() [10/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    CaseNode )
```

ARCS__CLASS__FORWARD() [11/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    CircleMoveNode )
```

ARCS__CLASS__FORWARD() [12/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    CommentNode )
```

ARCS__CLASS__FORWARD() [13/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ConveyorTrackingNode )
```

ARCS__CLASS__FORWARD() [14/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    DataModel )
```

ARCS__CLASS__FORWARD() [15/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    DefaultCaseNode )
```

ARCS__CLASS__FORWARD() [16/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    Device )
```

ARCS__CLASS__FORWARD() [17/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    DeviceManager )
```

ARCS_CLASS_FORWARD() [18/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    DeviceRegistrationManager )
```

ARCS_CLASS_FORWARD() [19/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    DigitalIo )
```

ARCS_CLASS_FORWARD() [20/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    DirectionNode )
```

ARCS_CLASS_FORWARD() [21/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    DoubleRegister )
```

ARCS_CLASS_FORWARD() [22/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    DriveScrewNotOKParameters )
```

ARCS_CLASS_FORWARD() [23/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    DriveScrewOKParameters )
```

ARCS_CLASS_FORWARD() [24/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ElseIfNode )
```

ARCS_CLASS_FORWARD() [25/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ElseNode )
```

ARCS_CLASS_FORWARD() [26/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    EtherNetIP )
```

ARCS_CLASS_FORWARD() [27/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    Expression )
```

ARCS_CLASS_FORWARD() [28/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    Feature )
```

ARCS_CLASS_FORWARD() [29/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    FeatureContributionModel )
```

ARCS__CLASS__FORWARD() [30/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    FeatureLine )
```

ARCS__CLASS__FORWARD() [31/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    FeatureModel )
```

ARCS__CLASS__FORWARD() [32/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    FeaturePlane )
```

ARCS__CLASS__FORWARD() [33/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    FeaturePoint )
```

ARCS__CLASS__FORWARD() [34/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    FeatureTool )
```

ARCS__CLASS__FORWARD() [35/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    FeatureVariable )
```

ARCS__CLASS__FORWARD() [36/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    FeedScrewParameters )
```

ARCS__CLASS__FORWARD() [37/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    FolderNode )
```

ARCS__CLASS__FORWARD() [38/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ForceNode )
```

ARCS__CLASS__FORWARD() [39/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ForceOptionNode )
```

ARCS__CLASS__FORWARD() [40/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    Function )
```

ARCS__CLASS__FORWARD() [41/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    FunctionModel )
```

ARCS_CLASS_FORWARD() [42/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    GlobalVariable )
```

ARCS_CLASS_FORWARD() [43/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    GripActionParameters )
```

ARCS_CLASS_FORWARD() [44/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    GripForceCapability )
```

ARCS_CLASS_FORWARD() [45/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    GripperApiProvider )
```

ARCS_CLASS_FORWARD() [46/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    GripperConfiguration )
```

ARCS_CLASS_FORWARD() [47/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    GripperContribution )
```

ARCS_CLASS_FORWARD() [48/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    GripperDevice )
```

ARCS_CLASS_FORWARD() [49/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    GripperNode )
```

ARCS_CLASS_FORWARD() [50/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    GripVacuumCapability )
```

ARCS_CLASS_FORWARD() [51/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    HaltNode )
```

ARCS_CLASS_FORWARD() [52/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    IfNode )
```

ARCS_CLASS_FORWARD() [53/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    InstallationApi )
```

ARCS__CLASS__FORWARD() [54/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    InstallationApiProvider )
```

ARCS__CLASS__FORWARD() [55/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    InstallationContributionConfiguration )
```

ARCS__CLASS__FORWARD() [56/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    InstallationCreationContext )
```

ARCS__CLASS__FORWARD() [57/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    InstallationNodeContribution )
```

ARCS__CLASS__FORWARD() [58/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    InstallationNodeService )
```

ARCS__CLASS__FORWARD() [59/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    InstallationNodeView )
```

ARCS__CLASS__FORWARD() [60/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    IntegerRegister )
```

ARCS__CLASS__FORWARD() [61/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    Io )
```

ARCS__CLASS__FORWARD() [62/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    IoModel )
```

ARCS__CLASS__FORWARD() [63/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    KeyboardManager )
```

ARCS__CLASS__FORWARD() [64/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    LoopNode )
```

ARCS__CLASS__FORWARD() [65/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ModbusIo )
```

ARCS_CLASS_FORWARD() [66/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    MoveNode )
```

ARCS_CLASS_FORWARD() [67/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    OfflineTrackNode )
```

ARCS_CLASS_FORWARD() [68/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    PalletNode )
```

ARCS_CLASS_FORWARD() [69/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    Payload )
```

ARCS_CLASS_FORWARD() [70/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    PayloadContributionModel )
```

ARCS_CLASS_FORWARD() [71/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    PayloadModel )
```

ARCS_CLASS_FORWARD() [72/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    PersistedVariable )
```

ARCS_CLASS_FORWARD() [73/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    PopupManager )
```

ARCS_CLASS_FORWARD() [74/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    PopupNode )
```

ARCS_CLASS_FORWARD() [75/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    PrepareToStartScrewdriverParameters )
```

ARCS_CLASS_FORWARD() [76/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    Profinet )
```

ARCS_CLASS_FORWARD() [77/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramApi )
```

ARCS__CLASS__FORWARD() [78/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramApiProvider )
```

ARCS__CLASS__FORWARD() [79/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramContributionConfiguration )
```

ARCS__CLASS__FORWARD() [80/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramCreationContext )
```

ARCS__CLASS__FORWARD() [81/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramHomeNode )
```

ARCS__CLASS__FORWARD() [82/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramModel )
```

ARCS__CLASS__FORWARD() [83/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramNode )
```

ARCS__CLASS__FORWARD() [84/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramNodeContribution )
```

ARCS__CLASS__FORWARD() [85/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramNodeFactory )
```

ARCS__CLASS__FORWARD() [86/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramNodeService )
```

ARCS__CLASS__FORWARD() [87/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramNodeView )
```

ARCS__CLASS__FORWARD() [88/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ProgramSelectionParameters )
```

ARCS__CLASS__FORWARD() [89/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    RecordTrackNode )
```


ARCS_CLASS_FORWARD() [90/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    Register )
```

ARCS_CLASS_FORWARD() [91/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    RegisteredCapabilities )
```

ARCS_CLASS_FORWARD() [92/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ReleaseActionParameters )
```

ARCS_CLASS_FORWARD() [93/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    RenderInterface )
```

ARCS_CLASS_FORWARD() [94/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ScrewdriverApiProvider )
```

ARCS_CLASS_FORWARD() [95/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ScrewdriverConfiguration )
```

ARCS_CLASS_FORWARD() [96/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ScrewdriverContribution )
```

ARCS_CLASS_FORWARD() [97/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ScrewdriverParameters )
```

ARCS_CLASS_FORWARD() [98/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ScrewdriverProgram )
```

ARCS_CLASS_FORWARD() [99/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ScrewdriverReadyParameters )
```

ARCS_CLASS_FORWARD() [100/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ScrewdrivingNode )
```

ARCS_CLASS_FORWARD() [101/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ScriptNode )
```

ARCS__CLASS__FORWARD() [102/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ScriptWriter )
```

ARCS__CLASS__FORWARD() [103/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    SeekNode )
```

ARCS__CLASS__FORWARD() [104/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    SetNode )
```

ARCS__CLASS__FORWARD() [105/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    SetPayloadNode )
```

ARCS__CLASS__FORWARD() [106/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    SpeedCapability )
```

ARCS__CLASS__FORWARD() [107/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    SubProgNode )
```

ARCS__CLASS__FORWARD() [108/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    SwitchNode )
```

ARCS__CLASS__FORWARD() [109/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    SystemApi )
```

ARCS__CLASS__FORWARD() [110/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    TCP )
```

ARCS__CLASS__FORWARD() [111/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    TcpContributionModel )
```

ARCS__CLASS__FORWARD() [112/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    TcpModel )
```

ARCS__CLASS__FORWARD() [113/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ThreadNode )
```

ARCS_CLASS_FORWARD() [114/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    TimerNode )
```

ARCS_CLASS_FORWARD() [115/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    TreeNode )
```

ARCS_CLASS_FORWARD() [116/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    UndoRedoManager )
```

ARCS_CLASS_FORWARD() [117/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    UntilNode )
```

ARCS_CLASS_FORWARD() [118/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    UserInteraction )
```

ARCS_CLASS_FORWARD() [119/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    UserInterfaceApi )
```

ARCS_CLASS_FORWARD() [120/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ValueFactory )
```

ARCS_CLASS_FORWARD() [121/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    Variable )
```

ARCS_CLASS_FORWARD() [122/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    VariableFactory )
```

ARCS_CLASS_FORWARD() [123/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    VariableModel )
```

ARCS_CLASS_FORWARD() [124/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ViewApiProvider )
```

ARCS_CLASS_FORWARD() [125/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    VisualLocalizationProgramNode )
```

ARCS_CLASS_FORWARD() [126/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    WaitNode )
```

ARCS_CLASS_FORWARD() [127/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    Waypoint )
```

ARCS_CLASS_FORWARD() [128/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    WaypointModel )
```

ARCS_CLASS_FORWARD() [129/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    WaypointNode )
```

ARCS_CLASS_FORWARD() [130/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    WidthCapability )
```

10.3 arcs::aubo_sdk 命名空间参考

函数

- [ARCS_CLASS_FORWARD](#) (RobotProxy)

10.3.1 函数说明

ARCS_CLASS_FORWARD()

```
arcs::aubo_sdk::ARCS_CLASS_FORWARD (
    RobotProxy )
```

Chapter 11

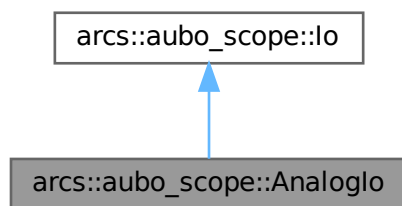
类说明

11.1 arcs::aubo_scope::AnalogIo 类参考

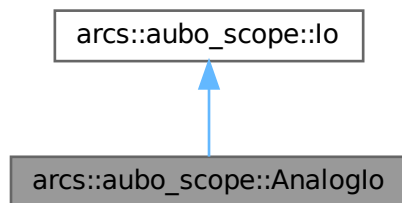
模拟 IO 接口此接口提供对模拟 I/O 的支持。

```
#include <analog_io.h>
```

类 arcs::aubo_scope::AnalogIo 继承关系图:



arcs::aubo_scope::AnalogIo 的协作图:



Public 成员函数

- [AnalogIo](#) (AnalogIo &f)
- [AnalogIo](#) (AnalogIo &&f)
- [~AnalogIo](#) ()
- [double getMinRangeValue](#) ()

- double `getMaxRangeValue` ()
- bool `isCurrent` ()
- bool `isVoltage` ()
- bool `setValue` (double value)
设置模拟I/O 输出信号。注意该I/O 必须是输出类型。
- double `getValue` ()
获取I/O 的最后一次读数

Public 成员函数继承自 `arcs::aubo_scope::Io`

- `Io` (Io &f)
- `Io` (Io &&f)
- virtual `~Io` ()
- std::string `getDisplayName` () const
获取当前安装中I/O 的用户自定义名称
- std::string `getDefaultName` () const
获取I/O 的机器人默认名称
- std::string `getValueStr` () const
获取I/O 的当前读数值 (字符串形式)
- `IoType` `getType` () const
获取I/O 类型
- `InterfaceType` `getInterfaceType` () const
获取此I/O 所属的接口类型
- int `getAction` () const
获取I/O 的动作编号
- bool `isInput` () const
判断I/O 是否支持读取值
- bool `isResolvable` () const
某些I/O (如MODBUS I/O 和模拟工具输入) 在某些情况下可能不可用。此方法用于判断 I/O 是否可用。
- size_t `hashCode` ()

Private 成员函数

- `AnalogIo` ()

Private 属性

- void * `d_` { nullptr }

友元

- class `DataSwitch`

额外继承的成员函数

Public 类型继承自 `arcs::aubo_scope::Io`

- enum `IoType` : int {
DIGITAL , ANALOG , INTEGER , BOOLEAN ,
DOUBLE }
- enum `InterfaceType` : int {
STANDARD , TOOL , CONFIGURABLE , LINKIO ,
MODBUS , GENERAL_PURPOSE , PROFINET , ETHERNET_IP }

Protected 成员函数继承自 `arcs::aubo_scope::Io`

- `Io` ()

11.1.1 详细描述

模拟 IO 接口此接口提供对模拟 I/O 的支持。
在文件 [analog_io.h](#) 第 20 行定义。

11.1.2 构造及析构函数说明

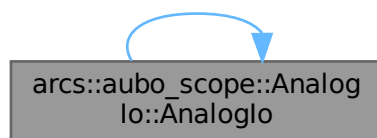
AnalogIo() [1/3]

```
arcs::aubo_scope::AnalogIo::AnalogIo (  
    AnalogIo & f)
```

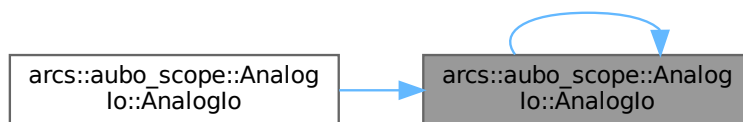
引用了 [AnalogIo\(\)](#)。

被这些函数引用 [AnalogIo\(\)](#) , 以及 [AnalogIo\(\)](#)。

函数调用图:



这是这个函数的调用关系图:

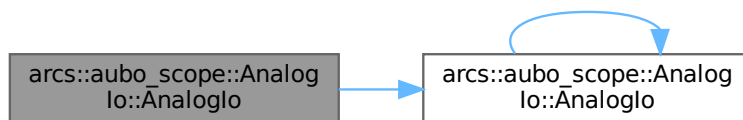


AnalogIo() [2/3]

```
arcs::aubo_scope::AnalogIo::AnalogIo (  
    AnalogIo && f)
```

引用了 [AnalogIo\(\)](#)。

函数调用图:



~AnalogIo()

```
arcs::aubo_scope::AnalogIo::~~AnalogIo ()
```

AnalogIo() [3/3]

```
arcs::aubo_scope::AnalogIo::AnalogIo () [private]
```

11.1.3 成员函数说明**getMaxRangeValue()**

```
double arcs::aubo_scope::AnalogIo::getMaxRangeValue ()
```

getMinRangeValue()

```
double arcs::aubo_scope::AnalogIo::getMinRangeValue ()
```

getValue()

```
double arcs::aubo_scope::AnalogIo::getValue ()
```

获取I/O 的最后一次读数

返回

I/O 的最后一次读数。

isCurrent()

```
bool arcs::aubo_scope::AnalogIo::isCurrent ()
```

isVoltage()

```
bool arcs::aubo_scope::AnalogIo::isVoltage ()
```

setValue()

```
bool arcs::aubo_scope::AnalogIo::setValue (
    double value)
```

设置模拟I/O 输出信号。注意该I/O 必须是输出类型。

参数

<i>value</i>	要设置的模拟值。

返回

如果模拟信号设置成功返回 `true`, 否则返回 `false`。

11.1.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [analog_io.h](#) 第 63 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.1.5 类成员变量说明

d_

```
void* arcs::aubo_scope::AnalogIo::d_ { nullptr } [private]
```

在文件 [analog_io.h](#) 第 65 行定义.

该类的文档由以下文件生成:

- include/aubo_caps/domain/io/[analog_io.h](#)

11.2 arcs::aubo_scope::AnalogIoFilter 结构体参考

```
#include <analog_io.h>
```

Public 成员函数

- [AnalogIoFilter](#) ([Io::InterfaceType](#) interface, bool is_input)
- bool [operator\(\)](#) (IoPtr io)

Private 属性

- [Io::InterfaceType](#) interface_
- bool is_input_

11.2.1 详细描述

在文件 [analog_io.h](#) 第 68 行定义.

11.2.2 构造及析构函数说明

AnalogIoFilter()

```
arcs::aubo_scope::AnalogIoFilter::AnalogIoFilter (
    Io::InterfaceType interface,
    bool is_input) [inline]
```

在文件 [analog_io.h](#) 第 70 行定义.

引用了 [interface_](#), 以及 [is_input_](#).

11.2.3 成员函数说明

operator()()

```
bool arcs::aubo_scope::AnalogIoFilter::operator() (
    IoPtr io) [inline]
```

在文件 [analog_io.h](#) 第 74 行定义.

引用了 [interface_](#), 以及 [is_input_](#).

11.2.4 类成员变量说明

interface_

```
Io::InterfaceType arcs::aubo_scope::AnalogIoFilter::interface_ [private]
```

在文件 [analog_io.h](#) 第 82 行定义.

被这些函数引用 [AnalogIoFilter\(\)](#), 以及 [operator\(\)\(\)](#).

is_input_

```
bool arcs::aubo_scope::AnalogIoFilter::is_input_ [private]
```

在文件 [analog_io.h](#) 第 83 行定义.

被这些函数引用 [AnalogIoFilter\(\)](#), 以及 [operator\(\)\(\)](#).

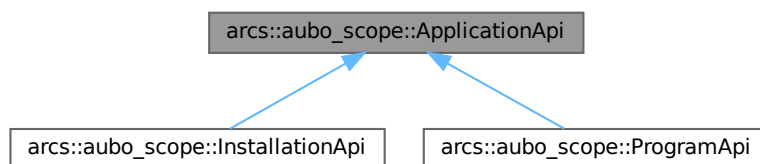
该结构体的文档由以下文件生成:

- include/aubo_caps/domain/io/[analog_io.h](#)

11.3 arcs::aubo_scope::ApplicationApi 类参考

#include <application_api.h>

类 arcs::aubo_scope::ApplicationApi 继承关系图:



Public 成员函数

- [ApplicationApi](#) (ApplicationApi &f)
- [ApplicationApi](#) (ApplicationApi &&f)
- virtual [~ApplicationApi](#) ()
- IoModelPtr [getIoModel](#) ()
IoModel 获取IO 模型接口，用于操作机器人的输入和输出。
- FeatureModelPtr [getFeatureModel](#) ()
FeatureModel 获取特征模型接口，用于操作当前安装中的特征。
- TcpModelPtr [getTcpModel](#) ()
TcpModel 获取TCP 模型接口，用于访问当前安装中的TCP。
- PayloadModelPtr [getPayloadModel](#) ()
PayloadModel 获取负载模型接口，用于访问当前AuboScope 安装中的负载。
- VariableModelPtr [getVariableModel](#) ()
VariableModel 获取变量模型接口，用于操作变量。
- ValueFactoryPtr [getValueFactory](#) ()
ValueFactory 获取值工厂接口，用于访问能够创建值对象的各种工厂。
- template<typename T>
std::shared_ptr< T > [getInstallationNode](#) ()
此方法用于获取指定的*InstallationNodeContribution*实例。
- InstallationNodeContributionPtr [getInstallationNode](#) (const char *typeidOfClass)
- DeviceManagerPtr [getDeviceManager](#) ()
DeviceManager 此方法用于获取指定的*DeviceManager*实例。
- aubo_sdk::RobotProxyPtr [getRobotProxy](#) ()
获取机器人代理，提供与机器人底层SDK 的交互接口。

Protected 成员函数

- [ApplicationApi](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.3.1 详细描述

在文件 [application_api.h](#) 第 41 行定义.

11.3.2 构造及析构函数说明

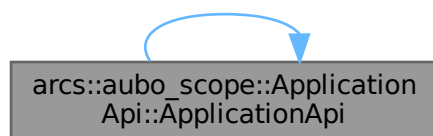
ApplicationApi() [1/3]

```
arcs::aubo_scope::ApplicationApi::ApplicationApi (  
    ApplicationApi & f)
```

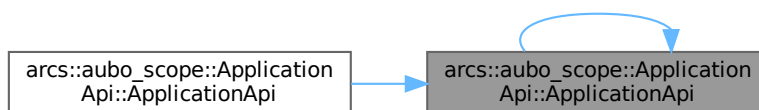
引用了 [ApplicationApi\(\)](#).

被这些函数引用 [ApplicationApi\(\)](#) , 以及 [ApplicationApi\(\)](#).

函数调用图:



这是这个函数的调用关系图:

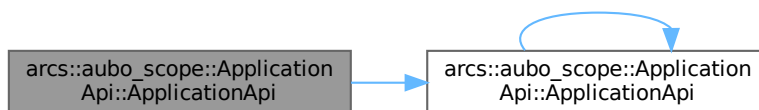


ApplicationApi() [2/3]

```
arcs::aubo_scope::ApplicationApi::ApplicationApi (  
    ApplicationApi && f)
```

引用了 [ApplicationApi\(\)](#).

函数调用图:



~ApplicationApi()

```
virtual arcs::aubo_scope::ApplicationApi::~~ApplicationApi () [virtual]
```

ApplicationApi() [3/3]

```
arcs::aubo_scope::ApplicationApi::ApplicationApi () [protected]
```

11.3.3 成员函数说明

getInstallationNode() [1/2]

```
template<typename T>
std::shared_ptr< T > arcs::aubo_scope::ApplicationApi::getInstallationNode () [inline]
```

此方法用于获取指定的[InstallationNodeContribution](#)实例。

@return An interface with access to various queryable-only system

resources

参数

<i>installationType</i>	要返回的安装节点贡献的类。
<i>< T ></i>	用于指定 InstallationNodeContribution 的泛型。

返回

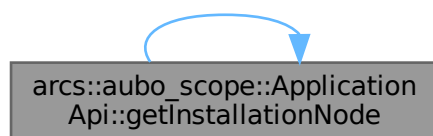
安装节点实例。

在文件 [application_api.h](#) 第 166 行定义.

引用了 [getInstallationNode\(\)](#).

被这些函数引用 [getInstallationNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:



getInstallationNode() [2/2]

```
InstallationNodeContributionPtr arcs::aubo_scope::ApplicationApi::getInstallationNode (
    const char * typeidOfClass)
```

getRobotProxy()

```
aubo_sdk::RobotProxyPtr arcs::aubo_scope::ApplicationApi::getRobotProxy ()
```

获取机器人代理, 提供与机器人底层SDK 的交互接口。

返回

机器人代理指针。

11.3.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [application_api.h](#) 第 213 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.3.5 类成员变量说明

d_

```
void* arcs::aubo_scope::ApplicationApi::d_ { nullptr } [private]
```

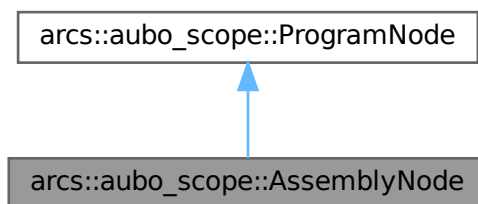
在文件 [application_api.h](#) 第 214 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/application_api.h](#)

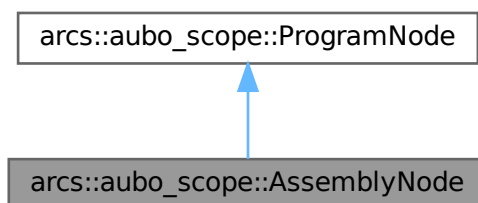
11.4 arcs::aubo_scope::AssemblyNode 类参考

```
#include <assembly_node.h>
```

类 `arcs::aubo_scope::AssemblyNode` 继承关系图:



`arcs::aubo_scope::AssemblyNode` 的协作图:



Public 类型

- enum `ConfigType` { `NONE` , `HOLE_SEARCH` , `COMPLIANT_INSERTION` }

Public 成员函数

- `AssemblyNode` (`AssemblyNode &f`)
- `AssemblyNode` (`AssemblyNode &&f`)
- virtual `~AssemblyNode` ()
- void `addAssemblyUntilNode` ()

Public 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` (`ProgramNode &f`)
- `ProgramNode` (`ProgramNode &&f`)
- virtual `~ProgramNode` ()
- `ProgramNodeContributionPtr` `getProgramNodeContribution` ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `ScriptWriter::writeChildren`。

Private 成员函数

- `AssemblyNode` ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.4.1 详细描述

在文件 [assembly_node.h](#) 第 10 行定义.

11.4.2 成员枚举类型说明

ConfigType

```
enum arcs::aubo\_scope::AssemblyNode::ConfigType
```

枚举值

NONE	
HOLE_SEARCH	
COMPLIANT_INSERTION	

在文件 [assembly_node.h](#) 第 13 行定义.

11.4.3 构造及析构造函数说明

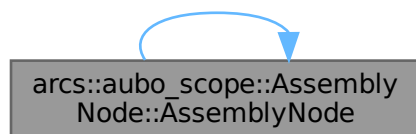
AssemblyNode() [1/3]

```
arcs::aubo\_scope::AssemblyNode::AssemblyNode (  
    AssemblyNode & f)
```

引用了 [AssemblyNode\(\)](#).

被这些函数引用 [AssemblyNode\(\)](#) , 以及 [AssemblyNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:



AssemblyNode() [2/3]

```
arcs::aubo_scope::AssemblyNode::AssemblyNode (
    AssemblyNode && f)
```

引用了 [AssemblyNode\(\)](#).

函数调用图:



~AssemblyNode()

```
virtual arcs::aubo_scope::AssemblyNode::~~AssemblyNode () [virtual]
```

AssemblyNode() [3/3]

```
arcs::aubo_scope::AssemblyNode::AssemblyNode () [private]
```

11.4.4 成员函数说明

addAssemblyUntilNode()

```
void arcs::aubo_scope::AssemblyNode::addAssemblyUntilNode ()
```

11.4.5 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 `assembly_node.h` 第 27 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.4.6 类成员变量说明

d_

```
void* arcs::aubo_scope::AssemblyNode::d_ { nullptr } [private]
```

在文件 [assembly_node.h](#) 第 29 行定义.

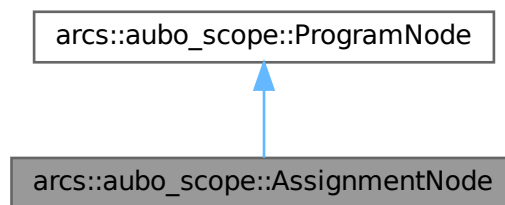
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/assembly_node.h](#)

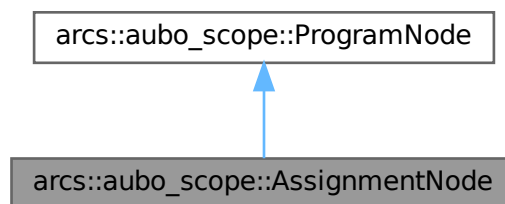
11.5 arcs::aubo_scope::AssignmentNode 类参考

```
#include <assignment_node.h>
```

类 arcs::aubo_scope::AssignmentNode 继承关系图:



arcs::aubo_scope::AssignmentNode 的协作图:



Public 类型

- enum [ConfigType](#) { [EXPRESSION](#) , [OPERATOR](#) }
The configuration type used to determine which type of configuration this instance is.
- enum [InputMode](#) { [INPUT_BOOL](#) , [INPUT_INTEGER](#) , [INPUT_FLOAT](#) , [INPUT_STRING](#) }

Public 成员函数

- [AssignmentNode](#) ([AssignmentNode](#) &f)
- [AssignmentNode](#) ([AssignmentNode](#) &&f)
- virtual [~AssignmentNode](#) ()
- [ConfigType](#) [getConfigType](#) ()

- void `setConfigType` (`ConfigType` type)
- `VariablePtr` `getVariable` ()
- void `setVariable` (`VariablePtr` var)
- `ExpressionPtr` `getExpression` ()
- void `setExpression` (`ExpressionPtr` exp)

Public 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` (`ProgramNode` &f)
- `ProgramNode` (`ProgramNode` &&f)
- virtual `~ProgramNode` ()
- `ProgramNodeContributionPtr` `getProgramNodeContribution` ()
获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `ScriptWriter::writeChildren`。

Private 成员函数

- `AssignmentNode` ()

Private 属性

- void * `d_` { nullptr }

友元

- class `DataSwitch`

额外继承的成员函数

Protected 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` ()

11.5.1 详细描述

在文件 `assignment_node.h` 第 12 行定义.

11.5.2 成员枚举类型说明

ConfigType

enum `arcs::aubo_scope::AssignmentNode::ConfigType`

The configuration type used to determine which type of configuration this instance is.

枚举值

EXPRESSION	<code>Expression</code> assignment has been selected. The config instance can be cast to <code>ExpressionAssignmentNodeConfig</code> .
OPERATOR	Operator assignment has been selected. This type has no further information. Setting this type of config will be ignored and will have no effect on the node.

在文件 `assignment_node.h` 第 19 行定义.

InputMode

enum `arcs::aubo_scope::AssignmentNode::InputMode`

枚举值

INPUT_BOOL	
INPUT_INTEGER	
INPUT_FLOAT	
INPUT_STRING	

在文件 [assignment_node.h](#) 第 39 行定义.

11.5.3 构造及析构造函数说明

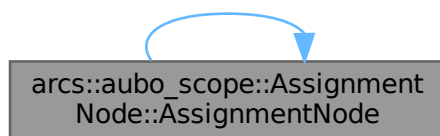
AssignmentNode() [1/3]

```
arcs::aubo_scope::AssignmentNode::AssignmentNode (  
    AssignmentNode & f)
```

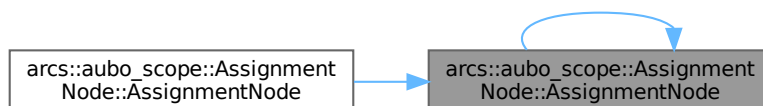
引用了 [AssignmentNode\(\)](#).

被这些函数引用 [AssignmentNode\(\)](#) , 以及 [AssignmentNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

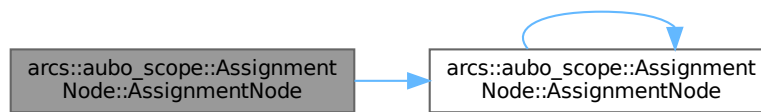


AssignmentNode() [2/3]

```
arcs::aubo_scope::AssignmentNode::AssignmentNode (  
    AssignmentNode && f)
```

引用了 [AssignmentNode\(\)](#).

函数调用图:



~AssignmentNode()

```
virtual arcs::aubo_scope::AssignmentNode::~~AssignmentNode () [virtual]
```

AssignmentNode() [3/3]

```
arcs::aubo_scope::AssignmentNode::AssignmentNode () [private]
```

11.5.4 成员函数说明

getConfigType()

```
ConfigType arcs::aubo_scope::AssignmentNode::getConfigType ()
```

返回

getExpression()

```
ExpressionPtr arcs::aubo_scope::AssignmentNode::getExpression ()
```

返回

the expression whose evaluation is assigned to the variable.

getVariable()

```
VariablePtr arcs::aubo_scope::AssignmentNode::getVariable ()
```

返回

the variable assigned to the expression evaluation.

setConfigType()

```
void arcs::aubo_scope::AssignmentNode::setConfigType (
    ConfigType type)
```

setExpression()

```
void arcs::aubo_scope::AssignmentNode::setExpression (
    ExpressionPtr exp)
```

setVariable()

```
void arcs::aubo_scope::AssignmentNode::setVariable (
    VariablePtr var)
```

11.5.5 友元及相关符号说明

DataSwitch

`friend class DataSwitch [friend]`

在文件 [assignment_node.h](#) 第 75 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.5.6 类成员变量说明

d_

`void* arcs::aubo_scope::AssignmentNode::d_ { nullptr } [private]`

在文件 [assignment_node.h](#) 第 77 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/assignment_node.h](#)

11.6 arcs::aubo_scope::AuboCapInfo 类参考

This interface provides various metadata information about an installed AuboCap, such as the name of the AuboCap.

```
#include <extension_info.h>
```

Public 成员函数

- `virtual ~AuboCapInfo ()=default`
- `virtual std::string getSymbolicName ()=0`
@retAubon The symbolic name (id) of the AuboCap.
- `virtual std::string getName ()=0`
@retAubon The name of the AuboCap.
- `virtual std::string getVendor ()=0`
@retAubon The vendor of the AuboCap.

11.6.1 详细描述

This interface provides various metadata information about an installed AuboCap, such as the name of the AuboCap.

自从

1.1.0

在文件 [extension_info.h](#) 第 17 行定义.

11.6.2 构造及析构函数说明

~AuboCapInfo()

```
virtual arcs::aubo_scope::AuboCapInfo::~~AuboCapInfo () [virtual], [default]
```

11.6.3 成员函数说明

getName()

```
virtual std::string arcs::aubo_scope::AuboCapInfo::getName () [pure virtual]
```

@retAubon The name of the AuboCap.

getSymbolicName()

```
virtual std::string arcs::aubo_scope::AuboCapInfo::getSymbolicName () [pure virtual]
@retAubon The symbolic name (id) of the AuboCap.
```

getVendor()

```
virtual std::string arcs::aubo_scope::AuboCapInfo::getVendor () [pure virtual]
@retAubon The vendor of the AuboCap.
该类的文档由以下文件生成:
```

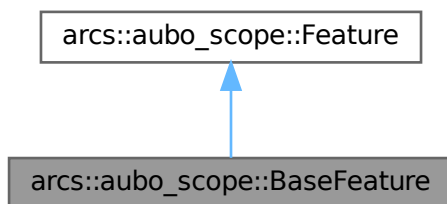
- include/aubo_caps/domain/[extension_info.h](#)

11.7 arcs::aubo_scope::BaseFeature 类参考

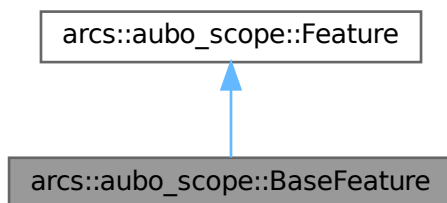
This interface represents the Base feature in AuboScope.

```
#include <base_feature.h>
```

类 arcs::aubo_scope::BaseFeature 继承关系图:



arcs::aubo_scope::BaseFeature 的协作图:



Public 成员函数

- [BaseFeature](#) (BaseFeature &f)
- [BaseFeature](#) (BaseFeature &&f)
- virtual [~BaseFeature](#) ()

Public 成员函数继承自 [arcs::aubo_scope::Feature](#)

- [Feature](#) (Feature &f)
- [Feature](#) (Feature &&f)
- virtual [~Feature](#) ()
- std::string [getDisplay_name](#) ()
获取特征的显示名称。注意：特征可随时被重命名。
- bool [isDefined](#) ()
返回特征的位置是否已完全定义/指定。
- bool [isVariable](#) ()
返回特征是否可作为变量在程序中使用。
- bool [isJoggable](#) ()
- bool [isResolvable](#) ()
特征无法保证始终存在于 *AuboScope* 中。终端用户可以加载不包含该特征的不同安装，或者终端用户或添加该特征的 *AuboCap* 可以将其移除。此方法用于判断特征是否存在。
- std::vector< double > [getPose](#) ()
返回特征的笛卡尔位置。
- size_t [hashCode](#) ()

Private 成员函数

- [BaseFeature](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::Feature](#)

- [Feature](#) ()

11.7.1 详细描述

This interface represents the Base feature in AuboScope.
在文件 [base_feature.h](#) 第 13 行定义。

11.7.2 构造及析构造函数说明

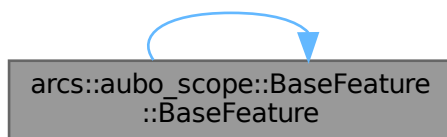
[BaseFeature\(\)](#) [1/3]

```
arcs::aubo_scope::BaseFeature::BaseFeature (
    BaseFeature & f)
```

引用了 [BaseFeature\(\)](#).

被这些函数引用 [BaseFeature\(\)](#) , 以及 [BaseFeature\(\)](#).

函数调用图:



这是这个函数的调用关系图:



BaseFeature() [2/3]

```
arcs::aubo_scope::BaseFeature::BaseFeature (
    BaseFeature && f)
```

引用了 [BaseFeature\(\)](#).

函数调用图:



~BaseFeature()

```
virtual arcs::aubo_scope::BaseFeature::~~BaseFeature () [virtual]
```

BaseFeature() [3/3]

```
arcs::aubo_scope::BaseFeature::BaseFeature () [private]
```

11.7.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [base_feature.h](#) 第 21 行定义.

引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.7.4 类成员变量说明

d_

```
void* arcs::aubo_scope::BaseFeature::d_ { nullptr } [private]
```

在文件 [base_feature.h](#) 第 23 行定义.

该类的文档由以下文件生成:

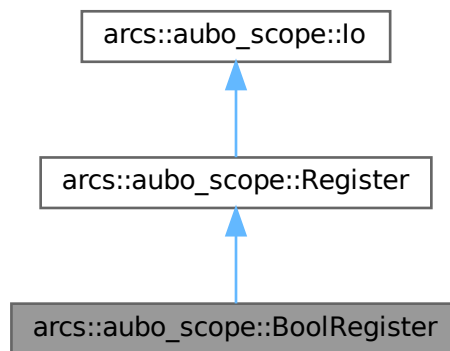
- [include/aubo_caps/domain/feature/base_feature.h](#)

11.8 arcs::aubo_scope::BoolRegister 类参考

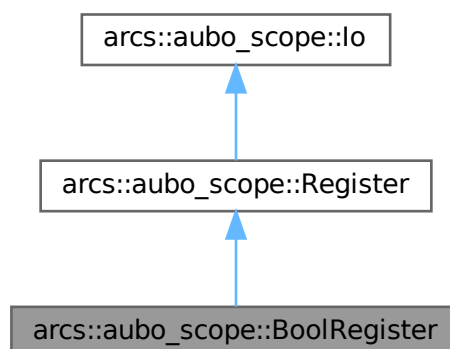
This interface provides support for bool registers.

```
#include <boolean_register.h>
```

类 arcs::aubo_scope::BoolRegister 继承关系图:



arcs::aubo_scope::BoolRegister 的协作图:



Public 成员函数

- virtual `~BoolRegister ()`=default

Public 成员函数继承自 `arcs::aubo_scope::Register`

- virtual `~Register ()`=default

Public 成员函数继承自 `arcs::aubo_scope::Io`

- `Io (Io &f)`
- `Io (Io &&f)`
- virtual `~Io ()`
- `std::string getDisplayName () const`
获取当前安装中 *I/O* 的用户自定义名称
- `std::string getDefaultName () const`
获取 *I/O* 的机器人默认名称
- `std::string getValueStr () const`
获取 *I/O* 的当前读数值 (字符串形式)
- `IoType getType () const`
获取 *I/O* 类型
- `InterfaceType getInterfaceType () const`
获取此 *I/O* 所属的接口类型
- `int getAction () const`
获取 *I/O* 的动作编号
- `bool isInput () const`
判断 *I/O* 是否支持读取值
- `bool isResolvable () const`
某些 *I/O* (如 *MODBUS I/O* 和模拟工具输入) 在某些情况下可能不可用。此方法用于判断 *I/O* 是否可用。
- `size_t hashCode ()`

额外继承的成员函数**Public 类型继承自 `arcs::aubo_scope::Io`**

- `enum IoType : int {
DIGITAL , ANALOG , INTEGER , BOOLEAN ,
DOUBLE }`
- `enum InterfaceType : int {
STANDARD , TOOL , CONFIGURABLE , LINKIO ,
MODBUS , GENERAL_PURPOSE , PROFINET , ETHERNET_IP }`

Protected 成员函数继承自 `arcs::aubo_scope::Io`

- `Io ()`

11.8.1 详细描述

This interface provides support for bool registers.
在文件 `boolean_register.h` 第 14 行定义.

11.8.2 构造及析构造函数说明**`~BoolRegister()`**

```
virtual arcs::aubo_scope::BoolRegister::~~BoolRegister () [virtual], [default]
```

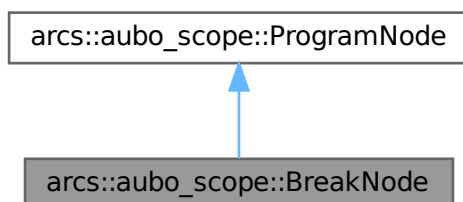
该类的文档由以下文件生成:

- `include/aubo_caps/domain/io/boolean_register.h`

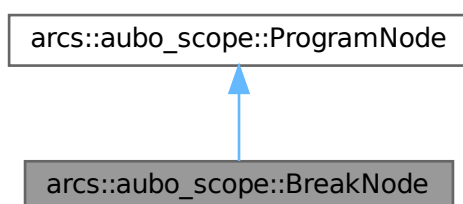
11.9 arcs::aubo_scope::BreakNode 类参考

```
#include <break_node.h>
```

类 arcs::aubo_scope::BreakNode 继承关系图:



arcs::aubo_scope::BreakNode 的协作图:



Public 成员函数

- [BreakNode](#) (BreakNode &f)
- [BreakNode](#) (BreakNode &&f)
- virtual [~BreakNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `ScriptWriter::writeChildren`。

Private 成员函数

- [BreakNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.9.1 详细描述

在文件 [break_node.h](#) 第 10 行定义.

11.9.2 构造及析构函数说明

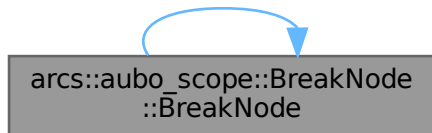
[BreakNode\(\)](#) [1/3]

```
arcs::aubo_scope::BreakNode::BreakNode (  
    BreakNode & f)
```

引用了 [BreakNode\(\)](#).

被这些函数引用 [BreakNode\(\)](#) , 以及 [BreakNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

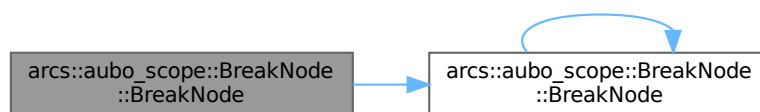


[BreakNode\(\)](#) [2/3]

```
arcs::aubo_scope::BreakNode::BreakNode (  
    BreakNode && f)
```

引用了 [BreakNode\(\)](#).

函数调用图:



~BreakNode()

```
virtual arcs::aubo_scope::BreakNode::~~BreakNode () [virtual]
```

BreakNode() [3/3]

```
arcs::aubo_scope::BreakNode::BreakNode () [private]
```

11.9.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [break_node.h](#) 第 18 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.9.4 类成员变量说明

d_

```
void* arcs::aubo_scope::BreakNode::d_ { nullptr } [private]
```

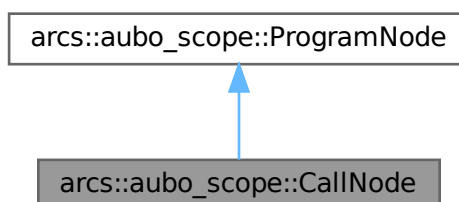
在文件 [break_node.h](#) 第 20 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/break_node.h](#)

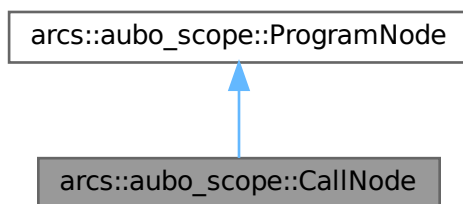
11.10 arcs::aubo_scope::CallNode 类参考

```
#include <call_node.h>
```

类 arcs::aubo_scope::CallNode 继承关系图:



arcs::aubo_scope::CallNode 的协作图:



Public 成员函数

- [CallNode](#) (CallNode &f)
- [CallNode](#) (CallNode &&f)
- virtual [~CallNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [CallNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.10.1 详细描述

在文件 [call_node.h](#) 第 10 行定义.

11.10.2 构造及析构函数说明

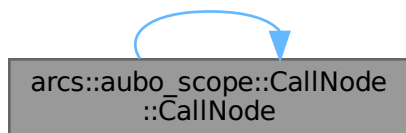
CallNode() [1/3]

```
arcs::aubo_scope::CallNode::CallNode (  
    CallNode & f)
```

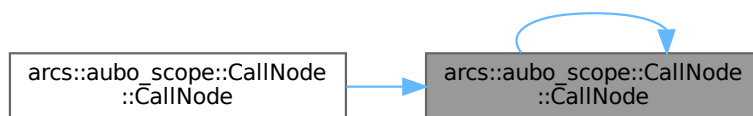
引用了 [CallNode\(\)](#).

被这些函数引用 [CallNode\(\)](#) , 以及 [CallNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

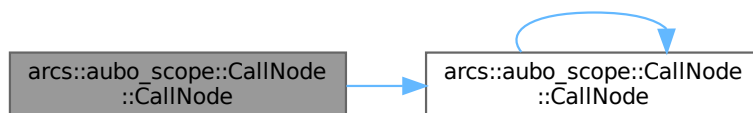


CallNode() [2/3]

```
arcs::aubo_scope::CallNode::CallNode (  
    CallNode && f)
```

引用了 [CallNode\(\)](#).

函数调用图:



~CallNode()

```
virtual arcs::aubo_scope::CallNode::~~CallNode () [virtual]
```

CallNode() [3/3]

```
arcs::aubo_scope::CallNode::CallNode () [private]
```

11.10.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [call_node.h](#) 第 18 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.10.4 类成员变量说明

d_

```
void* arcs::aubo_scope::CallNode::d_ { nullptr } [private]
```

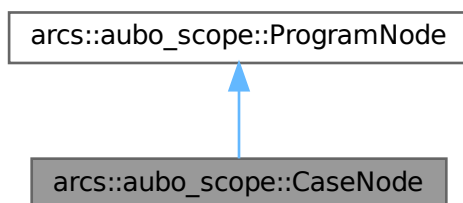
在文件 [call_node.h](#) 第 20 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/call_node.h](#)

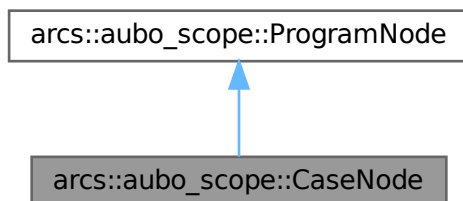
11.11 arcs::aubo_scope::CaseNode 类参考

```
#include <case_node.h>
```

类 arcs::aubo_scope::CaseNode 继承关系图:



arcs::aubo_scope::CaseNode 的协作图:



Public 成员函数

- [CaseNode](#) (CaseNode &f)
- [CaseNode](#) (CaseNode &&f)
- virtual [~CaseNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [CaseNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数**Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)**

- [ProgramNode](#) ()

11.11.1 详细描述

在文件 [case_node.h](#) 第 10 行定义.

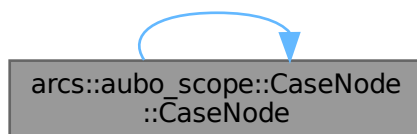
11.11.2 构造及析构函数说明**CaseNode() [1/3]**

```
arcs::aubo_scope::CaseNode::CaseNode (
    CaseNode & f)
```

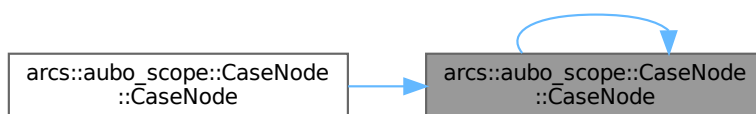
引用了 [CaseNode\(\)](#).

被这些函数引用 [CaseNode\(\)](#), 以及 [CaseNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

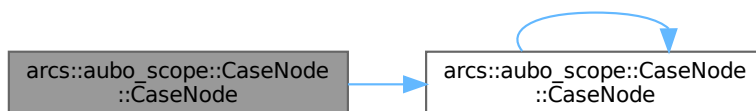


CaseNode() [2/3]

```
arcs::aubo_scope::CaseNode::CaseNode (
    CaseNode && f)
```

引用了 [CaseNode\(\)](#).

函数调用图:



~CaseNode()

```
virtual arcs::aubo_scope::CaseNode::~~CaseNode () [virtual]
```

CaseNode() [3/3]

```
arcs::aubo_scope::CaseNode::CaseNode () [private]
```

11.11.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [case_node.h](#) 第 18 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.11.4 类成员变量说明

d_

```
void* arcs::aubo_scope::CaseNode::d_ { nullptr } [private]
```

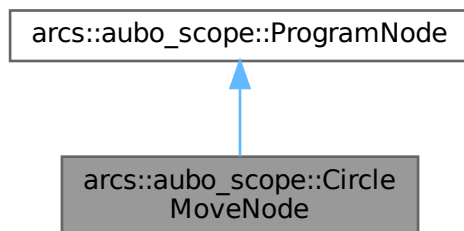
在文件 [case_node.h](#) 第 20 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/case_node.h](#)

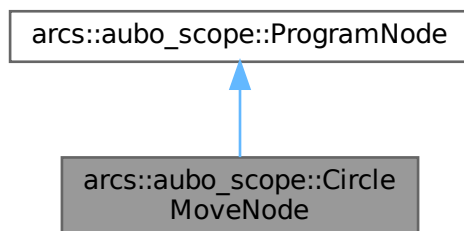
11.12 arcs::aubo_scope::CircleMoveNode 类参考

```
#include <circle_move_node.h>
```

类 arcs::aubo_scope::CircleMoveNode 继承关系图:



arcs::aubo_scope::CircleMoveNode 的协作图:



Public 成员函数

- [CircleMoveNode](#) (CircleMoveNode &f)
- [CircleMoveNode](#) (CircleMoveNode &&f)
- virtual [~CircleMoveNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `ScriptWriter::writeChildren`。

Private 成员函数

- [CircleMoveNode](#) ()

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数**Protected 成员函数继承自 `arcs::aubo_scope::ProgramNode`**

- `ProgramNode ()`

11.12.1 详细描述

在文件 `circle_move_node.h` 第 10 行定义.

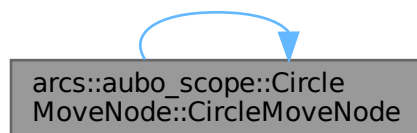
11.12.2 构造及析构函数说明**CircleMoveNode() [1/3]**

```
arcs::aubo_scope::CircleMoveNode::CircleMoveNode (
    CircleMoveNode & f)
```

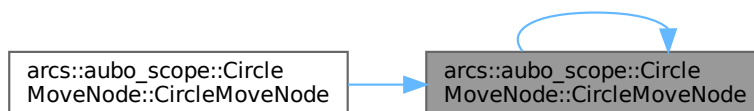
引用了 `CircleMoveNode()`.

被这些函数引用 `CircleMoveNode()` , 以及 `CircleMoveNode()`.

函数调用图:



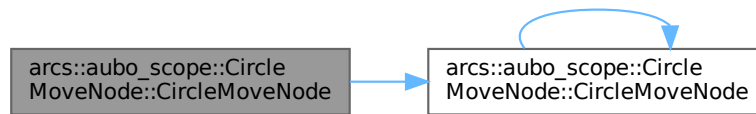
这是这个函数的调用关系图:

**CircleMoveNode() [2/3]**

```
arcs::aubo_scope::CircleMoveNode::CircleMoveNode (
    CircleMoveNode && f)
```

引用了 `CircleMoveNode()`.

函数调用图:



~CircleMoveNode()

```
virtual arcs::aubo_scope::CircleMoveNode::~~CircleMoveNode () [virtual]
```

CircleMoveNode() [3/3]

```
arcs::aubo_scope::CircleMoveNode::CircleMoveNode () [private]
```

11.12.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [circle_move_node.h](#) 第 18 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.12.4 类成员变量说明

d_

```
void* arcs::aubo_scope::CircleMoveNode::d_ { nullptr } [private]
```

在文件 [circle_move_node.h](#) 第 20 行定义.

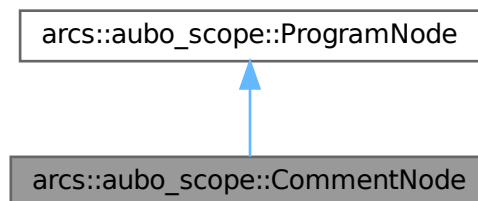
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/circle_move_node.h](#)

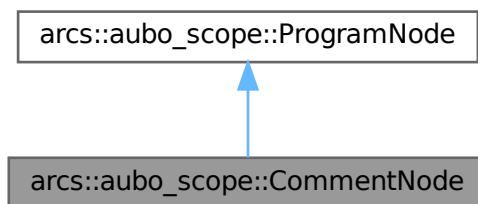
11.13 arcs::aubo_scope::CommentNode 类参考

```
#include <comment_node.h>
```

类 arcs::aubo_scope::CommentNode 继承关系图:



arcs::aubo_scope::CommentNode 的协作图:



Public 成员函数

- `CommentNode` (`CommentNode &f`)
- `CommentNode` (`CommentNode &&f`)
- `virtual ~CommentNode` ()
- `std::string getComment` () const
- `void setComment` (const `std::string &param_string`)

Public 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` (`ProgramNode &f`)
- `ProgramNode` (`ProgramNode &&f`)
- `virtual ~ProgramNode` ()
- `ProgramNodeContributionPtr getProgramNodeContribution` ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `ScriptWriter::writeChildren`。

Private 成员函数

- `CommentNode` ()

Private 属性

- `void * d_` { `nullptr` }

友元

- `class DataSwitch`

额外继承的成员函数

Protected 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` ()

11.13.1 详细描述

在文件 `comment_node.h` 第 11 行定义.

11.13.2 构造及析构函数说明

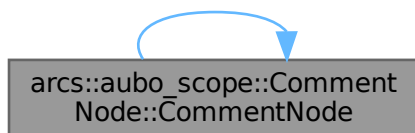
CommentNode() [1/3]

```
arcs::aubo_scope::CommentNode::CommentNode (  
    CommentNode & f)
```

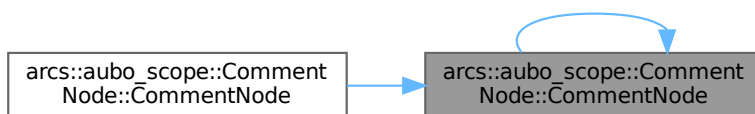
引用了 [CommentNode\(\)](#).

被这些函数引用 [CommentNode\(\)](#) , 以及 [CommentNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

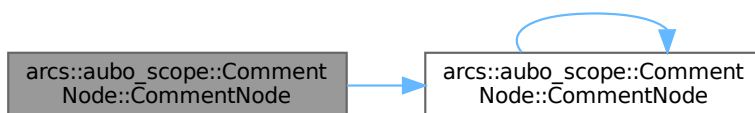


CommentNode() [2/3]

```
arcs::aubo_scope::CommentNode::CommentNode (  
    CommentNode && f)
```

引用了 [CommentNode\(\)](#).

函数调用图:



~CommentNode()

```
virtual arcs::aubo_scope::CommentNode::~~CommentNode () [virtual]
```

CommentNode() [3/3]

```
arcs::aubo_scope::CommentNode::CommentNode () [private]
```

11.13.3 成员函数说明**getComment()**

```
std::string arcs::aubo_scope::CommentNode::getComment () const
```

返回

The text of the Comment node.

setComment()

```
void arcs::aubo_scope::CommentNode::setComment (
    const std::string & param_string)
```

- Set the text for the Comment node.

参数

<i>comment</i>	The Comment for the node.

返回

This node.

11.13.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [comment_node.h](#) 第 32 行定义。
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.13.5 类成员变量说明**d_**

```
void* arcs::aubo_scope::CommentNode::d_ { nullptr } [private]
```

在文件 [comment_node.h](#) 第 34 行定义。
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/comment_node.h](#)

11.14 arcs::aubo_scope::ContributionProvider< T > 模板类参考

贡献提供者此接口提供对贡献的访问。

```
#include <contribution_provider.h>
```

Public 成员函数

- virtual [~ContributionProvider](#) ()=default
- virtual T * [get](#) ()=0

获取一个贡献。在用户操作的结果中调用此方法以获取当前选中的节点。

11.14.1 详细描述

```
template<class T>
class arcs::aubo_scope::ContributionProvider< T >
```

贡献提供者此接口提供对贡献的访问。

注意：此接口不适用于具有基于 HTML 用户界面的贡献。

模板参数

< T>	表示接口提供访问的贡献类型的（通用）类型参数

在文件 [contribution_provider.h](#) 第 31 行定义.

11.14.2 构造及析构函数说明

```
~ContributionProvider()
```

```
template<class T>
virtual arcs::aubo_scope::ContributionProvider< T >::~~ContributionProvider () [virtual], [default]
```

11.14.3 成员函数说明

```
get()
```

```
template<class T>
virtual T * arcs::aubo_scope::ContributionProvider< T >::get () [pure virtual]
```

获取一个贡献。在用户操作的结果中调用此方法以获取当前选中的节点。

返回

相关的贡献

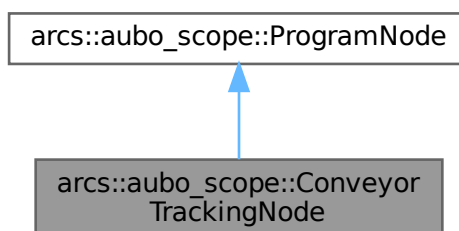
该类的文档由以下文件生成:

- [include/aubo_caps/contribution/contribution_provider.h](#)

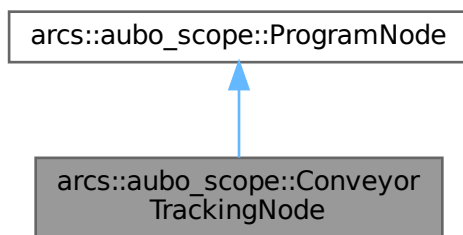
11.15 arcs::aubo_scope::ConveyorTrackingNode 类参考

```
#include <conveyor_tracking_node.h>
```

类 arcs::aubo_scope::ConveyorTrackingNode 继承关系图:



arcs::aubo_scope::ConveyorTrackingNode 的协作图:



Public 类型

- enum ConveyorTrackingTargetType { NEW_WORKPIECE , PREVIOUS_WORKPIECE }
- enum ConveyorStartActionQueueType { CLEAR_QUEUE , RESERVE_QUEUE , ASK }
- enum ConveyorStartActionType { CONTINUE_TRACKING , ABANDON_TRACKING }

Public 成员函数

- ConveyorTrackingNode (ConveyorTrackingNode &f)
- ConveyorTrackingNode (ConveyorTrackingNode &&f)
- virtual ~ConveyorTrackingNode ()

Public 成员函数继承自 arcs::aubo_scope::ProgramNode

- ProgramNode (ProgramNode &f)
- ProgramNode (ProgramNode &&f)
- virtual ~ProgramNode ()
- ProgramNodeContributionPtr getProgramNodeContribution ()
 获取程序节点贡献。不推荐用户直接调用 ProgramNodeContribution::generateScript, 应使用 Script←Writer::writeChildren。

Private 成员函数

- ConveyorTrackingNode ()

Private 属性

- void * d_ { nullptr }

友元

- class DataSwitch

额外继承的成员函数

Protected 成员函数继承自 arcs::aubo_scope::ProgramNode

- ProgramNode ()

11.15.1 详细描述

在文件 conveyor_tracking_node.h 第 10 行定义.

11.15.2 成员枚举类型说明

ConveyorStartActionQueueType

```
enum arcs::aubo_scope::ConveyorTrackingNode::ConveyorStartActionQueueType
```

枚举值

CLEAR_QUEUE	
RESERVE_QUEUE	
ASK	

在文件 [conveyor_tracking_node.h](#) 第 19 行定义.

ConveyorStartActionType

```
enum arcs::aubo_scope::ConveyorTrackingNode::ConveyorStartActionType
```

枚举值

CONTINUE_TRACKING	
ABANDON_TRACKING	

在文件 [conveyor_tracking_node.h](#) 第 26 行定义.

ConveyorTrackingTargetType

```
enum arcs::aubo_scope::ConveyorTrackingNode::ConveyorTrackingTargetType
```

枚举值

NEW_WORKPIECE	
PREVIOUS_WORKPIECE	

在文件 [conveyor_tracking_node.h](#) 第 13 行定义.

11.15.3 构造及析构函数说明

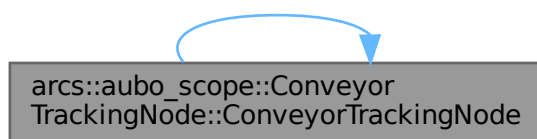
ConveyorTrackingNode() [1/3]

```
arcs::aubo_scope::ConveyorTrackingNode::ConveyorTrackingNode (  
    ConveyorTrackingNode & f)
```

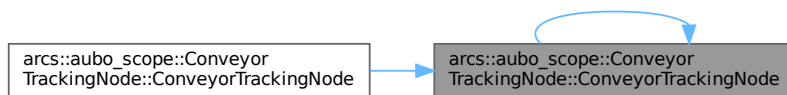
引用了 [ConveyorTrackingNode\(\)](#).

被这些函数引用 [ConveyorTrackingNode\(\)](#) , 以及 [ConveyorTrackingNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:



ConveyorTrackingNode() [2/3]

```
arcs::aubo_scope::ConveyorTrackingNode::ConveyorTrackingNode (
    ConveyorTrackingNode && f)
```

引用了 [ConveyorTrackingNode\(\)](#).

函数调用图:



~ConveyorTrackingNode()

```
virtual arcs::aubo_scope::ConveyorTrackingNode::~~ConveyorTrackingNode () [virtual]
```

ConveyorTrackingNode() [3/3]

```
arcs::aubo_scope::ConveyorTrackingNode::ConveyorTrackingNode () [private]
```

11.15.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [conveyor_tracking_node.h](#) 第 37 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.15.5 类成员变量说明

d_

```
void* arcs::aubo_scope::ConveyorTrackingNode::d_ { nullptr } [private]
```

在文件 [conveyor_tracking_node.h](#) 第 39 行定义。

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h](#)

11.16 arcs::aubo_scope::DataModel 类参考

数据模型此接口用于存储和管理表示当前配置的数据,例如[ProgramNodeContribution](#)或[InstallationNodeContribution](#)的配置。提供了在字典中添加、删除、获取和修改值的方法。

```
#include <data_model.h>
```

Public 成员函数

- [DataModel](#) (DataModel &f)
- [DataModel](#) (DataModel &&f)
- [~DataModel](#) ()
- void [set](#) (const std::string &key, bool value)
将 *bool* 值赋给指定键。
- bool [get](#) (const std::string &key, bool default_value) const
获取赋给指定键的 *bool* 值。
- void [set](#) (const std::string &key, int value)
将 *int* 值赋给指定键。
- int [get](#) (const std::string &key, int default_value) const
获取赋给指定键的 *int* 值。
- void [set](#) (const std::string &key, long value)
将 *long* 值赋给指定键。
- void [set](#) (const std::string &key, uint64_t value)
- long [get](#) (const std::string &key, long default_value) const
获取赋给指定键的 *long* 值。
- uint64_t [get](#) (const std::string &key, uint64_t default_value) const
- void [set](#) (const std::string &key, float value)
将 *float* 值赋给指定键。
- float [get](#) (const std::string &key, float default_value) const
获取赋给指定键的 *float* 值。
- void [set](#) (const std::string &key, double value)
将 *double* 值赋给指定键。
- double [get](#) (const std::string &key, double default_value) const
获取赋给指定键的 *double* 值。
- void [set](#) (const std::string &key, const std::string &value)
将 *std::string* 值赋给指定键。
- std::string [get](#) (const std::string &key, const std::string &default_value) const
获取赋给指定键的 *std::string* 值。
- void [set](#) (const std::string &key, VariablePtr value)
将 *Variable* 值赋给指定键。
- void [set](#) (const std::string &key, ExpressionPtr value)
- void [set](#) (const std::string &key, PayloadPtr value)
- void [set](#) (const std::string &key, IoPtr value)
- void [set](#) (const std::string &key, WaypointPtr value)
- VariablePtr [get](#) (const std::string &key, VariablePtr default_value) const

获取赋给指定键的 *Variable* 值。

- ExpressionPtr *get* (const std::string &key, ExpressionPtr default_value) const
- PayloadPtr *get* (const std::string &key, PayloadPtr default_value) const
- IoPtr *get* (const std::string &key, IoPtr default_value) const
- WaypointPtr *get* (const std::string &key, WaypointPtr default_value) const
- void *set* (const std::string &key, const std::vector< bool > &value)
将 *bool*[] 数组赋给指定键。
- std::vector< bool > *get* (const std::string &key, const std::vector< bool > &default_value) const
获取赋给指定键的 *bool*[] 数组。
- void *set* (const std::string &key, const std::vector< int > &value)
将 *int*[] 数组赋给指定键。
- std::vector< int > *get* (const std::string &key, const std::vector< int > &default_value) const
获取赋给指定键的 *int*[] 数组。
- void *set* (const std::string &key, const std::vector< long > &value)
将 *long*[] 数组赋给指定键。
- void *set* (const std::string &key, const std::vector< uint64_t > &value)
- std::vector< long > *get* (const std::string &key, const std::vector< long > &default_value) const
获取赋给指定键的 *long*[] 数组。
- std::vector< uint64_t > *get* (const std::string &key, const std::vector< uint64_t > &default_value) const
- void *set* (const std::string &key, const std::vector< float > &value)
将 *float*[] 数组赋给指定键。
- std::vector< float > *get* (const std::string &key, const std::vector< float > &default_value) const
获取赋给指定键的 *float*[] 数组。
- void *set* (const std::string &key, const std::vector< double > &value)
将 *double*[] 数组赋给指定键。
- std::vector< double > *get* (const std::string &key, const std::vector< double > &default_value) const
获取赋给指定键的 *double*[] 数组。
- void *set* (const std::string &key, const std::vector< std::string > &value)
将 *std::string*[] 数组赋给指定键。
- std::vector< std::string > *get* (const std::string &key, const std::vector< std::string > &default_value) const
获取赋给指定键的 *std::string*[] 数组。
- void *set* (const std::string &key, TCPPtr value)
将 *TCP* 值赋给指定键。
- TCPPtr *get* (const std::string &key, TCPPtr default_value) const
获取赋给指定键的 *TCP* 值。
- void *set* (const std::string &key, FeaturePtr value)
将 *Feature* 值赋给指定键。
- FeaturePtr *get* (const std::string &key, FeaturePtr default_value) const
获取赋给指定键的 *Feature* 值。
- std::set< std::string > *getKeys* () const
获取数据模型中所有键的集合。
- bool *isSet* (const std::string &key) const
检查数据模型中是否存在指定键。
- bool *remove* (const std::string &key)
从数据模型中移除一个键值对。

Private 成员函数

- *DataModel* ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.16.1 详细描述

数据模型此接口用于存储和管理表示当前配置的数据,例如[ProgramNodeContribution](#)或[InstallationNodeContribution](#)的配置。提供了在字典中添加、删除、获取和修改值的方法。

为已使用的键设置新值将覆盖当前值。无论值的类型如何都会发生(例如,在键 *myBool* 下存储值 *true*,之后又在同一键下存储一个 *Angle* 对象,将会用提供的 *Angle* 对象覆盖 *true* 值)。

每个 AuboCap 安装画面都有一个底层的[DataModel](#) 对象。该对象随每个 AuboScope 安装一起保存和加载。

同样,每个贡献的程序节点实例都有一个底层的[DataModel](#) 对象。该对象随节点所在程序一起保存和加载。基于 HTML 的程序节点贡献中,对[DataModel](#) 对象的所有修改都支持撤销/重做操作。基于 Qt 的 AuboStudio 插件必须使用[UndoRedoManager](#) 将更改记录到撤销/重做栈中。

检索对象时,键和对象类型都必须与之前存储的匹配。这意味着如果使用键 *myAngle* 存储了一个[TCP](#) 对象,然后尝试使用键 *myAngle* 通过 `get(std::string key, Angle default_value)` 检索它,将不会返回存储的值,因为类型不匹配。相反,将返回提供的 *default_value*。

在文件 [data_model.h](#) 第 82 行定义。

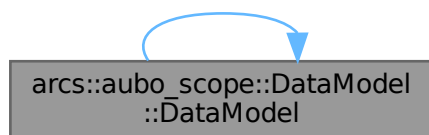
11.16.2 构造及析构函数说明**DataModel() [1/3]**

```
arcs::aubo_scope::DataModel::DataModel (
    DataModel & f)
```

引用了 [DataModel\(\)](#)。

被这些函数引用 [DataModel\(\)](#) , 以及 [DataModel\(\)](#)。

函数调用图:



这是这个函数的调用关系图:

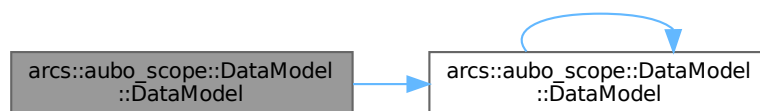


DataModel() [2/3]

```
arcs::aubo_scope::DataModel::DataModel (
    DataModel && f)
```

引用了 [DataModel\(\)](#).

函数调用图:

**~DataModel()**

```
arcs::aubo_scope::DataModel::~~DataModel ()
```

DataModel() [3/3]

```
arcs::aubo_scope::DataModel::DataModel () [private]
```

11.16.3 成员函数说明**get()** [1/21]

```
bool arcs::aubo_scope::DataModel::get (
    const std::string & key,
    bool default_value) const
```

获取赋给指定键的 `bool` 值。

参数

<i>key</i>	数据模型中的键。
<i>default_value</i>	键不存在时返回的值。

返回

赋给键的值。如果不存在，返回 `default_value`。

get() [2/21]

```
std::string arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::string & default_value) const
```

获取赋给指定键的 `std::string` 值。

get() [3/21]

```
std::vector< bool > arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::vector< bool > & default_value) const
```

获取赋给指定键的 bool[] 数组。

get() [4/21]

```
std::vector< double > arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::vector< double > & default_value) const
```

获取赋给指定键的 double[] 数组。

get() [5/21]

```
std::vector< float > arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::vector< float > & default_value) const
```

获取赋给指定键的 float[] 数组。

get() [6/21]

```
std::vector< int > arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::vector< int > & default_value) const
```

获取赋给指定键的 int[] 数组。

get() [7/21]

```
std::vector< long > arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::vector< long > & default_value) const
```

获取赋给指定键的 long[] 数组。

get() [8/21]

```
std::vector< std::string > arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::vector< std::string > & default_value) const
```

获取赋给指定键的 std::string[] 数组。

get() [9/21]

```
std::vector< uint64_t > arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::vector< uint64_t > & default_value) const
```

get() [10/21]

```
double arcs::aubo_scope::DataModel::get (
    const std::string & key,
    double default_value) const
```

获取赋给指定键的 double 值。

get() [11/21]

```
ExpressionPtr arcs::aubo_scope::DataModel::get (
    const std::string & key,
    ExpressionPtr default_value) const
```

get() [12/21]

```
FeaturePtr arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    FeaturePtr default_value) const
```

获取赋给指定键的 **Feature** 值。

get() [13/21]

```
float arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    float default_value) const
```

获取赋给指定键的 `float` 值。

get() [14/21]

```
int arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    int default_value) const
```

获取赋给指定键的 `int` 值。

参数

<i>key</i>	数据模型中的键。
<i>default_value</i>	键不存在时返回的值。

返回

赋给键的值。

get() [15/21]

```
IoPtr arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    IoPtr default_value) const
```

get() [16/21]

```
long arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    long default_value) const
```

获取赋给指定键的 `long` 值。

get() [17/21]

```
PayloadPtr arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    PayloadPtr default_value) const
```

get() [18/21]

```
TCPPtr arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    TCPPtr default_value) const
```

获取赋给指定键的 **TCP** 值。

get() [19/21]

```
uint64_t arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    uint64_t default_value) const
```

get() [20/21]

```
VariablePtr arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    VariablePtr default_value) const  
获取赋给指定键的Variable值。
```

返回

赋给键的值。

get() [21/21]

```
WaypointPtr arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    WaypointPtr default_value) const
```

getKeys()

```
std::set< std::string > arcs::aubo_scope::DataModel::getKeys () const  
获取数据模型中所有键的集合。
```

返回

键的集合。

isSet()

```
bool arcs::aubo_scope::DataModel::isSet (  
    const std::string & key) const  
检查数据模型中是否存在指定键。
```

参数

<i>key</i>	数据模型中的键。

返回

如果键存在返回 `true`，否则返回 `false`。

remove()

```
bool arcs::aubo_scope::DataModel::remove (  
    const std::string & key)  
从数据模型中移除一个键值对。
```

参数

<i>key</i>	数据模型中的键。

返回

成功返回 `true`，否则返回 `false`。

异常

<i>IllegalStateException</i>	如果在Qt-based AuboCap 程序节点中调用时不在UndoableChanges范围内。

set() [1/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    bool value)
```

将 `bool` 值赋给指定键。

参数

<i>key</i>	数据模型中的键（不可为 <code>null</code> 或空 <code>std::string</code> ）。
<i>value</i>	赋给键的值。

异常

<i>IllegalArgumentException</i>	如果键为 <code>null</code> 或空 <code>std::string</code> 。
<i>IllegalStateException</i>	如果在Qt-based AuboCap 程序节点中调用时不在UndoableChanges范围内（另见 UndoRedoManager ）。

set() [2/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    const std::string & value)
```

将 `std::string` 值赋给指定键。

set() [3/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    const std::vector< bool > & value)
```

将 `bool []` 数组赋给指定键。

set() [4/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    const std::vector< double > & value)
```

将 double[] 数组赋给指定键。

set() [5/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    const std::vector< float > & value)
```

将 float[] 数组赋给指定键。

set() [6/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    const std::vector< int > & value)
```

将 int[] 数组赋给指定键。

set() [7/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    const std::vector< long > & value)
```

将 long[] 数组赋给指定键。

set() [8/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    const std::vector< std::string > & value)
```

将 std::string[] 数组赋给指定键。

set() [9/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    const std::vector< uint64_t > & value)
```

set() [10/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    double value)
```

将 double 值赋给指定键。

set() [11/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    ExpressionPtr value)
```

set() [12/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    FeaturePtr value)
```

将 [Feature](#) 值赋给指定键。

set() [13/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    float value)
```

将 float 值赋给指定键。

set() [14/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    int value)
```

将 int 值赋给指定键。

参数

<i>key</i>	数据模型中的键。
<i>value</i>	赋给键的值。

异常

<i>IllegalArgumentException</i>	如果键为 null 或空字符串。
<i>IllegalStateException</i>	如果不在UndoableChanges范围内调用。

set() [15/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    IoPtr value)
```

set() [16/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    long value)
```

将 long 值赋给指定键。

set() [17/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    PayloadPtr value)
```

set() [18/21]

```
void arcs::aubo_scope::DataModel::set (  
    const std::string & key,  
    TCPPtr value)
```

将TCP值赋给指定键。

set() [19/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    uint64_t value)
```

set() [20/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    VariablePtr value)
```

将Variable值赋给指定键。

set() [21/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    WaypointPtr value)
```

11.16.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [data_model.h](#) 第 708 行定义。
引用了 [DataSwitch](#)。
被这些函数引用 [DataSwitch](#)。

11.16.5 类成员变量说明

d_

```
void* arcs::aubo_scope::DataModel::d_ { nullptr } [private]
```

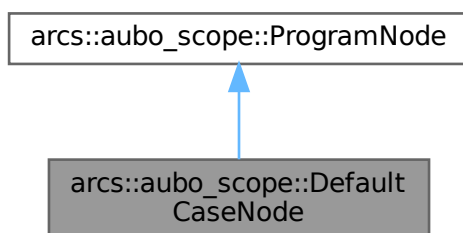
在文件 [data_model.h](#) 第 710 行定义。
该类的文档由以下文件生成:

- [include/aubo_caps/domain/data/data_model.h](#)

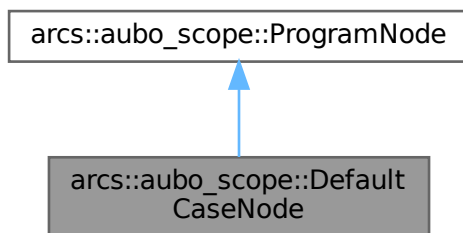
11.17 arcs::aubo_scope::DefaultCaseNode 类参考

```
#include <default_case_node.h>
```

类 arcs::aubo_scope::DefaultCaseNode 继承关系图:



arcs::aubo_scope::DefaultCaseNode 的协作图:



Public 成员函数

- [DefaultCaseNode](#) (DefaultCaseNode &f)
- [DefaultCaseNode](#) (DefaultCaseNode &&f)
- virtual [~DefaultCaseNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [DefaultCaseNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.17.1 详细描述

在文件 [default_case_node.h](#) 第 10 行定义.

11.17.2 构造及析构函数说明

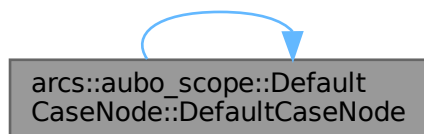
DefaultCaseNode() [1/3]

```
arcs::aubo_scope::DefaultCaseNode::DefaultCaseNode (  
    DefaultCaseNode & f)
```

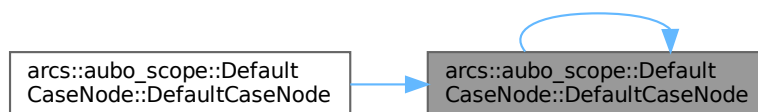
引用了 [DefaultCaseNode\(\)](#).

被这些函数引用 [DefaultCaseNode\(\)](#) , 以及 [DefaultCaseNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

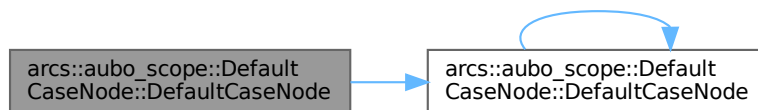


DefaultCaseNode() [2/3]

```
arcs::aubo_scope::DefaultCaseNode::DefaultCaseNode (  
    DefaultCaseNode && f)
```

引用了 [DefaultCaseNode\(\)](#).

函数调用图:



~DefaultCaseNode()

```
virtual arcs::aubo_scope::DefaultCaseNode::~~DefaultCaseNode () [virtual]
```

DefaultCaseNode() [3/3]

```
arcs::aubo_scope::DefaultCaseNode::DefaultCaseNode () [private]
```

11.17.3 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [default_case_node.h](#) 第 18 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.17.4 类成员变量说明**d_**

```
void* arcs::aubo_scope::DefaultCaseNode::d_ { nullptr } [private]
```

在文件 [default_case_node.h](#) 第 20 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/default_case_node.h](#)

11.18 arcs::aubo_scope::DefaultSingletonTraits< Type > 模板结构体参考**11.18.1 详细描述**

```
template<typename Type>
```

```
struct arcs::aubo_scope::DefaultSingletonTraits< Type >
```

在文件 [popup_manager.h](#) 第 12 行定义.

该结构体的文档由以下文件生成:

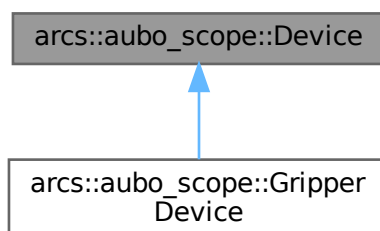
- [include/aubo_caps/domain/user_interface/popup_manager.h](#)

11.19 arcs::aubo_scope::Device 类参考

设备接口此基础接口表示 AuboScope 支持的设备，例如夹爪设备。

```
#include <device.h>
```

类 arcs::aubo_scope::Device 继承关系图:



Public 成员函数

- virtual `~Device()`=0
- virtual `AuboCapInfoPtr getContributorInfo()`=0
获取将此夹具设备添加/贡献到 *AuboScope* 的 *AuboCap* 信息。
- virtual `std::string getId()`=0
- virtual `std::string getDisplayName()`=0
- virtual `bool isResolvable()`=0
设备无法保证始终存在于 *AuboScope* 中。如果添加设备的 *AuboCap* 未安装，设备将不可用。此方法用于判断设备是否可用。

11.19.1 详细描述

设备接口此基础接口表示 *AuboScope* 支持的设备，例如夹具设备。
在文件 `device.h` 第 18 行定义。

11.19.2 构造及析构函数说明

`~Device()`

```
virtual arcs::aubo_scope::Device::~Device() [pure virtual]
```

11.19.3 成员函数说明

`getContributorInfo()`

```
virtual AuboCapInfoPtr arcs::aubo_scope::Device::getContributorInfo() [pure virtual]
```

获取将此夹具设备添加/贡献到 *AuboScope* 的 *AuboCap* 信息。

返回

此夹具设备的 *AuboCap* 信息。

`getDisplayName()`

```
virtual std::string arcs::aubo_scope::Device::getDisplayName() [pure virtual]
```

AuboScope UI 中显示的夹具名称，例如在夹具工具栏和程序节点中。

注意：此名称不应用于唯一标识特定的单个夹具，因为名称可能会被翻译（取决于 *AuboScope* 中选择的语言）或在添加夹具的 *AuboCap* 版本之间发生更改。

返回

此夹具在 *AuboScope* 中显示的名称。

`getId()`

```
virtual std::string arcs::aubo_scope::Device::getId() [pure virtual]
```

获取此夹具的唯一标识符 (id)。可用于在特定多夹具设备中区分不同的夹具。
注意此 id 并非“全局”唯一，仅对属于特定多夹具设备的各个夹具唯一。

返回

此夹具的唯一标识符。

isResolvable()

```
virtual bool arcs::aubo_scope::Device::isResolvable () [pure virtual]
```

设备无法保证始终存在于AuboScope中。如果添加设备的AuboCap未安装，设备将不可用。此方法用于判断设备是否可用。

返回

如果此夹爪设备存在于AuboScope中返回 true，否则返回 false。

该类的文档由以下文件生成:

- include/aubo_caps/domain/device/[device.h](#)

11.20 arcs::aubo_scope::DeviceManager 类参考

设备管理器设备管理器的基础接口。

```
#include <device_manager.h>
```

Public 成员函数

- virtual [~DeviceManager](#) ()=default
- virtual std::vector< GripperDevicePtr > [getGrippers](#) ()=0
获取AuboScope中可用的夹爪设备列表
- virtual std::vector< DevicePtr > [getScrewDrivers](#) ()=0
获取AuboScope中可用的螺丝刀设备列表

11.20.1 详细描述

设备管理器设备管理器的基础接口。

设备管理器为AuboScope支持的特定类型设备提供功能。例如，设备管理器可以提供访问所有已注册设备（该类型）以及在程序中使用特定设备的功能。

在文件 [device_manager.h](#) 第 30 行定义。

11.20.2 构造及析构函数说明**~DeviceManager()**

```
virtual arcs::aubo_scope::DeviceManager::~DeviceManager () [virtual], [default]
```

11.20.3 成员函数说明**getGrippers()**

```
virtual std::vector< GripperDevicePtr > arcs::aubo_scope::DeviceManager::getGrippers () [pure virtual]
```

获取AuboScope中可用的夹爪设备列表

返回

夹爪设备列表。

getScrewDrivers()

```
virtual std::vector< DevicePtr > arcs::aubo_scope::DeviceManager::getScrewDrivers () [pure virtual]
```

获取AuboScope中可用的螺丝刀设备列表

返回

螺丝刀设备列表。

该类的文档由以下文件生成:

- include/aubo_caps/domain/device/[device_manager.h](#)

11.21 arcs::aubo_scope::DeviceRegistrationManager 类参考

设备注册管理器设备注册管理器的基础接口。

```
#include <device_registration_manager.h>
```

Public 成员函数

- virtual [~DeviceRegistrationManager](#) ()=default
- virtual void [registerGripperContribution](#) (GripperContributionPtr contribution)=0
注册夹爪贡献
- virtual void [registerScrewDriverContribution](#) (ScrewdriverContributionPtr contribution)=0
注册螺丝刀贡献

11.21.1 详细描述

设备注册管理器设备注册管理器的基础接口。

设备注册管理器用于将程序/安装节点贡献或服务注册为符合 AuboScope 支持的特定设备类型。
在文件 [device_registration_manager.h](#) 第 27 行定义。

11.21.2 构造及析构函数说明

~DeviceRegistrationManager()

```
virtual arcs::aubo_scope::DeviceRegistrationManager::~~DeviceRegistrationManager () [virtual], [default]
```

11.21.3 成员函数说明

registerGripperContribution()

```
virtual void arcs::aubo_scope::DeviceRegistrationManager::registerGripperContribution (  
    GripperContributionPtr contribution) [pure virtual]  
注册夹爪贡献
```

参数

<i>contribution</i>	夹爪贡献指针。

registerScrewDriverContribution()

```
virtual void arcs::aubo_scope::DeviceRegistrationManager::registerScrewDriverContribution (  
    ScrewdriverContributionPtr contribution) [pure virtual]  
注册螺丝刀贡献
```

参数

<i>contribution</i>	螺丝刀贡献指针。

该类的文档由以下文件生成:

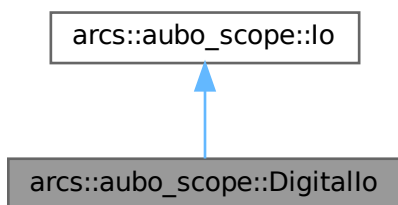
- include/aubo_caps/domain/device/[device_registration_manager.h](#)

11.22 arcs::aubo_scope::DigitalIo 类参考

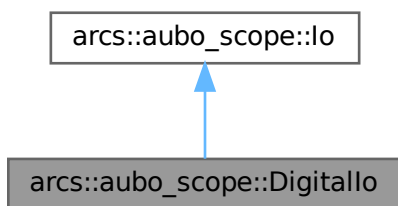
数字 IO 接口此接口提供对数字 I/O 的支持。

```
#include <digital_io.h>
```

类 arcs::aubo_scope::DigitalIo 继承关系图:



arcs::aubo_scope::DigitalIo 的协作图:



Public 成员函数

- `DigitalIo` (`DigitalIo &f`)
- `DigitalIo` (`DigitalIo &&f`)
- `~DigitalIo` ()
- `bool setValue` (`bool high`)
设置数字I/O 输出信号。注意该I/O 必须是输出类型。
- `bool getValue` ()
获取I/O 的最后一次读数

Public 成员函数继承自 arcs::aubo_scope::Io

- `Io` (`Io &f`)
- `Io` (`Io &&f`)
- `virtual ~Io` ()
- `std::string getDisplayName` () `const`
获取当前安装中I/O 的用户自定义名称
- `std::string getDefaultName` () `const`
获取I/O 的机器人默认名称

- `std::string getValueStr () const`
获取I/O 的当前读数值（字符串形式）
- `IoType getType () const`
获取I/O 类型
- `InterfaceType getInterfaceType () const`
获取此I/O 所属的接口类型
- `int getAction () const`
获取I/O 的动作编号
- `bool isInput () const`
判断I/O 是否支持读取值
- `bool isResolvable () const`
某些I/O（如MODBUS I/O 和模拟工具输入）在某些情况下可能不可用。此方法用于判断 I/O 是否可用。
- `size_t hashCode ()`

Private 成员函数

- `DigitalIo ()`

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数

Public 类型继承自 `arcs::aubo_scope::Io`

- `enum IoType : int {
DIGITAL , ANALOG , INTEGER , BOOLEAN ,
DOUBLE }`
- `enum InterfaceType : int {
STANDARD , TOOL , CONFIGURABLE , LINKIO ,
MODBUS , GENERAL_PURPOSE , PROFINET , ETHERNET_IP }`

Protected 成员函数继承自 `arcs::aubo_scope::Io`

- `Io ()`

11.22.1 详细描述

数字 IO 接口此接口提供对数字 I/O 的支持。
在文件 `digital_io.h` 第 20 行定义.

11.22.2 构造及析构函数说明

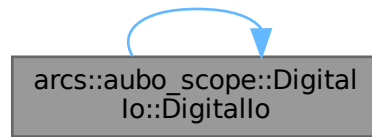
DigitalIo() [1/3]

```
arcs::aubo_scope::DigitalIo::DigitalIo (
    DigitalIo & f)
```

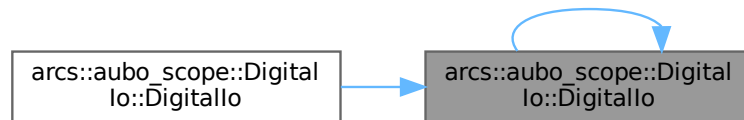
引用了 `DigitalIo()`.

被这些函数引用 `DigitalIo()` , 以及 `DigitalIo()`.

函数调用图:



这是这个函数的调用关系图:

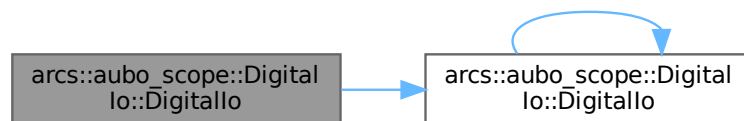


DigitalIo() [2/3]

```
arcs::aubo_scope::DigitalIo::DigitalIo (
    DigitalIo && f)
```

引用了 [DigitalIo\(\)](#).

函数调用图:



~DigitalIo()

```
arcs::aubo_scope::DigitalIo::~~DigitalIo ()
```

DigitalIo() [3/3]

```
arcs::aubo_scope::DigitalIo::DigitalIo () [private]
```


11.22.3 成员函数说明

getValue()

```
bool arcs::aubo_scope::DigitalIo::getValue ()
```

获取I/O 的最后一次读数

返回

I/O 的最后一次读数。

setValue()

```
bool arcs::aubo_scope::DigitalIo::setValue (
    bool high)
```

设置数字I/O 输出信号。注意该I/O 必须是输出类型。

参数

<i>high</i>	要设置的数字值。

返回

如果数字信号设置成功返回 `true`，否则返回 `false`。

11.22.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [digital_io.h](#) 第 57 行定义。
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.22.5 类成员变量说明

d_

```
void* arcs::aubo_scope::DigitalIo::d_ { nullptr } [private]
```

在文件 [digital_io.h](#) 第 59 行定义。
该类的文档由以下文件生成:

- [include/aubo_caps/domain/io/digital_io.h](#)

11.23 arcs::aubo_scope::DigitalIoFilter 结构体参考

```
#include <digital_io.h>
```

Public 成员函数

- [DigitalIoFilter](#) ([Io::InterfaceType](#) interface, bool is_input)
- bool [operator\(\)](#) (IoPtr io)

Private 属性

- [Io::InterfaceType](#) interface_
- bool is_input_

11.23.1 详细描述

在文件 `digital_io.h` 第 62 行定义.

11.23.2 构造及析构造函数说明

DigitalIoFilter()

```
arcs::aubo_scope::DigitalIoFilter::DigitalIoFilter (
    Io::InterfaceType interface,
    bool is_input) [inline]
```

在文件 `digital_io.h` 第 64 行定义.

引用了 `interface_`, 以及 `is_input_`.

11.23.3 成员函数说明

operator()()

```
bool arcs::aubo_scope::DigitalIoFilter::operator() (
    IoPtr io) [inline]
```

在文件 `digital_io.h` 第 68 行定义.

引用了 `interface_`, 以及 `is_input_`.

11.23.4 类成员变量说明

interface_

```
Io::InterfaceType arcs::aubo_scope::DigitalIoFilter::interface_ [private]
```

在文件 `digital_io.h` 第 76 行定义.

被这些函数引用 `DigitalIoFilter()`, 以及 `operator()()`.

is_input_

```
bool arcs::aubo_scope::DigitalIoFilter::is_input_ [private]
```

在文件 `digital_io.h` 第 77 行定义.

被这些函数引用 `DigitalIoFilter()`, 以及 `operator()()`.

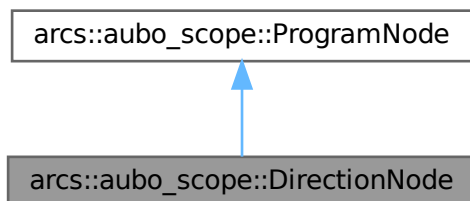
该结构体的文档由以下文件生成:

- `include/aubo_caps/domain/io/digital_io.h`

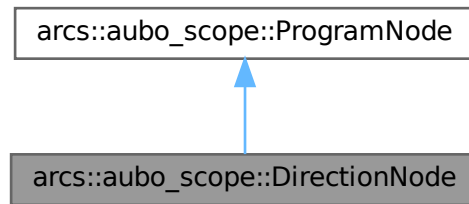
11.24 arcs::aubo_scope::DirectionNode 类参考

```
#include <direction_node.h>
```

类 `arcs::aubo_scope::DirectionNode` 继承关系图:



arcs::aubo_scope::DirectionNode 的协作图:



Public 类型

- enum `ParameterType` : int { `SHARED_MOTION` , `TOOL_MOTION` }
- enum `Axis` : int {
 `X_PLUS` , `X_MINUS` , `Y_PLUS` , `Y_MINUS` ,
 `Z_PLUS` , `Z_MINUS` }
- enum `DirectionSelectionType` : int { `EXPRESSION` , `AXIS` }
- enum `FeatureSelectionType` : int { `UNRESOLVED_FEATURE` , `SHARED` , `FEATURE` }

Public 成员函数

- `DirectionNode` (`DirectionNode &f`)
- `DirectionNode` (`DirectionNode &&f`)
- virtual `~DirectionNode` ()
- `ParameterType` `getType` ()
- void `setType` (`ParameterType` type)
- `Axis` `getAxis` ()
- void `setAxis` (`Axis` axis)
- `FeaturePtr` `getFeature` ()
- void `setFeature` (`FeaturePtr` feature)
- `DirectionSelectionType` `getDirectionSelectionType` ()
- void `setDirectionSelectionType` (`DirectionSelectionType` selection)
- `ExpressionPtr` `getExpression` ()
- void `setExpression` (`ExpressionPtr` expression)
- `FeatureSelectionType` `getFeatureSelectionType` ()
- void `setFeatureSelectionType` (`FeatureSelectionType` selection)
- double `getToolSpeed` ()
- void `setToolSpeed` (double speed)
- double `getToolAcceleration` ()
- void `setToolAcceleration` (double acc)

Public 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` (`ProgramNode &f`)
- `ProgramNode` (`ProgramNode &&f`)
- virtual `~ProgramNode` ()
- `ProgramNodeContributionPtr` `getProgramNodeContribution` ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `ScriptWriter::writeChildren`。

Private 成员函数

- `DirectionNode ()`

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数**Protected 成员函数继承自 `arcs::aubo_scope::ProgramNode`**

- `ProgramNode ()`

11.24.1 详细描述

在文件 `direction_node.h` 第 12 行定义.

11.24.2 成员枚举类型说明**Axis**

```
enum arcs::aubo_scope::DirectionNode::Axis : int
```

枚举值

X_PLUS	Move in the positive X direction relative to the feature.
X_MINUS	Move in the negative X direction relative to the feature.
Y_PLUS	Move in the positive Y direction relative to the feature.
Y_MINUS	Move in the negative Y direction relative to the feature.
Z_PLUS	Move in the positive Z direction relative to the feature.
Z_MINUS	Move in the negative Z direction relative to the feature.

在文件 `direction_node.h` 第 33 行定义.

DirectionSelectionType

```
enum arcs::aubo_scope::DirectionNode::DirectionSelectionType : int
```

枚举值

EXPRESSION	A direction vector is specified using an Expression . The Selection instance can be cast to <code>DirectionVectorExpression</code>
AXIS	Direction is specified using an Axis . The Selection instance can be cast to <code>DirectionAxis</code>

在文件 `direction_node.h` 第 66 行定义.

FeatureSelectionType

```
enum arcs::aubo_scope::DirectionNode::FeatureSelectionType : int
```

枚举值

UNRESOLVED_FEATURE	The selected feature is no longer available (e.g. because the user has removed the feature in the Installation or the needed Installation is not currently loaded)
SHARED	Feature is inherited from the MoveL or MoveP parent.
FEATURE	Feature is selected. The Selection instance can be cast to DirectionNodeFeature

在文件 `direction_node.h` 第 82 行定义.

ParameterType

```
enum arcs::aubo_scope::DirectionNode::ParameterType : int
```

枚举值

SHARED_MOTION	The tool motion parameters are inherited from the MoveL or MoveP parent.
TOOL_MOTION	The tool motion parameters are selected. The Selection instance can be cast to ToolMotionParameters.

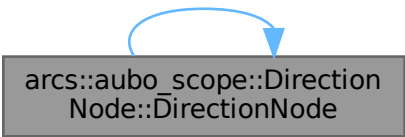
在文件 `direction_node.h` 第 15 行定义.

11.24.3 构造及析构函数说明

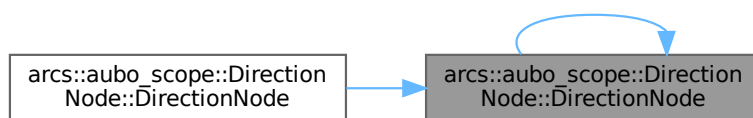
DirectionNode() [1/3]

```
arcs::aubo_scope::DirectionNode::DirectionNode (  
    DirectionNode & f)
```

引用了 `DirectionNode()`.
被这些函数引用 `DirectionNode()` , 以及 `DirectionNode()`.
函数调用图:



这是这个函数的调用关系图:

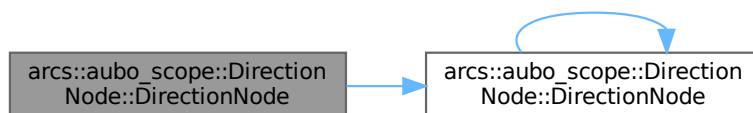


DirectionNode() [2/3]

```
arcs::aubo_scope::DirectionNode::DirectionNode (
    DirectionNode && f)
```

引用了 [DirectionNode\(\)](#).

函数调用图:



~DirectionNode()

```
virtual arcs::aubo_scope::DirectionNode::~~DirectionNode () [virtual]
```

DirectionNode() [3/3]

```
arcs::aubo_scope::DirectionNode::DirectionNode () [private]
```

11.24.4 成员函数说明

getAxis()

```
Axis arcs::aubo_scope::DirectionNode::getAxis ()
```

返回

the [Axis](#) used to define the direction.

getDirectionSelectionType()

```
DirectionSelectionType arcs::aubo_scope::DirectionNode::getDirectionSelectionType ()
```

getExpression()

```
ExpressionPtr arcs::aubo_scope::DirectionNode::getExpression ()
```

返回

the expression used to define the direction vector.

getFeature()

```
FeaturePtr arcs::aubo_scope::DirectionNode::getFeature ()
```

返回

the feature selected for this Direction node.

getFeatureSelectionType()

```
FeatureSelectionType arcs::aubo_scope::DirectionNode::getFeatureSelectionType ()
```

getToolAcceleration()

```
double arcs::aubo_scope::DirectionNode::getToolAcceleration ()
```

返回

the tool acceleration to use for the movement

getToolSpeed()

```
double arcs::aubo_scope::DirectionNode::getToolSpeed ()
```

返回

the tool speed to be achieved.

getType()

```
ParameterType arcs::aubo_scope::DirectionNode::getType ()
```

setAxis()

```
void arcs::aubo_scope::DirectionNode::setAxis (  
    Axis axis)
```

setDirectionSelectionType()

```
void arcs::aubo_scope::DirectionNode::setDirectionSelectionType (  
    DirectionSelectionType selection)
```

setExpression()

```
void arcs::aubo_scope::DirectionNode::setExpression (  
    ExpressionPtr expression)
```

setFeature()

```
void arcs::aubo_scope::DirectionNode::setFeature (  
    FeaturePtr feature)
```

setFeatureSelectionType()

```
void arcs::aubo_scope::DirectionNode::setFeatureSelectionType (  
    FeatureSelectionType selection)
```

setToolAcceleration()

```
void arcs::aubo_scope::DirectionNode::setToolAcceleration (  
    double acc)
```

setToolSpeed()

```
void arcs::aubo_scope::DirectionNode::setToolSpeed (  
    double speed)
```

setType()

```
void arcs::aubo_scope::DirectionNode::setType (  
    ParameterType type)
```

11.24.5 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [direction_node.h](#) 第 147 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.24.6 类成员变量说明

d_

```
void* arcs::aubo_scope::DirectionNode::d_ { nullptr } [private]
```

在文件 [direction_node.h](#) 第 149 行定义.

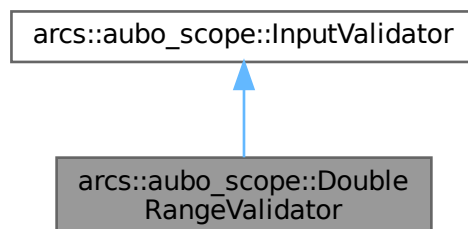
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/direction_node.h](#)

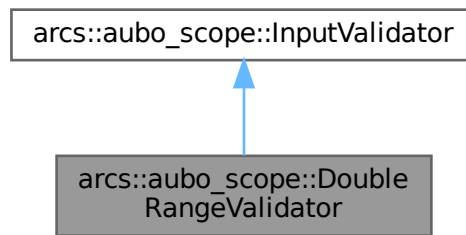
11.25 arcs::aubo_scope::DoubleRangeValidator 结构体参考

```
#include <input_validator.h>
```

类 arcs::aubo_scope::DoubleRangeValidator 继承关系图:



arcs::aubo_scope::DoubleRangeValidator 的协作图:



Public 成员函数

- [DoubleRangeValidator](#) (double min**Value**, double max**Value**, bool is_min_inclusive=true, bool is_max_inclusive=true)
- [DoubleRangeValidator](#) (std::function< std::pair< double, double >()> range_func)
- [DoubleRangeValidator](#) (std::function< std::tuple< double, double, bool, bool >()> range_func)
- bool [isValid](#) (std::string value) override
验证输入值是否有效。
- std::string [getMessage](#) (std::string value) override
当值无效时返回有意义的提示信息。

Private 成员函数

- void [updateValue](#) ()

Private 属性

- double [min_value](#) { DBL_MIN }
- double [max_value](#) { DBL_MAX }
- std::function< std::pair< double, double >()> [range_func](#) { nullptr }
- std::function< std::tuple< double, double, bool, bool >()> [range_func_with_flags](#) { nullptr }
- bool [is_min_inclusive](#) { true }
- bool [is_max_inclusive](#) { true }

11.25.1 详细描述

在文件 [input_validator.h](#) 第 118 行定义.

11.25.2 构造及析构函数说明

DoubleRangeValidator() [1/3]

```
arcs::aubo_scope::DoubleRangeValidator::DoubleRangeValidator (
    double minValue,
    double maxValue,
    bool is_min_inclusive = true,
    bool is_max_inclusive = true)
```

DoubleRangeValidator() [2/3]

```
arcs::aubo_scope::DoubleRangeValidator::DoubleRangeValidator (
    std::function< std::pair< double, double >()> range_func)
```

DoubleRangeValidator() [3/3]

```
arcs::aubo_scope::DoubleRangeValidator::DoubleRangeValidator (
    std::function< std::tuple< double, double, bool, bool >()> range_func)
```

11.25.3 成员函数说明**getMessage()**

```
std::string arcs::aubo_scope::DoubleRangeValidator::getMessage (
    std::string value) [override], [virtual]
```

当值无效时返回有意义的提示信息。

参数

<i>value</i>	无效的值，如果合理可包含在消息中

返回

提示信息

实现了 [arcs::aubo_scope::InputValidator](#).

isValid()

```
bool arcs::aubo_scope::DoubleRangeValidator::isValid (
    std::string value) [override], [virtual]
```

验证输入值是否有效。

参数

<i>value</i>	需要验证的值

返回

如果值有效则返回 `true`

实现了 [arcs::aubo_scope::InputValidator](#).

updateValue()

```
void arcs::aubo_scope::DoubleRangeValidator::updateValue () [private]
```

11.25.4 类成员变量说明**is_max_inclusive__**

```
bool arcs::aubo_scope::DoubleRangeValidator::is_max_inclusive_ { true } [private]
```

在文件 [input_validator.h](#) 第 143 行定义.

is_min_inclusive__

```
bool arcs::aubo_scope::DoubleRangeValidator::is_min_inclusive_ { true } [private]
```

在文件 [input_validator.h](#) 第 142 行定义.

max_value__

```
double arcs::aubo_scope::DoubleRangeValidator::max_value_ { DBL_MAX } [private]
```

在文件 [input_validator.h](#) 第 138 行定义.

min_value__

```
double arcs::aubo_scope::DoubleRangeValidator::min_value_ { DBL_MIN } [private]
```

在文件 [input_validator.h](#) 第 137 行定义.

range_func__

```
std::function<std::pair<double, double>()> arcs::aubo_scope::DoubleRangeValidator::range_func_ { nullptr } [private]
```

在文件 [input_validator.h](#) 第 139 行定义.

range_func_with_flags__

```
std::function<std::tuple<double, double, bool, bool>()> arcs::aubo_scope::DoubleRangeValidator::range_func_with_flags_ { nullptr } [private]
```

在文件 [input_validator.h](#) 第 141 行定义.

该结构体的文档由以下文件生成:

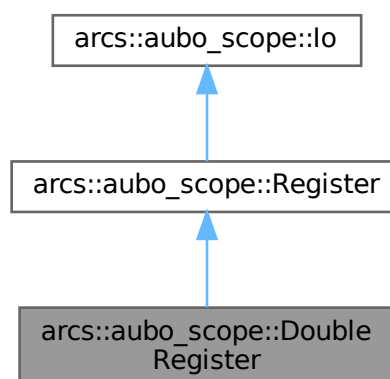
- [include/aubo_caps/domain/user_interface/input_validator.h](#)

11.26 arcs::aubo_scope::DoubleRegister 类参考

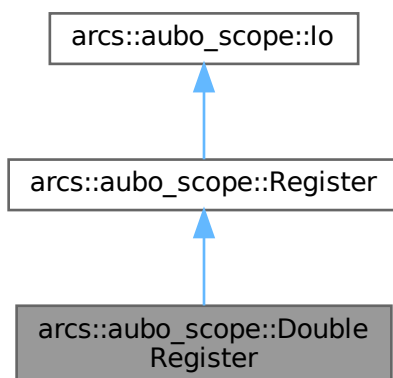
This interface provides support for float registers.

```
#include <double_register.h>
```

类 arcs::aubo_scope::DoubleRegister 继承关系图:



arcs::aubo_scope::DoubleRegister 的协作图:



Public 成员函数

- virtual `~DoubleRegister ()`=default

Public 成员函数继承自 `arcs::aubo_scope::Register`

- virtual `~Register ()`=default

Public 成员函数继承自 `arcs::aubo_scope::Io`

- `Io (Io &f)`
- `Io (Io &&f)`
- virtual `~Io ()`
- `std::string getDisplayName () const`
获取当前安装中I/O的用户自定义名称
- `std::string getDefaultName () const`
获取I/O的机器人默认名称
- `std::string getValueStr () const`
获取I/O的当前读数值 (字符串形式)
- `IoType getType () const`
获取I/O类型
- `InterfaceType getInterfaceType () const`
获取此I/O所属的接口类型
- `int getAction () const`
获取I/O的动作编号
- `bool isInput () const`
判断I/O是否支持读取值
- `bool isResolvable () const`
某些I/O (如MODBUS I/O和模拟工具输入) 在某些情况下可能不可用。此方法用于判断I/O是否可用。
- `size_t hashCode ()`

额外继承的成员函数

Public 类型继承自 [arcs::aubo_scope::Io](#)

- enum [IoType](#) : int {
DIGITAL , ANALOG , INTEGER , BOOLEAN ,
DOUBLE }
- enum [InterfaceType](#) : int {
STANDARD , TOOL , CONFIGURABLE , LINKIO ,
MODBUS , GENERAL_PURPOSE , PROFINET , ETHERNET_IP }

Protected 成员函数继承自 [arcs::aubo_scope::Io](#)

- [Io](#) ()

11.26.1 详细描述

This interface provides support for float registers.
在文件 [double_register.h](#) 第 14 行定义.

11.26.2 构造及析构函数说明

~DoubleRegister()

```
virtual arcs::aubo_scope::DoubleRegister::~~DoubleRegister () [virtual], [default]
```

该类的文档由以下文件生成:

- [include/aubo_caps/domain/io/double_register.h](#)

11.27 arcs::aubo_scope::DriveScrewNotOKParameters 类参考

```
#include <screw_driver_configuration.h>
```

Public 成员函数

- virtual [~DriveScrewNotOKParameters](#) ()=default

11.27.1 详细描述

This interface provides parameters relevant for generating script code for the Drive Screw Not OK feed-back capability (registered using [ScrewdriverFeedbackCapabilities#registerDriveScrewNotOKCapability](#)(ScriptC

These parameters are passed when AuboScope calls the implementation of the [ScriptCodeGenerator#generateScript](#) method responsible for the generating the script code.

The interface is intentionally empty, but could have parameters in future releases.

在文件 [screw_driver_configuration.h](#) 第 107 行定义.

11.27.2 构造及析构函数说明

~DriveScrewNotOKParameters()

```
virtual arcs::aubo_scope::DriveScrewNotOKParameters::~~DriveScrewNotOKParameters () [virtual], [default]
```

该类的文档由以下文件生成:

- [include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h](#)

11.28 arcs::aubo_scope::DriveScrewOKParameters 类参考

```
#include <screw_driver_configuration.h>
```

Public 成员函数

- virtual [~DriveScrewOKParameters](#) ()=default

11.28.1 详细描述

This interface provides parameters relevant for generating script code for the Drive Screw OK feedback capability (registered using `ScrewdriverFeedbackCapabilities#registerDriveScrewOKCapability`(ScriptCodeGener

These parameters are passed when AuboScope calls the implementation of the `ScriptCodeGenerator#generateScript` method responsible for the generating the script code.

The interface is intentionally empty, but could have parameters in future releases.

在文件 [screw_driver_configuration.h](#) 第 130 行定义.

11.28.2 构造及析构造函数说明

`~DriveScrewOKParameters()`

```
virtual arcs::aubo_scope::DriveScrewOKParameters::~~DriveScrewOKParameters () [virtual], [default]
```

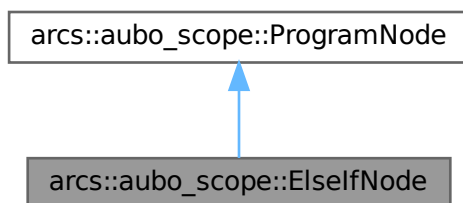
该类的文档由以下文件生成:

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_configuration.h](#)

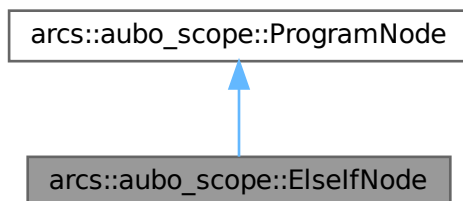
11.29 arcs::aubo_scope::ElseIfNode 类参考

```
#include <else_if_node.h>
```

类 arcs::aubo_scope::ElseIfNode 继承关系图:



arcs::aubo_scope::ElseIfNode 的协作图:



Public 成员函数

- [ElseIfNode](#) (ElseIfNode &f)
- [ElseIfNode](#) (ElseIfNode &&f)
- virtual [~ElseIfNode](#) ()
- void [setExpression](#) (ExpressionPtr expression)
- ExpressionPtr [getExpression](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *Script←Writer::writeChildren*。

Private 成员函数

- [ElseIfNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数**Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)**

- [ProgramNode](#) ()

11.29.1 详细描述

在文件 [else_if_node.h](#) 第 11 行定义.

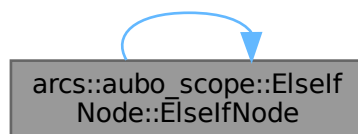
11.29.2 构造及析构函数说明**ElseIfNode() [1/3]**

```
arcs::aubo_scope::ElseIfNode::ElseIfNode (
    ElseIfNode & f)
```

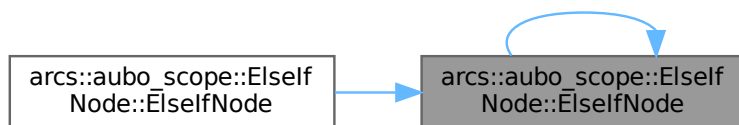
引用了 [ElseIfNode\(\)](#).

被这些函数引用 [ElseIfNode\(\)](#) , 以及 [ElseIfNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

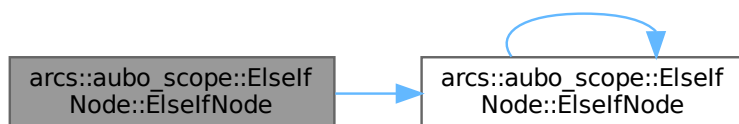


ElseIfNode() [2/3]

```
arcs::aubo_scope::ElseIfNode::ElseIfNode (
    ElseIfNode && f)
```

引用了 [ElseIfNode\(\)](#).

函数调用图:



~ElseIfNode()

```
virtual arcs::aubo_scope::ElseIfNode::~~ElseIfNode () [virtual]
```

ElseIfNode() [3/3]

```
arcs::aubo_scope::ElseIfNode::ElseIfNode () [private]
```

11.29.3 成员函数说明

getExpression()

```
ExpressionPtr arcs::aubo_scope::ElseIfNode::getExpression ()
```

返回

the [Expression](#) to be evaluated in this ElseIf node.

setExpression()

```
void arcs::aubo_scope::ElseIfNode::setExpression (
    ExpressionPtr expression)
```


参数

<i>expression</i>	The expression that will be evaluated in this ElseIfNode .

返回

The same [ElseIfNode](#).

11.29.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [else_if_node.h](#) 第 31 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.29.5 类成员变量说明**d_**

```
void* arcs::aubo_scope::ElseIfNode::d_ { nullptr } [private]
```

在文件 [else_if_node.h](#) 第 33 行定义.

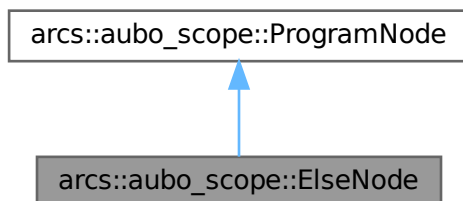
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/else_if_node.h](#)

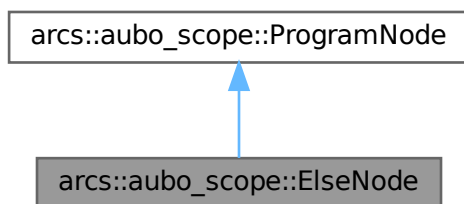
11.30 arcs::aubo_scope::ElseNode 类参考

```
#include <else_node.h>
```

类 arcs::aubo_scope::ElseNode 继承关系图:



arcs::aubo_scope::ElseNode 的协作图:



Public 成员函数

- [ElseNode](#) (ElseNode &f)
- [ElseNode](#) (ElseNode &&f)
- virtual [~ElseNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [ElseNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.30.1 详细描述

在文件 [else_node.h](#) 第 10 行定义.

11.30.2 构造及析构函数说明

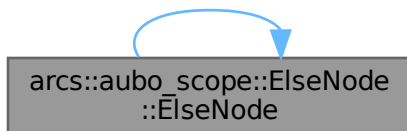
ElseNode() [1/3]

```
arcs::aubo_scope::ElseNode::ElseNode (  
    ElseNode & f)
```

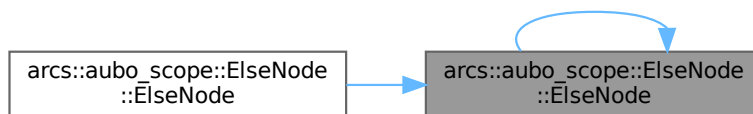
引用了 [ElseNode\(\)](#).

被这些函数引用 [ElseNode\(\)](#) , 以及 [ElseNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

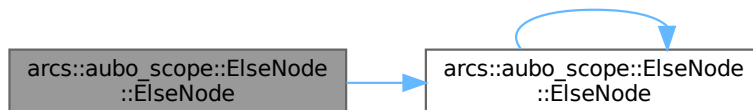


ElseNode() [2/3]

```
arcs::aubo_scope::ElseNode::ElseNode (  
    ElseNode && f)
```

引用了 [ElseNode\(\)](#).

函数调用图:



~ElseNode()

```
virtual arcs::aubo_scope::ElseNode::~~ElseNode () [virtual]
```

ElseNode() [3/3]

```
arcs::aubo_scope::ElseNode::ElseNode () [private]
```

11.30.3 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [else_node.h](#) 第 18 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.30.4 类成员变量说明**d_**

```
void* arcs::aubo_scope::ElseNode::d_ { nullptr } [private]
```

在文件 [else_node.h](#) 第 20 行定义.
该类的文档由以下文件生成:

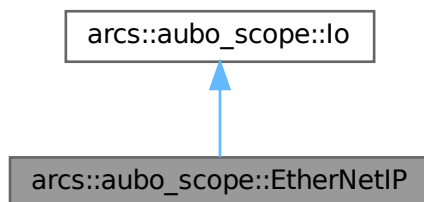
- [include/aubo_caps/domain/program/nodes/builtin/else_node.h](#)

11.31 arcs::aubo_scope::EtherNetIP 类参考

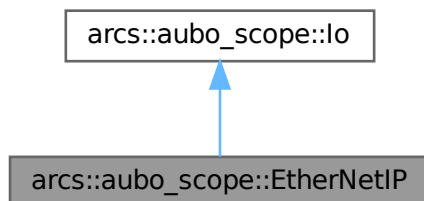
EtherNet/IP IO 接口此接口提供对 EtherNet/IP I/O 的支持。

```
#include <ethernet_ip.h>
```

类 arcs::aubo_scope::EtherNetIP 继承关系图:



arcs::aubo_scope::EtherNetIP 的协作图:



Public 成员函数

- [EtherNetIP](#) (EtherNetIP &f)
- [EtherNetIP](#) (EtherNetIP &&f)
- [~EtherNetIP](#) ()
- int [getByteOffset](#) ()
获取此EtherNet/IP I/O 的字节偏移量
- int [getBitOffset](#) ()
获取此EtherNet/IP I/O 的位偏移量
- bool [isSlave](#) ()
判断此EtherNet/IP I/O 是否为从站
- bool [setValue](#) (int value)
设置EtherNet/IP 信号的整数值。
- int [getValue](#) ()
获取EtherNet/IP 信号的整数值。

Public 成员函数继承自 [arcs::aubo_scope::Io](#)

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- virtual [~Io](#) ()
- std::string [getDisplayname](#) () const
获取当前安装中I/O 的用户自定义名称
- std::string [getDefaultName](#) () const
获取I/O 的机器人默认名称
- std::string [getValueStr](#) () const
获取I/O 的当前读数值（字符串形式）
- [IoType](#) [getType](#) () const
获取I/O 类型
- [InterfaceType](#) [getInterfaceType](#) () const
获取此I/O 所属的接口类型
- int [getAction](#) () const
获取I/O 的动作编号
- bool [isInput](#) () const
判断I/O 是否支持读取值
- bool [isResolvable](#) () const
某些I/O（如MODBUS I/O 和模拟工具输入）在某些情况下可能不可用。此方法用于判断 I/O 是否可用。
- size_t [hashCode](#) ()

Private 成员函数

- [EtherNetIP](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Public 类型继承自 `arcs::aubo_scope::Io`

- enum `IoType` : int {
 DIGITAL , ANALOG , INTEGER , BOOLEAN ,
 DOUBLE }
- enum `InterfaceType` : int {
 STANDARD , TOOL , CONFIGURABLE , LINKIO ,
 MODBUS , GENERAL_PURPOSE , PROFINET , ETHERNET_IP }

Protected 成员函数继承自 `arcs::aubo_scope::Io`

- `Io()`

11.31.1 详细描述

EtherNet/IP IO 接口此接口提供对 EtherNet/IP I/O 的支持。
在文件 `ethernet_ip.h` 第 20 行定义。

11.31.2 构造及析构函数说明

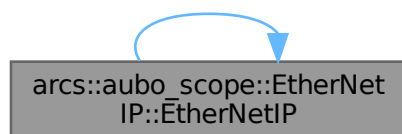
`EtherNetIP()` [1/3]

```
arcs::aubo_scope::EtherNetIP::EtherNetIP (  
    EtherNetIP & f)
```

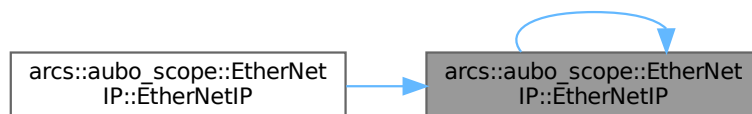
引用了 `EtherNetIP()`。

被这些函数引用 `EtherNetIP()` , 以及 `EtherNetIP()`。

函数调用图:



这是这个函数的调用关系图:

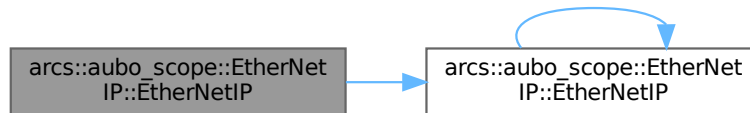


EtherNetIP() [2/3]

```
arcs::aubo_scope::EtherNetIP::EtherNetIP (
    EtherNetIP && f)
```

引用了 [EtherNetIP\(\)](#).

函数调用图:

**~EtherNetIP()**

```
arcs::aubo_scope::EtherNetIP::~~EtherNetIP ()
```

EtherNetIP() [3/3]

```
arcs::aubo_scope::EtherNetIP::EtherNetIP () [private]
```

11.31.3 成员函数说明**getBitOffset()**

```
int arcs::aubo_scope::EtherNetIP::getBitOffset ()
```

获取此EtherNet/IP I/O 的位偏移量

返回

位偏移量。

getByteOffset()

```
int arcs::aubo_scope::EtherNetIP::getByteOffset ()
```

获取此EtherNet/IP I/O 的字节偏移量

返回

字节偏移量。

getValue()

```
int arcs::aubo_scope::EtherNetIP::getValue ()
```

获取EtherNet/IP 信号的整数值。

返回

EtherNet/IP 信号的整数值。

isSlave()

```
bool arcs::aubo_scope::EtherNetIP::isSlave ()
```

判断此EtherNet/IP I/O 是否为从站

返回

是否为从站。

setValue()

```
bool arcs::aubo_scope::EtherNetIP::setValue (
    int value)
```

设置EtherNet/IP 信号的整数值。

参数

<i>value</i>	要设置的整数值。

返回

信号下发成功返回 `true`, 否则返回 `false`。

11.31.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [ethernet_ip.h](#) 第 91 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.31.5 类成员变量说明**d_**

```
void* arcs::aubo_scope::EtherNetIP::d_ { nullptr } [private]
```

在文件 [ethernet_ip.h](#) 第 93 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/io/ethernet_ip.h](#)

11.32 arcs::aubo_scope::EtherNetIPFilter 结构体参考

```
#include <ethernet_ip.h>
```

Public 成员函数

- [EtherNetIPFilter](#) ([Io::IoType](#) type, bool is_input)
- bool [operator\(\)](#) ([IoPtr](#) io)

Private 属性

- [Io::IoType](#) type_
- bool is_input_

11.32.1 详细描述

在文件 [ethernet_ip.h](#) 第 96 行定义.

11.32.2 构造及析构函数说明

EtherNetIPFilter()

```
arcs::aubo_scope::EtherNetIPFilter::EtherNetIPFilter (
    Io::IoType type,
    bool is_input) [inline]
```

在文件 [ethernet_ip.h](#) 第 98 行定义.

引用了 [is_input_](#) , 以及 [type_](#) .

11.32.3 成员函数说明

operator()()

```
bool arcs::aubo_scope::EtherNetIPFilter::operator() (
    IoPtr io) [inline]
```

在文件 [ethernet_ip.h](#) 第 102 行定义.

引用了 [is_input_](#) , 以及 [type_](#) .

11.32.4 类成员变量说明

is_input_

```
bool arcs::aubo_scope::EtherNetIPFilter::is_input_ [private]
```

在文件 [ethernet_ip.h](#) 第 110 行定义.

被这些函数引用 [EtherNetIPFilter\(\)](#) , 以及 [operator\(\)\(\)](#) .

type_

```
Io::IoType arcs::aubo_scope::EtherNetIPFilter::type_ [private]
```

在文件 [ethernet_ip.h](#) 第 109 行定义.

被这些函数引用 [EtherNetIPFilter\(\)](#) , 以及 [operator\(\)\(\)](#) .

该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/io/ethernet_ip.h](#)

11.33 arcs::aubo_scope::Expression 类参考

表达式表达式可用于配置各种内置程序节点, 例如 If 和 Wait 节点. 表达式通常是条件表达式 (计算结果为 true 或 false).

```
#include <expression.h>
```

Public 成员函数

- [Expression](#) (Expression &f)
- [Expression](#) (Expression &&f)
- virtual [~Expression](#) ()
- std::string [toString](#) ()
- void [append](#) (const std::string &expressionPart)
向表达式追加字符串部分。
- void [appendVariable](#) (VariablePtr variable)
向表达式追加变量对象。
- void [appendFeature](#) (FeaturePtr feature)
向表达式追加特征对象。
- void [appendWaypoint](#) (WaypointPtr waypoint)

向表达式追加路点对象。

- void `appendIo` (IoPtr io)
向表达式追加 I/O 对象。

Private 成员函数

- `Expression` ()

Private 属性

- void * `d_` { nullptr }

友元

- class `DataSwitch`

11.33.1 详细描述

表达式表达式可用于配置各种内置程序节点，例如 If 和 Wait 节点。表达式通常是条件表达式（计算结果为 true 或 false）。

表达式通过 `ExpressionBuilder` 接口构建。

在文件 `expression.h` 第 33 行定义。

11.33.2 构造及析构函数说明

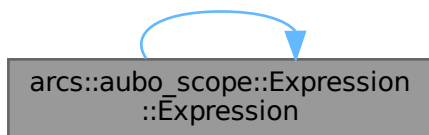
`Expression()` [1/3]

```
arcs::aubo_scope::Expression::Expression (
    Expression & f)
```

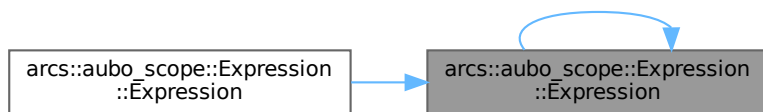
引用了 `Expression()`。

被这些函数引用 `Expression()`，以及 `Expression()`。

函数调用图：



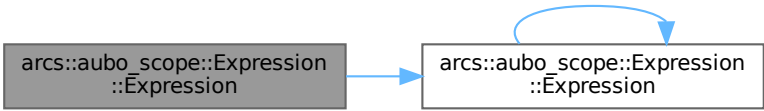
这是这个函数的调用关系图：



Expression() [2/3]

```
arcs::aubo_scope::Expression::Expression (  
    Expression && f)
```

引用了 [Expression\(\)](#).
函数调用图:



~Expression()

```
virtual arcs::aubo_scope::Expression::~~Expression () [virtual]
```

Expression() [3/3]

```
arcs::aubo_scope::Expression::Expression () [private]
```

11.33.3 成员函数说明

append()

```
void arcs::aubo_scope::Expression::append (  
    const std::string & expressionPart)
```

向表达式追加字符串部分。

参数

<i>expressionPart</i>	要追加的表达式部分。

appendFeature()

```
void arcs::aubo_scope::Expression::appendFeature (  
    FeaturePtr feature)
```

向表达式追加特征对象。

appendIo()

```
void arcs::aubo_scope::Expression::appendIo (  
    IoPtr io)
```

向表达式追加I/O 对象。

appendVariable()

```
void arcs::aubo_scope::Expression::appendVariable (  
    VariablePtr variable)
```

向表达式追加变量对象。

appendWaypoint()

```
void arcs::aubo_scope::Expression::appendWaypoint (
```

```
    WaypointPtr waypoint)
```

向表达式追加路点对象。

toString()

```
std::string arcs::aubo_scope::Expression::toString ()
```

11.33.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [expression.h](#) 第 79 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.33.5 类成员变量说明**d_**

```
void* arcs::aubo_scope::Expression::d_ { nullptr } [private]
```

在文件 [expression.h](#) 第 81 行定义.

该类的文档由以下文件生成:

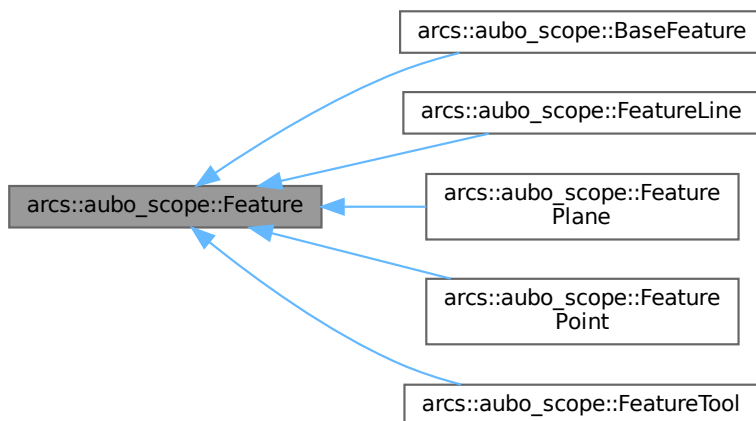
- [include/aubo_caps/domain/value/expression/expression.h](#)

11.34 arcs::aubo_scope::Feature 类参考

特征接口此基础接口表示 AuboScope 中所有可用的特征类型。

```
#include <feature.h>
```

类 arcs::aubo_scope::Feature 继承关系图:

**Public 成员函数**

- [Feature](#) (Feature &f)
- [Feature](#) (Feature &&f)
- virtual [~Feature](#) ()

- `std::string getDisplayName ()`
获取特征的显示名称。注意：特征可随时被重命名。
- `bool isDefined ()`
返回特征的位置是否已完全定义/指定。
- `bool isVariable ()`
返回特征是否可作为变量在程序中使用。
- `bool isJoggable ()`
- `bool isResolvable ()`
特征无法保证始终存在于 *AuboScope* 中。终端用户可以加载不包含该特征的不同安装，或者终端用户或添加该特征的 *AuboCap* 可以将其移除。此方法用于判断特征是否存在。
- `std::vector< double > getPose ()`
返回特征的笛卡尔位置。
- `size_t hashCode ()`

Protected 成员函数

- `Feature ()`

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

11.34.1 详细描述

特征接口此基础接口表示 *AuboScope* 中所有可用的特征类型。
在文件 `feature.h` 第 22 行定义。

11.34.2 构造及析构函数说明

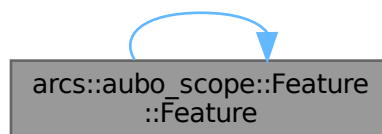
Feature() [1/3]

```
arcs::aubo_scope::Feature::Feature (
    Feature & f)
```

引用了 `Feature()`。

被这些函数引用 `Feature()`，以及 `Feature()`。

函数调用图：



这是这个函数的调用关系图:

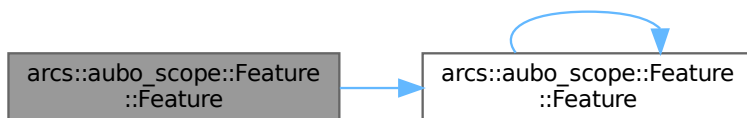


Feature() [2/3]

```
arcs::aubo_scope::Feature::Feature (
    Feature && f)
```

引用了 [Feature\(\)](#).

函数调用图:



~Feature()

```
virtual arcs::aubo_scope::Feature::~~Feature () [virtual]
```

Feature() [3/3]

```
arcs::aubo_scope::Feature::Feature () [protected]
```

11.34.3 成员函数说明

getDisplayName()

```
std::string arcs::aubo_scope::Feature::getDisplayName ()
```

获取特征的显示名称。注意：特征可随时被重命名。

返回

特征名称。

getPose()

```
std::vector< double > arcs::aubo_scope::Feature::getPose ()
```

返回特征的笛卡尔位置。

返回

表示特征笛卡尔位置的位姿。

hashCode()

```
size_t arcs::aubo_scope::Feature::hashCode ()
```

isDefined()

```
bool arcs::aubo_scope::Feature::isDefined ()
```

返回特征的位置是否已完全定义/指定。

返回

如果特征已定义返回 `true`, 否则返回 `false`。

isJoggable()

```
bool arcs::aubo_scope::Feature::isJoggable ()
```

isResolvable()

```
bool arcs::aubo_scope::Feature::isResolvable ()
```

特征无法保证始终存在于AuboScope中。终端用户可以加载不包含该特征的不同安装, 或者终端用户或添加该特征的AuboCap 可以将其移除。此方法用于判断特征是否存在。

返回

如果此特征存在于AuboScope中返回 `true`, 否则返回 `false`。

isVariable()

```
bool arcs::aubo_scope::Feature::isVariable ()
```

返回特征是否可作为变量在程序中使用。

返回

如果特征可作为变量使用返回 `true`, 否则返回 `false`。

11.34.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [feature.h](#) 第 108 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.34.5 类成员变量说明**d_**

```
void* arcs::aubo_scope::Feature::d_ { nullptr } [private]
```

在文件 [feature.h](#) 第 109 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/feature/feature.h](#)

11.35 arcs::aubo_scope::FeatureContributionModel 类参考

特征贡献模型此接口提供在 AuboScope 中添加、更新和删除特征的功能。

```
#include <feature_contribution_model.h>
```

Public 成员函数

- [FeatureContributionModel](#) ([FeatureContributionModel](#) &f)
- [FeatureContributionModel](#) ([FeatureContributionModel](#) &&f)
- [~FeatureContributionModel](#) ()
- [FeaturePtr](#) [addFeature](#) (const std::string &idKey, const std::string &suggestedName, const std::vector< double > &pose)
向当前 *AuboScope* 安装中添加一个特征。添加后终端用户可选择该特征，但不可修改。
- [FeaturePtr](#) [getFeature](#) (const std::string &idKey)
返回此前由此 *AuboCap* 使用相同 *idKey* 标识符添加的特征。用于验证特征是否存在。
- void [updateFeature](#) (const std::string &idKey, const std::vector< double > &newPose)
更新此 *AuboCap* 已添加特征的位姿。
- void [renameFeature](#) (const std::string &idKey, const std::string &newName)
- void [removeFeature](#) (const std::string &idKey)
从此 *AuboCap* 中移除已添加的特征。使用该特征的程序节点将变为未定义。

Private 成员函数

- [FeatureContributionModel](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.35.1 详细描述

特征贡献模型此接口提供在 *AuboScope* 中添加、更新和删除特征的功能。

注意：此接口只能在安装贡献中使用。

在文件 [feature_contribution_model.h](#) 第 28 行定义。

11.35.2 构造及析构函数说明

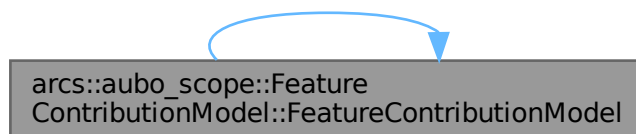
[FeatureContributionModel](#)() [1/3]

```
arcs::aubo_scope::FeatureContributionModel::FeatureContributionModel (
    FeatureContributionModel & f)
```

引用了 [FeatureContributionModel](#)()。

被这些函数引用 [FeatureContributionModel](#)()，以及 [FeatureContributionModel](#)()。

函数调用图:



这是这个函数的调用关系图:



FeatureContributionModel() [2/3]

```
arcs::aubo_scope::FeatureContributionModel::FeatureContributionModel (
    FeatureContributionModel && f)
```

引用了 [FeatureContributionModel\(\)](#).

函数调用图:



~FeatureContributionModel()

```
arcs::aubo_scope::FeatureContributionModel::~~FeatureContributionModel ()
```

FeatureContributionModel() [3/3]

```
arcs::aubo_scope::FeatureContributionModel::FeatureContributionModel () [private]
```

11.35.3 成员函数说明

addFeature()

```
FeaturePtr arcs::aubo_scope::FeatureContributionModel::addFeature (
    const std::string & idKey,
    const std::string & suggestedName,
    const std::vector< double > & pose)
```

向当前AuboScope 安装中添加一个特征。添加后终端用户可选择该特征，但不可修改。

参数

<i>idKey</i>	用于标识此特征的键。该键仅对此AuboCap 有效。
<i>suggestedName</i>	建议的特征名称。有效名称必须匹配正则表达式 <code>[a-zA-Z][a-zA-Z0-9_]{0,14}</code> 。
<i>pose</i>	特征相对于机器人基座的位姿。

返回

在AuboScope 中创建并注册的特征。

异常

<i>FeatureAlreadyAddedException</i>	如果之前已用相同 idKey 添加过特征。
<i>IllegalFeatureNameException</i>	如果建议的名称不匹配所需正则表达式。

getFeature()

```
FeaturePtr arcs::aubo_scope::FeatureContributionModel::getFeature (  
    const std::string & idKey)
```

返回此前由此AuboCap 使用相同 idKey 标识符添加的特征。用于验证特征是否存在。

参数

<i>idKey</i>	用于标识特征的键。

返回

此前添加的特征，如果不存在则返回 `null`。

removeFeature()

```
void arcs::aubo_scope::FeatureContributionModel::removeFeature (  
    const std::string & idKey)
```

从此AuboCap 中移除已添加的特征。使用该特征的程序节点将变为未定义。

参数

<i>idKey</i>	添加特征时使用的键。

异常

<i>FeatureNotFoundException</i>	如果不存在对应 idKey 的特征。

renameFeature()

```
void arcs::aubo_scope::FeatureContributionModel::renameFeature (  
    const std::string & idKey,  
    const std::string & newName)
```

updateFeature()

```
void arcs::aubo_scope::FeatureContributionModel::updateFeature (  
    const std::string & idKey,  
    const std::vector< double > & newPose)
```

更新此AuboCap 已添加特征的位姿。

参数

<i>idKey</i>	用于标识特征的键。
<i>newPose</i>	特征的新位姿。

异常

<i>FeatureNotFoundException</i>	如果不存在对应 idKey 的特征。

11.35.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [feature_contribution_model.h](#) 第 123 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.35.5 类成员变量说明

d_

```
void* arcs::aubo_scope::FeatureContributionModel::d_ { nullptr } [private]
```

在文件 [feature_contribution_model.h](#) 第 125 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/feature/feature_contribution_model.h](#)

11.36 arcs::aubo_scope::FeatureDefinedFilter 结构体参考

```
#include <feature.h>
```

Public 成员函数

- [bool operator\(\)](#) (FeaturePtr feature)

11.36.1 详细描述

在文件 [feature.h](#) 第 112 行定义.

11.36.2 成员函数说明

operator()

```
bool arcs::aubo_scope::FeatureDefinedFilter::operator() (  
    FeaturePtr feature) [inline]
```

在文件 [feature.h](#) 第 114 行定义.

该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/feature/feature.h](#)

11.37 arcs::aubo_scope::FeatureJoggableFilter 结构体参考

```
#include <feature.h>
```

Public 成员函数

- bool [operator\(\)](#) (FeaturePtr feature)

11.37.1 详细描述

在文件 [feature.h](#) 第 120 行定义.

11.37.2 成员函数说明

operator()

```
bool arcs::aubo_scope::FeatureJoggableFilter::operator() (  
    FeaturePtr feature) [inline]
```

在文件 [feature.h](#) 第 122 行定义.

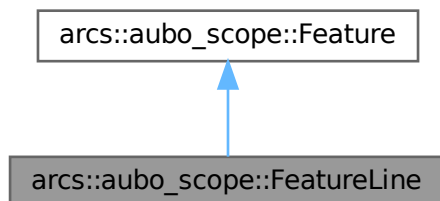
该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/feature/feature.h](#)

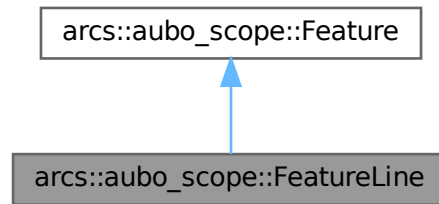
11.38 arcs::aubo_scope::FeatureLine 类参考

```
#include <feature_line.h>
```

类 arcs::aubo_scope::FeatureLine 继承关系图:



arcs::aubo_scope::FeatureLine 的协作图:



Public 成员函数

- [FeatureLine](#) (FeatureLine &f)
- [FeatureLine](#) (FeatureLine &&f)
- virtual [~FeatureLine](#) ()

Public 成员函数继承自 [arcs::aubo_scope::Feature](#)

- [Feature](#) (Feature &f)
- [Feature](#) (Feature &&f)
- virtual [~Feature](#) ()
- std::string [getDisplayname](#) ()
获取特征的显示名称。注意：特征可随时被重命名。
- bool [isDefined](#) ()
返回特征的位置是否已完全定义/指定。
- bool [isVariable](#) ()
返回特征是否可作为变量在程序中使用。
- bool [isJoggable](#) ()
- bool [isResolvable](#) ()
特征无法保证始终存在于 *AuboScope* 中。终端用户可以加载不包含该特征的不同安装，或者终端用户或添加该特征的 *AuboCap* 可以将其移除。此方法用于判断特征是否存在。
- std::vector< double > [getPose](#) ()
返回特征的笛卡尔位置。
- size_t [hashCode](#) ()

Private 成员函数

- [FeatureLine](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::Feature](#)

- [Feature](#) ()

11.38.1 详细描述

在文件 `feature_line.h` 第 12 行定义.

11.38.2 构造及析构函数说明

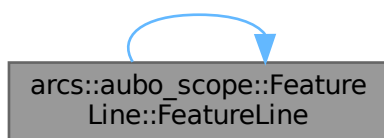
FeatureLine() [1/3]

```
arcs::aubo_scope::FeatureLine::FeatureLine (  
    FeatureLine & f)
```

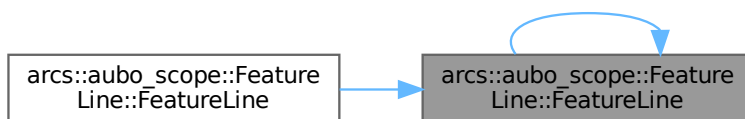
引用了 `FeatureLine()`.

被这些函数引用 `FeatureLine()` , 以及 `FeatureLine()`.

函数调用图:



这是这个函数的调用关系图:

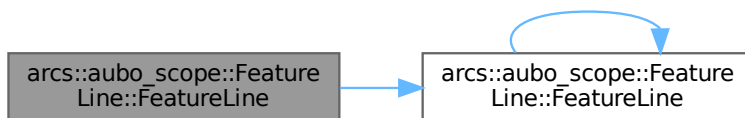


FeatureLine() [2/3]

```
arcs::aubo_scope::FeatureLine::FeatureLine (  
    FeatureLine && f)
```

引用了 `FeatureLine()`.

函数调用图:



~FeatureLine()

```
virtual arcs::aubo_scope::FeatureLine::~~FeatureLine () [virtual]
```

FeatureLine() [3/3]

```
arcs::aubo_scope::FeatureLine::FeatureLine () [private]
```

11.38.3 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [feature_line.h](#) 第 20 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.38.4 类成员变量说明**d_**

```
void* arcs::aubo_scope::FeatureLine::d_ { nullptr } [private]
```

在文件 [feature_line.h](#) 第 22 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/feature/feature_line.h](#)

11.39 arcs::aubo_scope::FeatureModel 类参考

特征模型提供从当前机器人安装中获取特征的方法。

```
#include <feature_model.h>
```

Public 成员函数

- [FeatureModel](#) (FeatureModel &f)
- [FeatureModel](#) (FeatureModel &&f)
- [~FeatureModel](#) ()
- [std::vector< FeaturePtr > getAll](#) ()
- [template<typename T> std::vector< FeaturePtr > getGeomFeture](#) ()
获取当前安装中的几何特征
- [FeaturePtr getByName](#) (const std::string &name)
根据名称获取特征
- [FeaturePtr getBaseFeature](#) ()
获取当前安装的基础特征
- [FeaturePtr getToolFeature](#) ()
获取当前安装的工具特征
- [std::vector< FeaturePtr > getFeatures](#) (std::function< bool(FeaturePtr)> filter)
根据过滤器获取特征列表

Private 成员函数

- [FeatureModel](#) ()

Private 属性

- [void * d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.39.1 详细描述

特征模型提供从当前机器人安装中获取特征的方法。
在文件 [feature_model.h](#) 第 24 行定义.

11.39.2 构造及析构函数说明

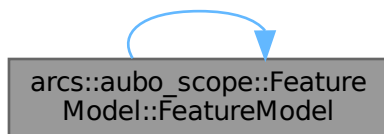
FeatureModel() [1/3]

```
arcs::aubo_scope::FeatureModel::FeatureModel (  
    FeatureModel & f)
```

引用了 [FeatureModel\(\)](#).

被这些函数引用 [FeatureModel\(\)](#) , 以及 [FeatureModel\(\)](#).

函数调用图:



这是这个函数的调用关系图:

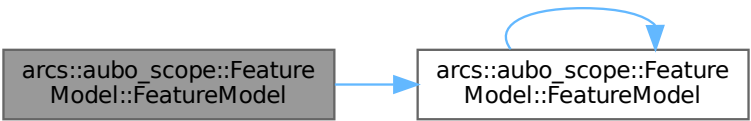


FeatureModel() [2/3]

```
arcs::aubo_scope::FeatureModel::FeatureModel (  
    FeatureModel && f)
```

引用了 [FeatureModel\(\)](#).

函数调用图:



~FeatureModel()

```
arcs::aubo_scope::FeatureModel::~~FeatureModel ()
```

FeatureModel() [3/3]

```
arcs::aubo_scope::FeatureModel::FeatureModel () [private]
```

11.39.3 成员函数说明

getAll()

```
std::vector< FeaturePtr > arcs::aubo_scope::FeatureModel::getAll ()
```

getBaseFeature()

```
FeaturePtr arcs::aubo_scope::FeatureModel::getBaseFeature ()  
获取当前安装的基础特征
```

返回

基础特征。

getByName()

```
FeaturePtr arcs::aubo_scope::FeatureModel::getByName (  
    const std::string & name)  
根据名称获取特征
```

参数

<i>name</i>	特征名称。

返回

对应名称的特征。

getFeatures()

```
std::vector< FeaturePtr > arcs::aubo_scope::FeatureModel::getFeatures (  
    std::function< bool(FeaturePtr)> filter)  
根据过滤器获取特征列表
```

参数

<i>filter</i>	过滤条件。

返回

满足过滤条件的特征集合。

getGeomFeture()

```
template<typename T>
std::vector< FeaturePtr > arcs::aubo_scope::FeatureModel::getGeomFeture ()
```

获取当前安装中的几何特征

返回

几何特征集合。

getToolFeature()

```
FeaturePtr arcs::aubo_scope::FeatureModel::getToolFeature ()
```

获取当前安装的工具特征

返回

工具特征。

11.39.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [feature_model.h](#) 第 104 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.39.5 类成员变量说明**d_**

```
void* arcs::aubo_scope::FeatureModel::d_ { nullptr } [private]
```

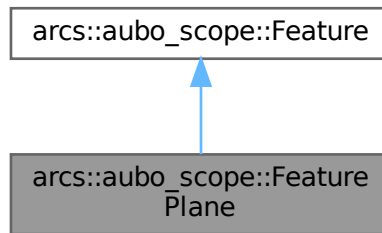
在文件 [feature_model.h](#) 第 106 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/feature/feature_model.h](#)

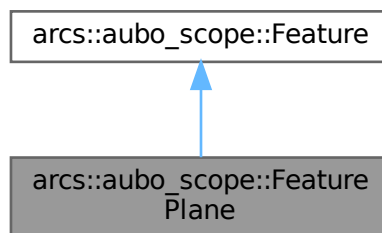
11.40 arcs::aubo_scope::FeaturePlane 类参考

```
#include <feature_plane.h>
```

类 arcs::aubo_scope::FeaturePlane 继承关系图:



arcs::aubo_scope::FeaturePlane 的协作图:



Public 成员函数

- [FeaturePlane](#) (FeaturePlane &f)
- [FeaturePlane](#) (FeaturePlane &&f)
- virtual [~FeaturePlane](#) ()

Public 成员函数继承自 [arcs::aubo_scope::Feature](#)

- [Feature](#) (Feature &f)
- [Feature](#) (Feature &&f)
- virtual [~Feature](#) ()
- std::string [getDisplayname](#) ()
获取特征的显示名称。注意：特征可随时被重命名。
- bool [isDefined](#) ()
返回特征的位置是否已完全定义/指定。
- bool [isVariable](#) ()
返回特征是否可作为变量在程序中使用。
- bool [isJoggable](#) ()
- bool [isResolvable](#) ()
特征无法保证始终存在于 *AuboScope* 中。终端用户可以加载不包含该特征的不同安装，或者终端用户或添加该特征的 *AuboCap* 可以将其移除。此方法用于判断特征是否存在。
- std::vector< double > [getPose](#) ()

返回特征的笛卡尔位置。

- `size_t hashCode ()`

Private 成员函数

- `FeaturePlane ()`

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数

Protected 成员函数继承自 `arcs::aubo_scope::Feature`

- `Feature ()`

11.40.1 详细描述

在文件 `feature_plane.h` 第 10 行定义.

11.40.2 构造及析构函数说明

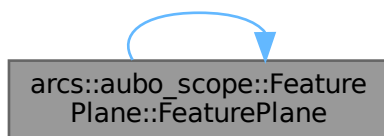
FeaturePlane() [1/3]

```
arcs::aubo_scope::FeaturePlane::FeaturePlane (
    FeaturePlane & f)
```

引用了 `FeaturePlane()`.

被这些函数引用 `FeaturePlane()` , 以及 `FeaturePlane()`.

函数调用图:



这是这个函数的调用关系图:

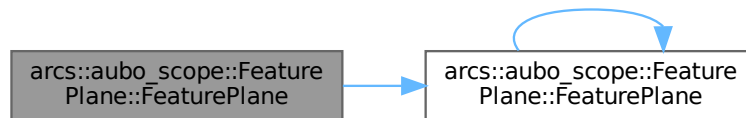


FeaturePlane() [2/3]

```
arcs::aubo_scope::FeaturePlane::FeaturePlane (
    FeaturePlane && f)
```

引用了 [FeaturePlane\(\)](#).

函数调用图:

**~FeaturePlane()**

```
virtual arcs::aubo_scope::FeaturePlane::~~FeaturePlane () [virtual]
```

FeaturePlane() [3/3]

```
arcs::aubo_scope::FeaturePlane::FeaturePlane () [private]
```

11.40.3 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [feature_plane.h](#) 第 18 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.40.4 类成员变量说明**d_**

```
void* arcs::aubo_scope::FeaturePlane::d_ { nullptr } [private]
```

在文件 [feature_plane.h](#) 第 20 行定义.

该类的文档由以下文件生成:

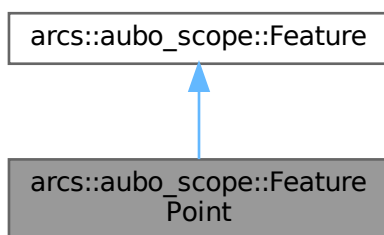
- [include/aubo_caps/domain/feature/feature_plane.h](#)

11.41 arcs::aubo_scope::FeaturePoint 类参考

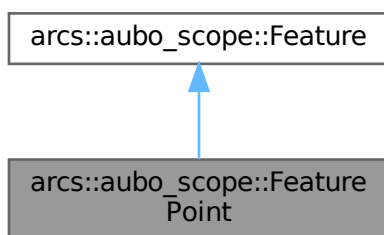
This interface represents a Point feature in AuboScope

```
#include <feature_point.h>
```

类 `arcs::aubo_scope::FeaturePoint` 继承关系图:



`arcs::aubo_scope::FeaturePoint` 的协作图:



Public 成员函数

- `FeaturePoint` (`FeaturePoint &f`)
- `FeaturePoint` (`FeaturePoint &&f`)
- `virtual ~FeaturePoint ()`

Public 成员函数继承自 `arcs::aubo_scope::Feature`

- `Feature` (`Feature &f`)
- `Feature` (`Feature &&f`)
- `virtual ~Feature ()`
- `std::string getDisplayName ()`
获取特征的显示名称。注意：特征可随时被重命名。
- `bool isDefined ()`
返回特征的位置是否已完全定义/指定。
- `bool isVariable ()`
返回特征是否可作为变量在程序中使用。
- `bool isJoggable ()`
- `bool isResolvable ()`
特征无法保证始终存在于 *AuboScope* 中。终端用户可以加载不包含该特征的不同安装，或者终端用户或添加该特征的 *AuboCap* 可以将其移除。此方法用于判断特征是否存在。
- `std::vector< double > getPose ()`

返回特征的笛卡尔位置。

- `size_t hashCode ()`

Private 成员函数

- `FeaturePoint ()`

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数

Protected 成员函数继承自 `arcs::aubo_scope::Feature`

- `Feature ()`

11.41.1 详细描述

This interface represents a Point feature in AuboScope
在文件 `feature_point.h` 第 13 行定义.

11.41.2 构造及析构函数说明

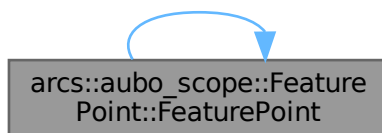
FeaturePoint() [1/3]

```
arcs::aubo_scope::FeaturePoint::FeaturePoint (
    FeaturePoint & f)
```

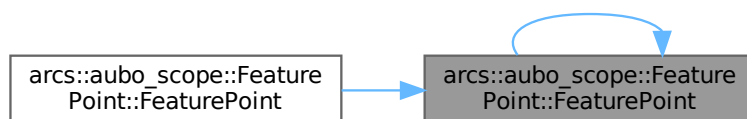
引用了 `FeaturePoint()`.

被这些函数引用 `FeaturePoint()` , 以及 `FeaturePoint()`.

函数调用图:



这是这个函数的调用关系图:

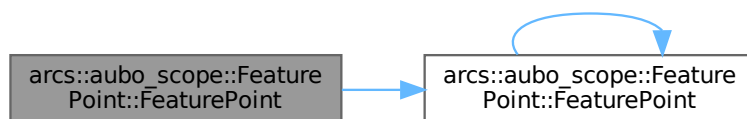


FeaturePoint() [2/3]

```
arcs::aubo_scope::FeaturePoint::FeaturePoint (
    FeaturePoint && f)
```

引用了 [FeaturePoint\(\)](#).

函数调用图:

**~FeaturePoint()**

```
virtual arcs::aubo_scope::FeaturePoint::~~FeaturePoint () [virtual]
```

FeaturePoint() [3/3]

```
arcs::aubo_scope::FeaturePoint::FeaturePoint () [private]
```

11.41.3 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [feature_point.h](#) 第 21 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.41.4 类成员变量说明**d_**

```
void* arcs::aubo_scope::FeaturePoint::d_ { nullptr } [private]
```

在文件 [feature_point.h](#) 第 23 行定义.

该类的文档由以下文件生成:

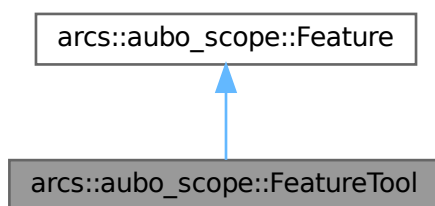
- [include/aubo_caps/domain/feature/feature_point.h](#)

11.42 arcs::aubo_scope::FeatureTool 类参考

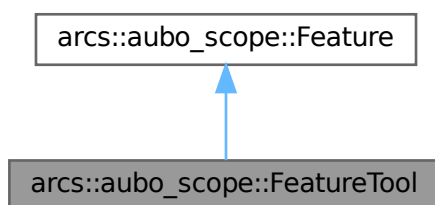
This interface represents the Tool feature in AuboScope.

```
#include <feature_tool.h>
```


类 arcs::aubo_scope::FeatureTool 继承关系图:



arcs::aubo_scope::FeatureTool 的协作图:



Public 成员函数

- [FeatureTool](#) (FeatureTool &f)
- [FeatureTool](#) (FeatureTool &&f)
- virtual [~FeatureTool](#) ()

Public 成员函数继承自 [arcs::aubo_scope::Feature](#)

- [Feature](#) (Feature &f)
- [Feature](#) (Feature &&f)
- virtual [~Feature](#) ()
- std::string [getDisplayname](#) ()
获取特征的显示名称。注意：特征可随时被重命名。
- bool [isDefined](#) ()
返回特征的位置是否已完全定义/指定。
- bool [isVariable](#) ()
返回特征是否可作为变量在程序中使用。
- bool [isJoggable](#) ()
- bool [isResolvable](#) ()
特征无法保证始终存在于 *AuboScope* 中。终端用户可以加载不包含该特征的不同安装，或者终端用户或添加该特征的 *AuboCap* 可以将其移除。此方法用于判断特征是否存在。
- std::vector< double > [getPose](#) ()
返回特征的笛卡尔位置。
- size_t [hashCode](#) ()

Private 成员函数

- [FeatureTool\(\)](#)

Private 属性

- `void * d\_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数**Protected 成员函数继承自 [arcs::aubo_scope::Feature](#)**

- [Feature\(\)](#)

11.42.1 详细描述

This interface represents the Tool feature in AuboScope.
在文件 [feature_tool.h](#) 第 13 行定义.

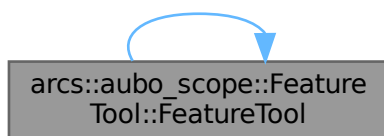
11.42.2 构造及析构函数说明**FeatureTool() [1/3]**

```
arcs::aubo_scope::FeatureTool::FeatureTool (
    FeatureTool & f)
```

引用了 [FeatureTool\(\)](#).

被这些函数引用 [FeatureTool\(\)](#) , 以及 [FeatureTool\(\)](#).

函数调用图:



这是这个函数的调用关系图:



FeatureTool() [2/3]

```
arcs::aubo_scope::FeatureTool::FeatureTool (
    FeatureTool && f)
```

引用了 [FeatureTool\(\)](#).

函数调用图:

**~FeatureTool()**

```
virtual arcs::aubo_scope::FeatureTool::~~FeatureTool () [virtual]
```

FeatureTool() [3/3]

```
arcs::aubo_scope::FeatureTool::FeatureTool () [private]
```

11.42.3 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [feature_tool.h](#) 第 21 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.42.4 类成员变量说明**d_**

```
void* arcs::aubo_scope::FeatureTool::d_ { nullptr } [private]
```

在文件 [feature_tool.h](#) 第 23 行定义.

该类的文档由以下文件生成:

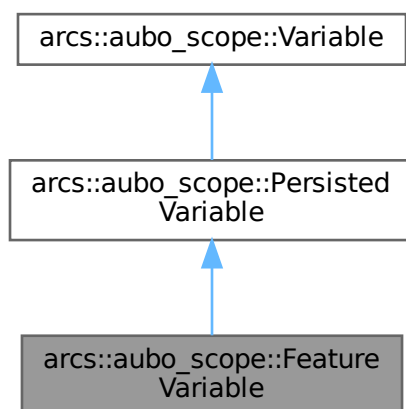
- [include/aubo_caps/domain/feature/feature_tool.h](#)

11.43 arcs::aubo_scope::FeatureVariable 类参考

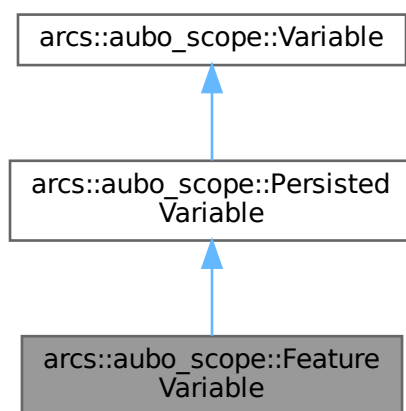
特征变量在安装中被标记为变量的特征，即使在机器人关闭和重新启动后也会存在。

```
#include <feature_variable.h>
```

类 `arcs::aubo_scope::FeatureVariable` 继承关系图:



`arcs::aubo_scope::FeatureVariable` 的协作图:



Public 成员函数

- `FeatureVariable` (`FeatureVariable &f`)
- `FeatureVariable` (`FeatureVariable &&f`)
- `virtual ~FeatureVariable ()`

Public 成员函数继承自 `arcs::aubo_scope::PersistedVariable`

- `PersistedVariable` (`PersistedVariable &f`)
- `PersistedVariable` (`PersistedVariable &&f`)
- `virtual ~PersistedVariable ()`
- `bool isResolvable ()`

持久化变量无法保证始终存在。当终端用户加载不包含该变量的不同安装或删除该变量时，持久化变量将不可用。如果变量无法解析，存储了 *PersistedVariable* 的程序节点将变为未定义。

Public 成员函数继承自 [arcs::aubo_scope::Variable](#)

- [Variable](#) (Variable &f)
- [Variable](#) (Variable &&f)
- virtual [~Variable](#) ()
- [Type](#) [getType](#) ()
获取变量类型
- std::string [getDisplay](#)[Name](#) ()
获取变量显示名称。注意：变量可随时被重命名。
- bool [equals](#) (VariablePtr object)
判断是否引用相同的内部变量
- size_t [hashCode](#) ()
获取哈希码
- std::string [toString](#) ()
获取变量的字符串表示

Private 成员函数

- [FeatureVariable](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Public 类型继承自 [arcs::aubo_scope::Variable](#)

- enum [Type](#) : int { [GLOBAL](#) , [VALUE_PERSISTED](#) , [FEATURE](#) }
The variable type used to determine which type of variable this instance is.

Protected 成员函数继承自 [arcs::aubo_scope::PersistedVariable](#)

- [PersistedVariable](#) ()

Protected 成员函数继承自 [arcs::aubo_scope::Variable](#)

- [Variable](#) ()

11.43.1 详细描述

特征变量在安装中被标记为变量的特征，即使在机器人关闭和重新启动后也会存在。
在文件 [feature_variable.h](#) 第 21 行定义。

11.43.2 构造及析构函数说明

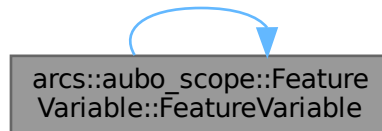
FeatureVariable() [1/3]

```
arcs::aubo_scope::FeatureVariable::FeatureVariable (
    FeatureVariable & f)
```

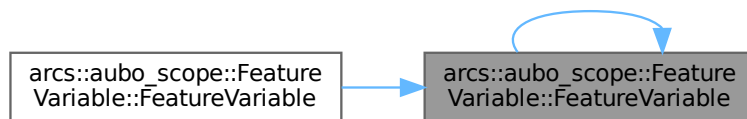
引用了 [FeatureVariable\(\)](#).

被这些函数引用 [FeatureVariable\(\)](#) , 以及 [FeatureVariable\(\)](#).

函数调用图:



这是这个函数的调用关系图:

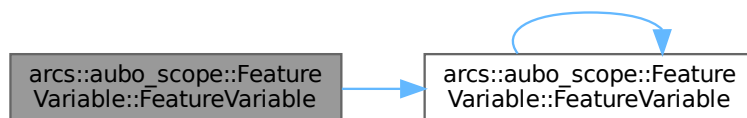


FeatureVariable() [2/3]

```
arcs::aubo_scope::FeatureVariable::FeatureVariable (
    FeatureVariable && f)
```

引用了 [FeatureVariable\(\)](#).

函数调用图:



~FeatureVariable()

```
virtual arcs::aubo_scope::FeatureVariable::~~FeatureVariable () [virtual]
```

FeatureVariable() [3/3]

```
arcs::aubo_scope::FeatureVariable::FeatureVariable () [private]
```

11.43.3 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [feature_variable.h](#) 第 29 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.43.4 类成员变量说明**d_**

```
void* arcs::aubo_scope::FeatureVariable::d_ { nullptr } [private]
```

在文件 [feature_variable.h](#) 第 31 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/variable/feature_variable.h](#)

11.44 arcs::aubo_scope::FeatureVariableFilter 结构体参考

```
#include <feature_variable.h>
```

Public 成员函数

- [bool operator\(\)](#) (VariablePtr var)

11.44.1 详细描述

在文件 [feature_variable.h](#) 第 34 行定义.

11.44.2 成员函数说明**operator()()**

```
bool arcs::aubo_scope::FeatureVariableFilter::operator() (
```

```
VariablePtr var) [inline]
```

在文件 [feature_variable.h](#) 第 36 行定义.

引用了 [arcs::aubo_scope::Variable::FEATURE](#).

该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/variable/feature_variable.h](#)

11.45 arcs::aubo_scope::FeedScrewParameters 类参考

```
#include <screw_driver_configuration.h>
```

Public 成员函数

- [virtual ~FeedScrewParameters \(\)](#)=default

11.45.1 详细描述

This interface provides parameters relevant for generating script code for the Feed Screw capability (registered using `ScrewdriverCapabilities#registerFeedScrewCapability(ScriptCodeGenerator)`).

These parameters are passed when AuboScope calls the implementation of the `ScriptCodeGenerator#generateScript` method responsible for the generating the script code.

The interface is intentionally empty, but could have parameters in future releases.

在文件 [screw_driver_configuration.h](#) 第 84 行定义.

11.45.2 构造及析构造函数说明

`~FeedScrewParameters()`

```
virtual arcs::aubo_scope::FeedScrewParameters::~~FeedScrewParameters () [virtual], [default]
```

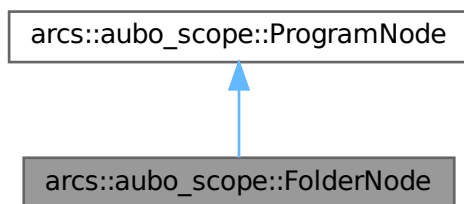
该类的文档由以下文件生成:

- `include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h`

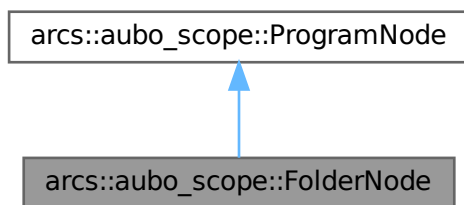
11.46 arcs::aubo_scope::FolderNode 类参考

```
#include <folder_node.h>
```

类 `arcs::aubo_scope::FolderNode` 继承关系图:



`arcs::aubo_scope::FolderNode` 的协作图:



Public 成员函数

- [FolderNode](#) (`FolderNode &f`)
- [FolderNode](#) (`FolderNode &&f`)
- `virtual ~FolderNode ()`
- `void setFolderName (const std::string &folder_name)`

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [FolderNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.46.1 详细描述

在文件 [folder_node.h](#) 第 10 行定义.

11.46.2 构造及析构函数说明

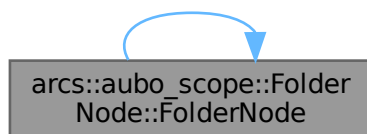
FolderNode() [1/3]

```
arcs::aubo_scope::FolderNode::FolderNode (
    FolderNode & f)
```

引用了 [FolderNode\(\)](#).

被这些函数引用 [FolderNode\(\)](#) , 以及 [FolderNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

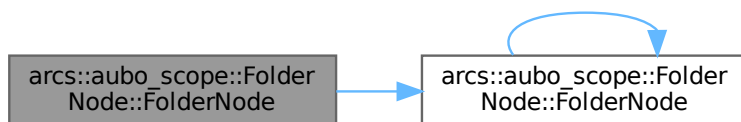


FolderNode() [2/3]

```
arcs::aubo_scope::FolderNode::FolderNode (
    FolderNode && f)
```

引用了 [FolderNode\(\)](#).

函数调用图:



~FolderNode()

```
virtual arcs::aubo_scope::FolderNode::~~FolderNode () [virtual]
```

FolderNode() [3/3]

```
arcs::aubo_scope::FolderNode::FolderNode () [private]
```

11.46.3 成员函数说明

setFolderName()

```
void arcs::aubo_scope::FolderNode::setFolderName (
    const std::string & folder_name)
```

- Set the name for the Folder node.

参数

<i>set</i>	The name for the node.

返回

This node.

11.46.4 友元及相关符号说明

DataSwitch

friend class DataSwitch [friend]
 在文件 [folder_node.h](#) 第 25 行定义.
 引用了 [DataSwitch](#).
 被这些函数引用 [DataSwitch](#).

11.46.5 类成员变量说明

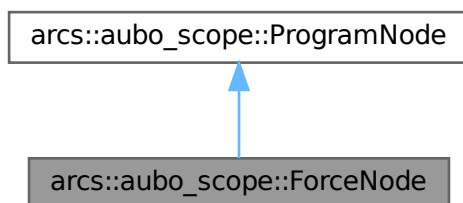
d_

void* arcs::aubo_scope::FolderNode::d_ { nullptr } [private]
 在文件 [folder_node.h](#) 第 27 行定义.
 该类的文档由以下文件生成:

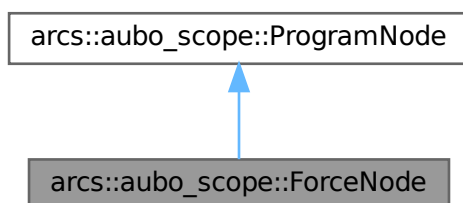
- [include/aubo_caps/domain/program/nodes/builtin/folder_node.h](#)

11.47 arcs::aubo_scope::ForceNode 类参考

#include <force_node.h>
 类 arcs::aubo_scope::ForceNode 继承关系图:



arcs::aubo_scope::ForceNode 的协作图:



Public 类型

- enum [ConfigType](#) { [SIMPLE](#) , [FRAME](#) , [POINT](#) , [MOTION](#) }
- The configuration type used to determine which type of configuration this instance is.*

Public 成员函数

- `ForceNode` (`ForceNode &f`)
- `ForceNode` (`ForceNode &&f`)
- `virtual ~ForceNode` ()

Public 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` (`ProgramNode &f`)
- `ProgramNode` (`ProgramNode &&f`)
- `virtual ~ProgramNode` ()
- `ProgramNodeContributionPtr` `getProgramNodeContribution` ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `ScriptWriter::writeChildren`。

Private 成员函数

- `ForceNode` ()

Private 属性

- `void * d_` { `nullptr` }

友元

- `class DataSwitch`

额外继承的成员函数**Protected 成员函数继承自 `arcs::aubo_scope::ProgramNode`**

- `ProgramNode` ()

11.47.1 详细描述

在文件 `force_node.h` 第 11 行定义.

11.47.2 成员枚举类型说明**ConfigType**

```
enum arcs::aubo_scope::ForceNode::ConfigType
```

The configuration type used to determine which type of configuration this instance is.

枚举值

SIMPLE	Simple type is selected. This type has no further information. Setting this type of config will be ignored and will have no effect on the node.
FRAME	Frame type is selected. The config instance can be cast to <code>FrameForceNodeConfig</code> .
POINT	Point type is selected. This type has no further information. Setting this type of config will be ignored and will have no effect on the node.
MOTION	Motion type is selected. This type has no further information. Setting this type of config will be ignored and will have no effect on the node.

在文件 `force_node.h` 第 18 行定义.

11.47.3 构造及析构函数说明

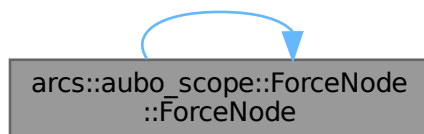
ForceNode() [1/3]

```
arcs::aubo_scope::ForceNode::ForceNode (  
    ForceNode & f)
```

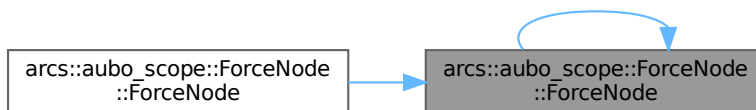
引用了 [ForceNode\(\)](#).

被这些函数引用 [ForceNode\(\)](#) , 以及 [ForceNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

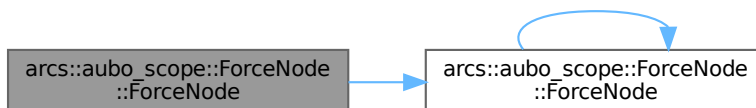


ForceNode() [2/3]

```
arcs::aubo_scope::ForceNode::ForceNode (  
    ForceNode && f)
```

引用了 [ForceNode\(\)](#).

函数调用图:



~ForceNode()

```
virtual arcs::aubo_scope::ForceNode::~~ForceNode () [virtual]
```

ForceNode() [3/3]

```
arcs::aubo_scope::ForceNode::ForceNode () [private]
```

11.47.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [force_node.h](#) 第 66 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.47.5 类成员变量说明

d_

```
void* arcs::aubo_scope::ForceNode::d_ { nullptr } [private]
```

在文件 [force_node.h](#) 第 68 行定义.

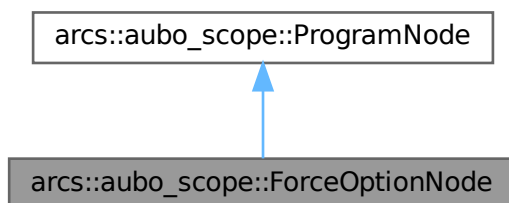
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/force_node.h](#)

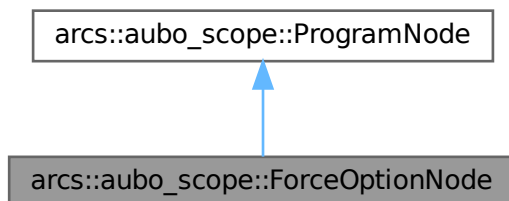
11.48 arcs::aubo_scope::ForceOptionNode 类参考

```
#include <force_option_node.h>
```

类 arcs::aubo_scope::ForceOptionNode 继承关系图:



arcs::aubo_scope::ForceOptionNode 的协作图:



Public 成员函数

- [ForceOptionNode](#) (ForceOptionNode &f)
- [ForceOptionNode](#) (ForceOptionNode &&f)
- virtual [~ForceOptionNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [ForceOptionNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数**Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)**

- [ProgramNode](#) ()

11.48.1 详细描述

在文件 [force_option_node.h](#) 第 10 行定义.

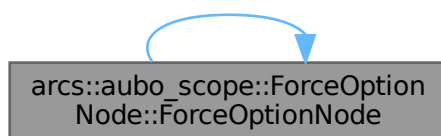
11.48.2 构造及析构函数说明**ForceOptionNode() [1/3]**

```
arcs::aubo_scope::ForceOptionNode::ForceOptionNode (
    ForceOptionNode & f)
```

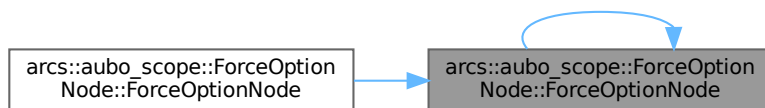
引用了 [ForceOptionNode\(\)](#).

被这些函数引用 [ForceOptionNode\(\)](#) , 以及 [ForceOptionNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

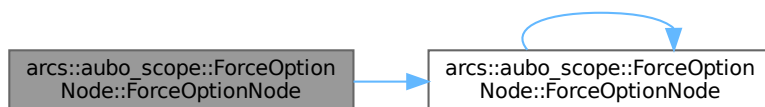


ForceOptionNode() [2/3]

```
arcs::aubo_scope::ForceOptionNode::ForceOptionNode (
    ForceOptionNode && f)
```

引用了 [ForceOptionNode\(\)](#).

函数调用图:



~ForceOptionNode()

```
virtual arcs::aubo_scope::ForceOptionNode::~~ForceOptionNode () [virtual]
```

ForceOptionNode() [3/3]

```
arcs::aubo_scope::ForceOptionNode::ForceOptionNode () [private]
```

11.48.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [force_option_node.h](#) 第 18 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.48.4 类成员变量说明

d_

```
void* arcs::aubo_scope::ForceOptionNode::d_ { nullptr } [private]
```

在文件 [force_option_node.h](#) 第 20 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/force_option_node.h](#)

11.49 arcs::aubo_scope::Function 类参考

函数接口表示在表达式编辑器中显示的函数。

```
#include <function.h>
```

Public 成员函数

- [Function](#) (Function &f)
- [Function](#) (Function &&f)
- [~Function](#) ()
- std::string [getName](#) ()
获取函数名称
- std::vector< std::string > [getArguments](#) ()
获取函数参数列表
- AuboCapInfoPtr [getContributorInfo](#) ()
获取将此函数添加/贡献到 *AuboScope* 的 *AuboCap* 信息。

Private 成员函数

- [Function](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.49.1 详细描述

函数接口表示在表达式编辑器中显示的函数。

另请参见[FunctionModel](#)。

在文件 [function.h](#) 第 25 行定义.

11.49.2 构造及析构函数说明

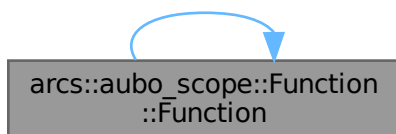
Function() [1/3]

```
arcs::aubo_scope::Function::Function (  
    Function & f)
```

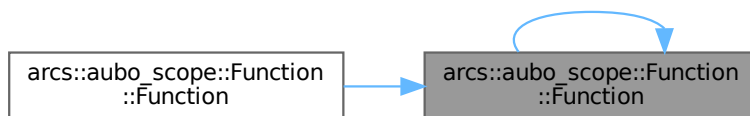
引用了 [Function\(\)](#).

被这些函数引用 [Function\(\)](#) , 以及 [Function\(\)](#).

函数调用图:



这是这个函数的调用关系图:

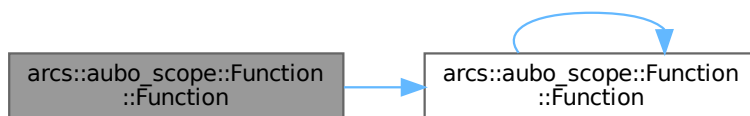


Function() [2/3]

```
arcs::aubo_scope::Function::Function (
    Function && f)
```

引用了 [Function\(\)](#).

函数调用图:



~Function()

```
arcs::aubo_scope::Function::~~Function ()
```

Function() [3/3]

```
arcs::aubo_scope::Function::Function () [private]
```

11.49.3 成员函数说明

getArguments()

```
std::vector< std::string > arcs::aubo_scope::Function::getArguments ()
```

获取函数参数列表

返回

函数参数列表。

getContributorInfo()

```
AuboCapInfoPtr arcs::aubo_scope::Function::getContributorInfo ()
```

获取将此函数添加/贡献到AuboScope 的AuboCap 信息。

返回

此函数的AuboCap 信息。

getName()

std::string arcs::aubo_scope::Function::getName ()
 获取函数名称

返回

函数名称。

11.49.4 友元及相关符号说明**DataSwitch**

friend class DataSwitch [friend]
 在文件 [function.h](#) 第 71 行定义.
 引用了 [DataSwitch](#).
 被这些函数引用 [DataSwitch](#).

11.49.5 类成员变量说明**d_**

void* arcs::aubo_scope::Function::d_ { nullptr } [private]
 在文件 [function.h](#) 第 73 行定义.
 该类的文档由以下文件生成:

- include/aubo_caps/domain/function/[function.h](#)

11.50 arcs::aubo_scope::FunctionModel 类参考

函数模型此接口提供在表达式编辑器中添加和删除脚本函数的方法，该编辑器可从赋值和 If 程序节点等位置访问。

```
#include <function_model.h>
```

Public 成员函数

- [FunctionModel](#) (FunctionModel &f)
- [FunctionModel](#) (FunctionModel &&f)
- [~FunctionModel](#) ()
- FunctionPtr [addFunction](#) (const std::string &name, const std::vector< std::string > &argument←Names)
 添加脚本函数到表达式编辑器。
- bool [removeFunction](#) (FunctionPtr func)
 移除函数。只能移除此 *AuboCap* 添加的函数。
- FunctionPtr [getFunction](#) (const std::string &name)
 根据名称获取函数
- std::vector< FunctionPtr > [getFunctions](#) ()
 获取所有已添加的函数

Private 成员函数

- [FunctionModel](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.50.1 详细描述

函数模型此接口提供在表达式编辑器中添加和删除脚本函数的方法，该编辑器可从赋值和 If 程序节点等位置访问。

添加的函数仅为终端用户提供了一种便捷的方式来使用贡献的函数。特定脚本函数的实现必须通过 `InstallationNodeContribution#generateScript(ScriptWriter)` 添加到程序的前导部分。

所有可贡献的函数都必须始终添加到前导部分，无论它们是否被添加到表达式编辑器中。否则，依赖脚本函数贡献的旧程序可能会停止工作。

另请参见 [Function](#)。

在文件 `function_model.h` 第 45 行定义。

11.50.2 构造及析构函数说明

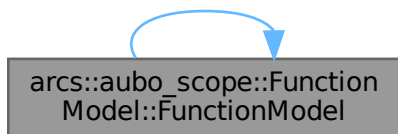
FunctionModel() [1/3]

```
arcs::aubo_scope::FunctionModel::FunctionModel (
    FunctionModel & f)
```

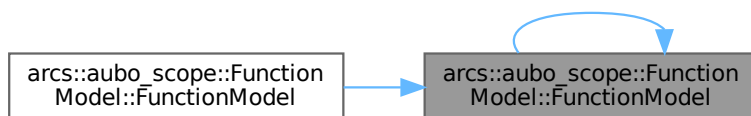
引用了 [FunctionModel\(\)](#)。

被这些函数引用 [FunctionModel\(\)](#)，以及 [FunctionModel\(\)](#)。

函数调用图:



这是这个函数的调用关系图:

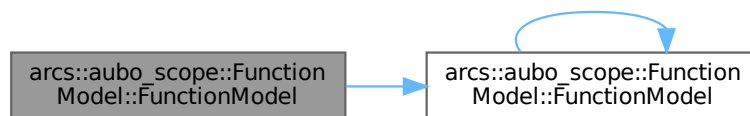


FunctionModel() [2/3]

```
arcs::aubo_scope::FunctionModel::FunctionModel (
    FunctionModel && f)
```

引用了 [FunctionModel\(\)](#)。

函数调用图:



~FunctionModel()

```
arcs::aubo_scope::FunctionModel::~~FunctionModel ()
```

FunctionModel() [3/3]

```
arcs::aubo_scope::FunctionModel::FunctionModel () [private]
```

11.50.3 成员函数说明

addFunction()

```
FunctionPtr arcs::aubo_scope::FunctionModel::addFunction (
    const std::string & name,
    const std::vector< std::string > & argumentNames)
```

添加脚本函数到表达式编辑器。

参数

<i>name</i>	函数名称。有效名称必须匹配正则表达式[a-zA-Z][a-zA-Z0-9_]{0,24}，共 25 个字符。部分AUBO Robot 保留名称不可使用。
<i>argumentNames</i>	函数参数提示列表，在表达式编辑器中以逗号分隔显示在尖括号内。最多指定五个参数。

返回

成功添加返回表示函数的对象，否则返回 null。

异常

<i>FunctionException</i>	如果名称非法或指定了超过五个参数。

getFunction()

```
FunctionPtr arcs::aubo_scope::FunctionModel::getFunction (
    const std::string & name)
```

根据名称获取函数

参数

<i>name</i>	函数名称。

返回

找到返回函数， 否则返回 `null`。

getFunctions()

```
std::vector< FunctionPtr > arcs::aubo_scope::FunctionModel::getFunctions ()
```

获取所有已添加的函数

返回

已添加函数的集合。

removeFunction()

```
bool arcs::aubo_scope::FunctionModel::removeFunction (  
    FunctionPtr func)
```

移除函数。只能移除此AuboCap 添加的函数。

参数

<i>function</i>	要移除的函数。

返回

成功移除返回 `true`， 否则返回 `false`。

11.50.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [function_model.h](#) 第 117 行定义。
引用了 [DataSwitch](#)。
被这些函数引用 [DataSwitch](#)。

11.50.5 类成员变量说明

d_

```
void* arcs::aubo_scope::FunctionModel::d_ { nullptr } [private]
```

在文件 [function_model.h](#) 第 119 行定义。
该类的文档由以下文件生成:

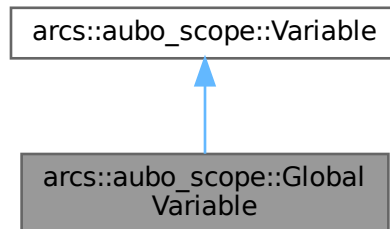
- `include/aubo_caps/domain/function/function_model.h`

11.51 arcs::aubo_scope::GlobalVariable 类参考

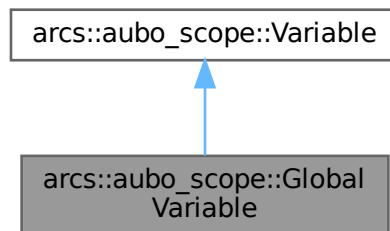
Global variables are available everywhere in a program.

```
#include <global_variable.h>
```

类 arcs::aubo_scope::GlobalVariable 继承关系图:



arcs::aubo_scope::GlobalVariable 的协作图:



Public 成员函数

- [GlobalVariable](#) (GlobalVariable &f)
- [GlobalVariable](#) (GlobalVariable &&f)
- virtual [~GlobalVariable](#) ()

Public 成员函数继承自 [arcs::aubo_scope::Variable](#)

- [Variable](#) (Variable &f)
- [Variable](#) (Variable &&f)
- virtual [~Variable](#) ()
- [Type](#) [getType](#) ()
获取变量类型
- std::string [getDisplayName](#) ()
获取变量显示名称。注意：变量可随时被重命名。
- bool [equals](#) (VariablePtr object)
判断是否引用相同的内部变量
- size_t [hashCode](#) ()

获取哈希码

- `std::string toString ()`
获取变量的字符串表示

Protected 成员函数

- `GlobalVariable ()`

Protected 成员函数继承自 `arcs::aubo_scope::Variable`

- `Variable ()`

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数

Public 类型继承自 `arcs::aubo_scope::Variable`

- `enum Type : int { GLOBAL , VALUE_PERSISTED , FEATURE }`

The variable type used to determine which type of variable this instance is.

11.51.1 详细描述

Global variables are available everywhere in a program.

在文件 `global_variable.h` 第 13 行定义.

11.51.2 构造及析构函数说明

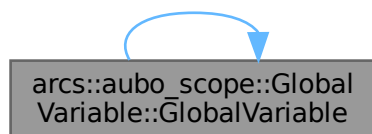
GlobalVariable() [1/3]

```
arcs::aubo_scope::GlobalVariable::GlobalVariable (
    GlobalVariable & f)
```

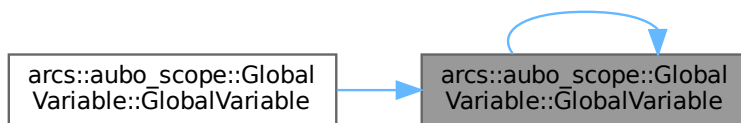
引用了 `GlobalVariable()`.

被这些函数引用 `GlobalVariable()` , 以及 `GlobalVariable()`.

函数调用图:



这是这个函数的调用关系图:

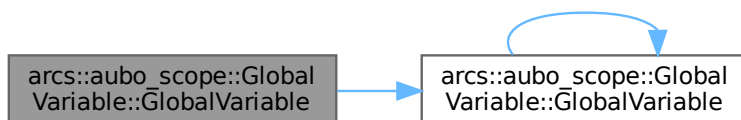


GlobalVariable() [2/3]

```
arcs::aubo_scope::GlobalVariable::GlobalVariable (
    GlobalVariable && f)
```

引用了 [GlobalVariable\(\)](#).

函数调用图:



~GlobalVariable()

```
virtual arcs::aubo_scope::GlobalVariable::~~GlobalVariable () [virtual]
```

GlobalVariable() [3/3]

```
arcs::aubo_scope::GlobalVariable::GlobalVariable () [protected]
```

11.51.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [global_variable.h](#) 第 24 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.51.4 类成员变量说明

d_

```
void* arcs::aubo_scope::GlobalVariable::d_ { nullptr } [private]
```

在文件 [global_variable.h](#) 第 25 行定义.

该类的文档由以下文件生成:

- `include/aubo_caps/domain/variable/global_variable.h`

11.52 arcs::aubo_scope::GlobalVariableFilter 结构体参考

```
#include <global_variable.h>
```

Public 成员函数

- bool [operator\(\)](#) (VariablePtr var)

11.52.1 详细描述

在文件 [global_variable.h](#) 第 28 行定义.

11.52.2 成员函数说明

[operator\(\)](#)()

```
bool arcs::aubo_scope::GlobalVariableFilter::operator() (
    VariablePtr var) [inline]
```

在文件 [global_variable.h](#) 第 30 行定义.

引用了 [arcs::aubo_scope::Variable::GLOBAL](#).

该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/variable/global_variable.h](#)

11.53 arcs::aubo_scope::GripActionParameters 类参考

```
#include <grip_action_parameters.h>
```

Public 成员函数

- virtual [~GripActionParameters](#) ()=default
- virtual double [getForce](#) ()=0
- virtual double [getWidth](#) ()=0
- virtual double [getVacuum](#) ()=0
- virtual double [getSpeed](#) ()=0
- virtual bool [isGripDetectionEnabled](#) ()=0

11.53.1 详细描述

This interface represents parameters for a gripping action defined/configured by the end user.

These parameters are used when generating script code for a gripping action (see [GripperContribution#generateGrip](#)).

在文件 [grip_action_parameters.h](#) 第 21 行定义.

11.53.2 构造及析构函数说明

[~GripActionParameters](#)()

```
virtual arcs::aubo_scope::GripActionParameters::~~GripActionParameters () [virtual], [default]
```

11.53.3 成员函数说明

[getForce](#)()

```
virtual double arcs::aubo_scope::GripActionParameters::getForce () [pure virtual]
```

Get the selected force to be used the gripping action.

This is only valid, if the force capability has been registered (using the method [double, double, Force.Unit](#)).

返回

The force to use for the gripping action

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the force capability

getSpeed()

```
virtual double arcs::aubo_scope::GripActionParameters::getSpeed () [pure virtual]
```

Get the selected speed to be used for the gripping action.

This is only valid, if the speed capability has been registered (using the method `double, double, double, Speed.Unit`).

返回

The speed to use for the gripping action

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the speed capability

getVacuum()

```
virtual double arcs::aubo_scope::GripActionParameters::getVacuum () [pure virtual]
```

Get the selected vacuum level to be used for the gripping action.

This is only valid, if the vacuum capability has been registered (using the method `double, double, Pressure.Unit`).

返回

The vacuum level to use for the gripping action

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the vacuum capability

getWidth()

```
virtual double arcs::aubo_scope::GripActionParameters::getWidth () [pure virtual]
```

Get the selected width to be used for the gripping action.

This is only valid, if the width capability has been registered (using the method `double, double, double, Length.Unit`).

返回

The width to use for the gripping action

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the width capability

isGripDetectionEnabled()

```
virtual bool arcs::aubo_scope::GripActionParameters::isGripDetectionEnabled () [pure virtual]
```

Use this method to determine whether the grip detection option in the Gripper program node is on or off.

Note that this method will report that grip detection is off, when script code is to be generated due to the user testing a grip configuration of a gripper program or operating the gripper using the toolbar.

This method should only be called, if the grip detected feedback capability has been registered (using `GripperFeedbackCapabilities#registerGripDetectedCapability(ScriptCodeGenerator)`).

The purpose of this method is to ensure that the appropriate script code for performing a grip action is generated when the end user has specified a new payload value. The user-defined payload value will be applied by AuboScope immediately after the generated grip action script code has finished executing. Depending on the enablement state of grip detection, the generated script should have the following properties:

- *Grip detection off:* The script code should not finish earlier than when it is appropriate to apply the new payload value (i.e. when the object has been gripped). This typically means that the script code should wait a time period corresponding to closing the gripper's "fingers" fully (if the gripper is vacuum operated then long enough to achieve some level of vacuum).
- *Grip detection on:* The payload will only be set if the grip detection was successful, i.e. it will be applied immediately after a grip was detected. This means that it is not necessary for the script code to wait (since the built-in detection timeout functionality will handle the waiting part).

返回

`true` if the end user has enabled grip detection, and `false` if grip detection is disabled.

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the grip detected feedback capability

该类的文档由以下文件生成:

- `include/aubo_caps/contribution/driver/gripper/grip_action_parameters.h`

11.54 arcs::aubo_scope::GripForceCapability 类参考

夹持力能力此接口表示夹爪已注册的力度能力，支持使用用户可配置的力度进行夹取。

```
#include <grip_force_capability.h>
```

Public 成员函数

- virtual `~GripForceCapability ()=0`
- virtual void `updateCapability (double minForce, double maxForce, double defaultGripForce)=0`

11.54.1 详细描述

夹持力能力此接口表示夹爪已注册的力度能力，支持使用用户可配置的力度进行夹取。

通过此接口，可以在能力注册后调整力度的范围和默认值。这通常不适用于大多数夹爪，但当范围取决于另一个自定义夹爪设置（可由用户配置）时，这可能是相关的。

在文件 `grip_force_capability.h` 第 29 行定义。

11.54.2 构造及析构函数说明**~GripForceCapability()**

```
virtual arcs::aubo_scope::GripForceCapability::~~GripForceCapability () [pure virtual]
```

11.54.3 成员函数说明

updateCapability()

```
virtual void arcs::aubo_scope::GripForceCapability::updateCapability (
    double minForce,
    double maxForce,
    double defaultGripForce) [pure virtual]
```

Update the range and default value for the force supported by the gripper.

Updating the range could result in an already entered value in the configuration of a gripper program node being outside the range. This will make the given program node undefined, and force the end user to adjust the value to be valid again.

参数

<i>minForce</i>	minimum force supported by the gripper
<i>maxForce</i>	maximum force supported by the gripper
<i>defaultGripForce</i>	default value for the force for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
<i>unit</i>	the unit for all specified values, null

异常

<i>InvalidCapabilityRange</i>	if <code>minForce > maxForce</code> or <code>defaultGripForce</code> is outside the range defined by <code>minForce</code> and <code>maxForce</code>

该类的文档由以下文件生成:

- include/aubo_caps/contribution/driver/gripper/capability/[grip_force_capability.h](#)

11.55 arcs::aubo_scope::GripperApiProvider 类参考

夹爪 API 提供者此接口提供对 AuboScope 中与夹爪相关的功能和服务的访问。

```
#include <gripper_api_provider.h>
```

Public 成员函数

- virtual [~GripperApiProvider](#) ()=default
- virtual SystemApiPtr [getSystemApi](#) ()=0
- virtual IoModelPtr [getIoModel](#) ()=0

11.55.1 详细描述

夹爪 API 提供者此接口提供对 AuboScope 中与夹爪相关的功能和服务的访问。

在文件 [gripper_api_provider.h](#) 第 22 行定义.

11.55.2 构造及析构函数说明

~GripperApiProvider()

```
virtual arcs::aubo_scope::GripperApiProvider::~~GripperApiProvider () [virtual], [default]
```

11.55.3 成员函数说明

getIoModel()

```
virtual IoModelPtr arcs::aubo_scope::GripperApiProvider::getIoModel () [pure virtual]
```

返回

I/O 模型接口（例如用于获取在设置夹爪的自定义用户输入中显示的 I/O）

getSystemApi()

```
virtual SystemApiPtr arcs::aubo_scope::GripperApiProvider::getSystemApi () [pure virtual]
```

返回

提供系统相关功能访问的接口

该类的文档由以下文件生成:

- include/aubo_caps/contribution/driver/gripper/[gripper_api_provider.h](#)

11.56 arcs::aubo_scope::GripperConfiguration 类参考

This interface provides access to register or setup various properties and capabilities of a gripper.

```
#include <gripper_configuration.h>
```

Public 成员函数

- virtual [~GripperConfiguration](#) ()=default
- virtual WidthCapabilityPtr [registerWidthCapability](#) (double minWidth, double maxWidth, double defaultGripWidth, double defaultReleaseWidth)=0
Register a width/position capability for a gripper that supports moving to a user configurable position (open/close to a configurable width).
- virtual GripForceCapabilityPtr [registerGrippingForceCapability](#) (double minForce, double maxForce, double defaultGripForce)=0
Register a force capability for a gripper that supports gripping using a user configurable force.
- virtual SpeedCapabilityPtr [registerSpeedCapability](#) (double minSpeed, double maxSpeed, double defaultGripSpeed, double defaultReleaseSpeed)=0
Register a speed capability for a gripper that supports operating using a user configurable speed.
- virtual GripVacuumCapabilityPtr [registerGrippingVacuumCapability](#) (double minVacuum, double maxVacuum, double defaultGripVacuum)=0
- virtual void [registerGripDetectedCapability](#) (std::function< void(ScriptWriterPtr scriptWriter)> scriptCodeGenerator)=0
- virtual void [registerReleaseDetectedCapability](#) (std::function< void(ScriptWriterPtr scriptWriter)> scriptCodeGenerator)=0

11.56.1 详细描述

This interface provides access to register or setup various properties and capabilities of a gripper.

It is optional to register these capabilities and properties.

在文件 [gripper_configuration.h](#) 第 20 行定义.

11.56.2 构造及析构函数说明

~GripperConfiguration()

```
virtual arcs::aubo_scope::GripperConfiguration::~~GripperConfiguration () [virtual], [default]
```

11.56.3 成员函数说明

registerGripDetectedCapability()

```
virtual void arcs::aubo_scope::GripperConfiguration::registerGripDetectedCapability (
    std::function< void(ScriptWriterPtr scriptWriter)> scriptCodeGenerator) [pure virtual]
```

[Register](#) this feedback capability if the gripper can inform AuboScope whether it has detected that an object has been gripped, after a grip action has been triggered. This will enable the end user to configure the actions to take when the gripper has detected that an object has been gripped and/or when the grip action timed out.

The provided implementation of the `ScriptCodeGenerator` interface must generate the script code for determining if a grip was detected. The return value of the script code **must** be a boolean, i.e. 'True' or 'False'.

Note: When this capability has been registered, the `GripperContribution#generateGripActionScript(ScriptWriter, Object)` method should generate different grip action script code depending on whether the end user has enabled or disabled the grip detection option (in the Gripper program node). For more details, see Javadoc for [GripActionParameters#isGripDetectionEnabled\(\)](#).

参数

<i>scriptCodeGenerator</i>	script generator that generates the script code for determining if a grip was detected. When the script code needs to be generated, the method <code>ScriptCodeGenerator#generateScript(ScriptWriter, Object)</code> will be called (by AuboScope). The resulting script code will be embedded in a script function and must return a boolean, i.e. 'True' or 'False'.

异常

<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered
<i>CalledOutsideGripperConfigurationPhase</i>	if this method is called at the wrong time, i.e. outside the scope of the <code>GripperContribution#configure←Gripper(GripperConfiguration, GripperAPIProvider)</code> method.

registerGrippingForceCapability()

```
virtual GripForceCapabilityPtr arcs::aubo_scope::GripperConfiguration::registerGrippingForceCapability (
    double minForce,
    double maxForce,
    double defaultGripForce) [pure virtual]
```

[Register](#) a force capability for a gripper that supports gripping using a user configurable force.

参数

<i>minForce</i>	minimum force supported by the gripper
<i>maxForce</i>	maximum force supported by the gripper
<i>defaultGripForce</i>	default value for the force for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
<i>unit</i>	the unit for all specified values, null

返回

a force capability allowing for dynamic adjustments of the capability properties including the value range and default value

异常

<i>InvalidCapabilityRange</i>	if <code>minForce > maxForce</code> or <code>defaultGripForce</code> is outside the range defined by <code>minForce</code> and <code>maxForce</code> .
<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered
<i>CalledOutsideGripperConfigurationPhase</i>	if this method is called at the wrong time by a <code>AuboCap</code> implementing the GripperContribution interface, i.e. called outside the scope of the configuration of the gripper.
<i>CalledOutsideMethodScope</i>	if this method is called at the wrong time by a (regular) <code>AuboCap</code> which has registered its program node contribution/service as conformant with a <code>AuboScope</code> gripper (using <code>GripperRegistrationManager</code> ↵ <code>#registerAsGripper(Class)</code>), i.e. called outside the scope of the call to either <code>ProgramNodeService#configureContribution</code> ↵ <code>ContributionConfiguration</code>) (for a Qt-based <code>AuboCap</code>) or <code>ProgramNodeServiceConfigurable#configure</code> ↵ <code>Contribution(ProgramNodeConfiguration)</code> (for a HTML-based <code>AuboCap</code>).

registerGrippingVacuumCapability()

```
virtual GripVacuumCapabilityPtr arcs::aubo_scope::GripperConfiguration::registerGrippingVacuumCapability(
    double minVacuum,
    double maxVacuum,
    double defaultGripVacuum) [pure virtual]
```

[Register](#) a vacuum capability for a gripper that supports gripping using a user configurable vacuum level.

If the gripper works with absolute pressure, specify a positive range for the supported vacuum level. If the gripper works with relative pressure (vacuum), specify a negative range for the supported vacuum level.

参数

<i>minVacuum</i>	minimum vacuum level supported by the gripper
<i>maxVacuum</i>	maximum vacuum level supported by the gripper
<i>defaultGripVacuum</i>	default value for the vacuum level for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
<i>unit</i>	the unit for all specified values, not null

返回

a vacuum capability allowing for dynamic adjustments of the capability properties including the value range and default value

异常

<i>InvalidCapabilityRange</i>	if minVacuum > maxVacuum or defaultGripVacuum is outside the range defined by minVacuum and maxVacuum.
<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered
<i>CalledOutsideGripperConfigurationPhase</i>	if this method is called at the wrong time by a AuboCap implementing the GripperContribution interface, i.e. called outside the scope of the configuration of the gripper.
<i>CalledOutsideMethodScope</i>	if this method is called at the wrong time by a (regular) AuboCap which has registered its program node contribution/service as conformant with a AuboScope gripper (using GripperRegistrationManager ← #registerAsGripper(Class)), i.e. called outside the scope of the call to either ProgramNodeService#configureContribution (← ContributionConfiguration) (for a Qt-based AuboCap) or ProgramNodeServiceConfigurable#configure ← Contribution(ProgramNodeConfiguration) (for a HTML-based AuboCap).

registerReleaseDetectedCapability()

```
virtual void arcs::aubo_scope::GripperConfiguration::registerReleaseDetectedCapability (
    std::function< void(ScriptWriterPtr scriptWriter)> scriptCodeGenerator) [pure virtual]
```

[Register](#) this feedback capability if the gripper can inform AuboScope whether it has detected an object has been released, after a release action has been triggered. This will enable the end user to configure the actions to take when the gripper has detected that an object has been released and/or when the release action timed out.

The provided implementation of the [ScriptCodeGenerator](#) interface must generate the script code for determining if a release was detected. The return value of the script code **must** be a boolean, i.e. 'True' or 'False'.

Note: When this capability has been registered, the [GripperContribution#generateReleaseActionScript](#)(Script) method should generate different release action script code depending on whether the end user has enabled or disabled the release detection option (in the Gripper program node). For more details, see Javadoc for [GripActionParameters#isGripDetectionEnabled\(\)](#).

参数

<i>scriptCodeGenerator</i>	script generator that generates the script code for determining if a release was detected. When the script code needs to be generated, the method ScriptCodeGenerator#generateScript (ScriptWriter, Object) will be called (by AuboScope). The resulting script code will be embedded in a script function and must return a boolean, i.e. 'True' or 'False'.

异常

<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered

<i>CalledOutsideGripperConfigurationPhase</i>	if this method is called at the wrong time, i.e. outside the scope of the <code>GripperContribution#configure←Gripper(GripperConfiguration, GripperAPIProvider)</code> method.

registerSpeedCapability()

```
virtual SpeedCapabilityPtr arcs::aubo_scope::GripperConfiguration::registerSpeedCapability (
    double minSpeed,
    double maxSpeed,
    double defaultGripSpeed,
    double defaultReleaseSpeed) [pure virtual]
```

Register a speed capability for a gripper that supports operating using a user configurable speed.

参数

<i>minSpeed</i>	minimum speed supported by the gripper
<i>maxSpeed</i>	maximum speed supported by the gripper
<i>defaultGripSpeed</i>	default value for the speed for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
<i>defaultReleaseSpeed</i>	default value for the speed for a grip action. This value is used for, e.g. performing a "default" release action using the toolbar
<i>unit</i>	the unit for all specified values, not null

返回

a speed capability allowing for dynamic adjustments of the capability properties including the value range and default values

异常

<i>InvalidCapabilityRange</i>	if <code>minSpeed > maxSpeed</code>
<i>InvalidCapabilityRange</i>	if <code>defaultGripSpeed</code> or <code>defaultReleaseSpeed</code> is outside the range defined by <code>minSpeed</code> and <code>maxSpeed</code> .
<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered
<i>CalledOutsideGripperConfigurationPhase</i>	if this method is called at the wrong time by a <code>AuboCap</code> implementing the <code>GripperContribution</code> interface, i.e. called outside the scope of the configuration of the gripper.

<i>CalledOutsideMethodScope</i>	if this method is called at the wrong time by a (regular) AuboCap which has registered its program node contribution/service as conformant with a AuboScope gripper (using <code>GripperRegistrationManager←#registerAsGripper(Class)</code>), i.e. called outside the scope of the call to either <code>ProgramNodeService#configureContribution(←ContributionConfiguration)</code> (for a Qt-based AuboCap) or <code>ProgramNodeServiceConfigurable#configure←Contribution(ProgramNodeConfiguration)</code> (for a HTML-based AuboCap).

registerWidthCapability()

```
virtual WidthCapabilityPtr arcs::aubo_scope::GripperConfiguration::registerWidthCapability (
    double minWidth,
    double maxWidth,
    double defaultGripWidth,
    double defaultReleaseWidth) [pure virtual]
```

[Register](#) a width/position capability for a gripper that supports moving to a user configurable position (open/close to a configurable width).

参数

<i>minWidth</i>	The minimum width supported by the gripper
<i>maxWidth</i>	The maximum width supported by the gripper
<i>defaultGripWidth</i>	default value for the width for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
<i>defaultReleaseWidth</i>	default value for the width for a release action. This value is used for, e.g. performing a "default" release action using the toolbar
<i>unit</i>	the unit for all specified values, not null

返回

a width capability allowing for dynamic adjustments of the capability properties including the value range and default values

异常

<i>InvalidCapabilityRange</i>	if <code>minWidth > maxWidth</code>
<i>InvalidCapabilityRange</i>	if <code>defaultGripWidth</code> or <code>defaultReleaseWidth</code> is outside the range defined by <code>minWidth</code> and <code>maxWidth</code> .
<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered
<i>CalledOutsideGripperConfigurationPhase</i>	if this method is called at the wrong time by a AuboCap implementing the GripperContribution interface, i.e. called outside the scope of the configuration of the gripper.

<i>CalledOutsideMethodScope</i>	if this method is called at the wrong time by a (regular) AuboCap which has registered its program node contribution/service as conformant with a AuboScope gripper (using <code>GripperRegistrationManager←#registerAsGripper(Class)</code>), i.e. called outside the scope of the call to either <code>ProgramNodeService#configureContribution(←ContributionConfiguration)</code> (for a Qt-based AuboCap) or <code>ProgramNodeServiceConfigurable#configure←Contribution(ProgramNodeConfiguration)</code> (for a HTML-based AuboCap).

该类的文档由以下文件生成:

- `include/aubo_caps/contribution/driver/gripper/gripper_configuration.h`

11.57 arcs::aubo_scope::GripperContribution 类参考

夹爪贡献此接口定义了向 AuboScope 添加夹爪设备功能的 API。

```
#include <gripper_contribution.h>
```

Public 成员函数

- virtual `~GripperContribution()`=default
- virtual `std::string getTitle()`=0
此方法必须返回夹爪贡献的标题。标题显示在以下位置:
- virtual void `configureGripper` (GripperConfigurationPtr gripperConfiguration, GripperApiProvider←Ptr apiProvider)=0
- virtual void `generateGripActionScript` (ScriptWriterPtr scriptWriter, GripActionParametersPtr parameters)=0
- virtual void `generateReleaseActionScript` (ScriptWriterPtr scriptWriter, ReleaseActionParameters←Ptr parameters)=0

11.57.1 详细描述

夹爪贡献此接口定义了向 AuboScope 添加夹爪设备功能的 API。

实现此接口将创建程序节点贡献、安装贡献和工具栏贡献，向终端用户暴露夹爪功能。请注意，工具栏贡献在 CB3 机器人上不受支持。

程序节点贡献允许终端用户编程夹取和释放操作。终端用户可用的配置选项基于夹爪支持的属性和功能。至少，所有夹爪必须支持”默认”夹取和释放操作。所有其他功能是可选的，可以在调用 `GripperContribution#configureGripper(GripperConfiguration, GripperAPIProvider)` 方法时注册。

安装节点贡献可在需要终端用户提供任何自定义输入来设置夹爪时使用（例如指定 IP 地址）。相应的生成 UI 将在安装节点中可访问。此功能是可选的，因为此类配置属性是特定于每个夹爪的。当调用 `GripperContribution#configureInstallation(CustomUserInputConfiguration, System←Configuration, TCPConfiguration, GripperAPIProvider)` 方法时，提供创建基于用户输入的自定义配置的访问权限。如果没有定义自定义用户输入、文本等，安装节点将不会在 AuboScope 中可见。前言脚本代码（由 `generatePreambleScript(ScriptWriter)` 生成）仍将添加到机器人程序中。

工具栏贡献为终端用户提供快速访问，以使用”默认”夹爪操作参数执行夹取和释放操作。

在文件 `gripper_contribution.h` 第 76 行定义。

11.57.2 构造及析构函数说明

~GripperContribution()

```
virtual arcs::aubo_scope::GripperContribution::~~GripperContribution () [virtual], [default]
```

11.57.3 成员函数说明

configureGripper()

```
virtual void arcs::aubo_scope::GripperContribution::configureGripper (
    GripperConfigurationPtr gripperConfiguration,
    GripperApiProviderPtr apiProvider) [pure virtual]
```

当调用此方法时，使用 configuration 参数注册或设置夹爪的可选属性和功能。

此方法在夹爪配置阶段调用，在贡献注册后、新安装被加载或创建时调用。

注意：如果夹爪仅支持基本的”默认”夹取和释放操作，请将此方法的实现留空。

参数

<i>gripperConfiguration</i>	可用于注册或设置夹爪属性和功能的配置实例
<i>apiProvider</i>	提供对 AuboScope 中可用于设置夹爪功能的相关功能和服务访问

generateGripActionScript()

```
virtual void arcs::aubo_scope::GripperContribution::generateGripActionScript (
    ScriptWriterPtr scriptWriter,
    GripActionParametersPtr parameters) [pure virtual]
```

当调用此方法时，必须生成使用夹爪执行夹取动作的脚本代码。

终端用户定义/配置的已注册可选夹爪功能参数将作为脚本生成的输入。

此方法在以下场景中调用：

- 当配置了夹取动作的夹爪程序节点应作为要执行的机器人程序的一部分生成脚本代码时
- 当终端用户测试夹爪程序节点的夹取配置时
- 当使用工具栏操作夹爪时

注意：

- 夹取物体后应用新的总负载**不是**夹爪的职责，因为这是 AuboScope 自动处理的。
- 当终端用户指定了新的负载时，用户定义的负载值将在夹取动作的生成脚本代码完成执行后立即由 AuboScope 应用。因此脚本代码不应在适当应用新负载值之前完成。

参数

<i>scriptWriter</i>	使用此脚本编写器实例生成夹取脚本代码
<i>parameters</i>	终端用户定义/配置的夹爪动作参数

generateReleaseActionScript()

```
virtual void arcs::aubo_scope::GripperContribution::generateReleaseActionScript (
    ScriptWriterPtr scriptWriter,
    ReleaseActionParametersPtr parameters) [pure virtual]
```

当调用此方法时，必须生成使用夹爪执行释放动作的脚本代码。

终端用户定义/配置的已注册可选夹爪功能参数将作为脚本生成的输入。

此方法在以下场景中调用：

- 当配置了释放动作的夹爪程序节点应作为要执行的机器人程序的一部分生成脚本代码时
- 当终端用户测试夹爪程序节点的释放配置时

- 当使用工具栏操作夹爪时

注意:

- 释放物体后应用新的总负载**不是**夹爪的职责，因为这是 AuboScope 自动处理的。
- 当终端用户指定了新的负载时，用户定义的负载值将在释放动作的生成脚本代码完成执行后立即由 AuboScope 应用。因此脚本代码不应在适当应用新负载值之前完成。

参数

<i>scriptWriter</i>	使用此脚本编写器实例生成释放夹爪的脚本代码
<i>parameters</i>	终端用户定义/配置的夹爪动作参数

getTitle()

```
virtual std::string arcs::aubo_scope::GripperContribution::getTitle () [pure virtual]
```

此方法必须返回夹爪贡献的标题。标题显示在以下位置:

- 程序标签页的”结构”区域中此类夹爪程序节点的显示（可从此处插入夹爪程序节点）
- 此夹爪贡献的程序节点画面
- 程序树中此夹爪贡献程序节点的标题
- 安装标签页左侧导航中此夹爪贡献的显示
- 此夹爪贡献的安装节点画面
- 此夹爪的工具栏贡献。注意：工具栏贡献在 CB3 机器人上不可用

此方法的返回值不能为 `null`，也不能为空字符串。

此方法在启动时调用一次。

参数

<i>locale</i>	AuboScope 的当前语言环境，可用于支持多种语言的标题

返回

此夹爪贡献的标题，不能为 `null` 也不能为空字符串

该类的文档由以下文件生成:

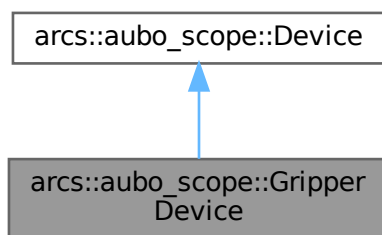
- `include/aubo_caps/contribution/driver/gripper/gripper_contribution.h`

11.58 arcs::aubo_scope::GripperDevice 类参考

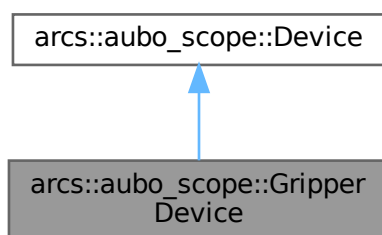
夹爪设备此接口表示 AuboScope 中的夹爪设备。

```
#include <gripper_device.h>
```

类 `arcs::aubo_scope::GripperDevice` 继承关系图:



`arcs::aubo_scope::GripperDevice` 的协作图:



Public 成员函数

- virtual `~GripperDevice()`=default

Public 成员函数继承自 `arcs::aubo_scope::Device`

- virtual `~Device()`=0
- virtual `AuboCapInfoPtr getContributorInfo()`=0
获取将此夹爪设备添加/贡献到 *AuboScope* 的 *AuboCap* 信息。
- virtual `std::string getId()`=0
- virtual `std::string getDisplayName()`=0
- virtual `bool isResolvable()`=0

设备无法保证始终存在于 *AuboScope* 中。如果添加设备的 *AuboCap* 未安装，设备将不可用。此方法用于判断设备是否可用。

11.58.1 详细描述

夹爪设备此接口表示 *AuboScope* 中的夹爪设备。
在文件 `gripper_device.h` 第 20 行定义。

11.58.2 构造及析构函数说明

~GripperDevice()

```
virtual arcs::aubo_scope::GripperDevice::~~GripperDevice () [virtual], [default]
```

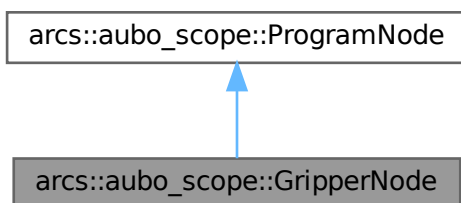
该类的文档由以下文件生成:

- include/aubo_caps/domain/device/[gripper_device.h](#)

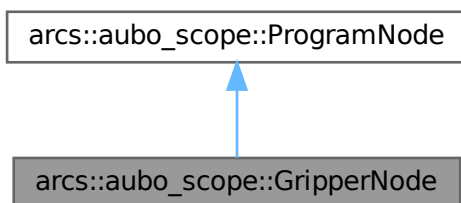
11.59 arcs::aubo_scope::GripperNode 类参考

```
#include <gripper_node.h>
```

类 arcs::aubo_scope::GripperNode 继承关系图:



arcs::aubo_scope::GripperNode 的协作图:



Public 类型

- enum [ConfigType](#) { [UNDEFINED](#) , [GRIP_ACTION](#) , [RELEASE_ACTION](#) }
- *The configuration type used to determine which type of configuration this instance is.*
- enum [SettingType](#) { [UNDEFINED0](#) , [PAYLOAD_SELECTION](#) , [CUSTOM_PARAMETERS](#) }
- *The setting type used to determine which type of setting this instance is.*

Public 成员函数

- [GripperNode](#) ([GripperNode](#) &f)
- [GripperNode](#) ([GripperNode](#) &&f)
- virtual [~GripperNode](#) ()
- [ConfigType](#) [getConfigType](#) ()

返回配置类型。适当转换此实例以访问特定的 *getter*。

- void [setConfigType](#) ([ConfigType](#) type)
- [GripperDevicePtr](#) [getGripperDevice](#) ()
- [SettingType](#) [getSettingType](#) ()

返回设置类型。适当转换此实例以访问特定的 *getter*。

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ([ProgramNode](#) &f)
- [ProgramNode](#) ([ProgramNode](#) &&f)
- virtual [~ProgramNode](#) ()
- [ProgramNodeContributionPtr](#) [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [GripperNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.59.1 详细描述

此接口表示一个程序节点，可用于编程所选夹爪设备的夹取和释放动作。

此接口的实例可使用 [GripperNodeFactory](#) 接口构建。

在文件 [gripper_node.h](#) 第 25 行定义。

11.59.2 成员枚举类型说明

ConfigType

enum [arcs::aubo_scope::GripperNode::ConfigType](#)

The configuration type used to determine which type of configuration this instance is.

枚举值

UNDEFINED	A Gripper action (i.e. grip or release) has not been selected The config instance can be cast to UndefinedActionConfig . This is only relevant if the Gripper node uses a multi-gripper device, i.e. a gripper device supporting the multi-gripper capability (see the MultiGripperSupport interface). The instance will provide access to the individual gripper selection.
GRIP_ACTION	The gripper is configured for a grip action The config instance can be cast to GripActionConfig .

RELEASE_ACTION	The gripper is configured for a release action The config instance can be cast to <code>ReleaseActionConfig</code> .

在文件 `gripper_node.h` 第 32 行定义.

SettingType

`enum arcs::aubo_scope::GripperNode::SettingType`
The setting type used to determine which type of setting this instance is.

枚举值

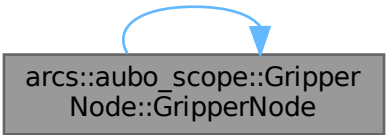
UNDEFINED0	No selection has been made.
PAYLOAD_SELECTION	A payload (from the installation) is selected. The setting instance can be cast to <code>PayloadSelection</code> .
CUSTOM_PARAMETERS	Custom payload parameters are specified. The setting instance can be cast to <code>CustomPayloadParameters</code> .

在文件 `gripper_node.h` 第 101 行定义.

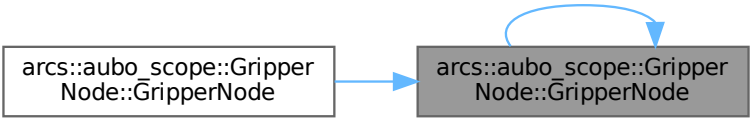
11.59.3 构造及析构函数说明

GripperNode() [1/3]

`arcs::aubo_scope::GripperNode::GripperNode (`
 `GripperNode & f)`
引用了 `GripperNode()`.
被这些函数引用 `GripperNode()` , 以及 `GripperNode()`.
函数调用图:



这是这个函数的调用关系图:

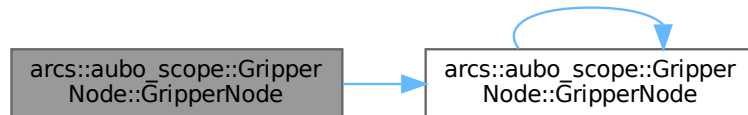


GripperNode() [2/3]

```
arcs::aubo_scope::GripperNode::GripperNode (  
    GripperNode && f)
```

引用了 [GripperNode\(\)](#).

函数调用图:

**~GripperNode()**

```
virtual arcs::aubo_scope::GripperNode::~~GripperNode () [virtual]
```

GripperNode() [3/3]

```
arcs::aubo_scope::GripperNode::GripperNode () [private]
```

11.59.4 成员函数说明**getConfigType()**

```
ConfigType arcs::aubo_scope::GripperNode::getConfigType ()
```

返回配置类型。适当转换此实例以访问特定的 getter。

返回

此配置的类型

getGripperDevice()

```
GripperDevicePtr arcs::aubo_scope::GripperNode::getGripperDevice ()
```

返回

此夹爪节点使用的夹爪设备

getSettingType()

```
SettingType arcs::aubo_scope::GripperNode::getSettingType ()
```

返回设置类型。适当转换此实例以访问特定的 getter。

返回

此负载设置的类型

setConfigType()

```
void arcs::aubo_scope::GripperNode::setConfigType (  
    ConfigType type)
```

11.59.5 友元及相关符号说明

DataSwitch

friend class DataSwitch [friend]
在文件 [gripper_node.h](#) 第 144 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.59.6 类成员变量说明

d_

void* arcs::aubo_scope::GripperNode::d_ { nullptr } [private]
在文件 [gripper_node.h](#) 第 146 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/gripper_node.h](#)

11.60 arcs::aubo_scope::GripVacuumCapability 类参考

真空夹持能力此接口表示夹爪已注册的真空能力，支持使用用户可配置的真空度进行夹取。

#include <grip_vacuum_capability.h>

Public 成员函数

- virtual [~GripVacuumCapability](#) ()=0
- virtual void [updateCapability](#) (double minVacuum, double maxVacuum, double defaultGripVacuum)=0

11.60.1 详细描述

真空夹持能力此接口表示夹爪已注册的真空能力，支持使用用户可配置的真空度进行夹取。

通过此接口，可以在能力注册后调整真空度的范围和默认值。这通常不适用于大多数夹爪，但当范围取决于另一个自定义夹爪设置（可由用户配置）时，这可能是相关的。

在文件 [grip_vacuum_capability.h](#) 第 30 行定义.

11.60.2 构造及析构函数说明

~GripVacuumCapability()

```
virtual arcs::aubo_scope::GripVacuumCapability::~~GripVacuumCapability () [pure virtual]
```

11.60.3 成员函数说明

updateCapability()

```
virtual void arcs::aubo_scope::GripVacuumCapability::updateCapability (
    double minVacuum,
    double maxVacuum,
    double defaultGripVacuum) [pure virtual]
```

Update the range and default value for the vacuum level supported by the gripper.

Updating the range could result in an already entered value in the configuration of a gripper program node being outside the range. This will make the given program node undefined, and force the end user to adjust the value to be valid again.

If the gripper works with absolute pressure, specify a positive range for the supported vacuum level. If the gripper works with relative pressure (vacuum), specify a negative range for the supported vacuum level.

参数

<i>minVacuum</i>	minimum vacuum level supported by the gripper
<i>maxVacuum</i>	maximum vacuum level supported by the gripper
<i>defaultGripVacuum</i>	default value for the vacuum level for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
<i>unit</i>	the unit for all specified values, not null

异常

<i>InvalidCapabilityRange</i>	if <code>minVacuum > maxVacuum</code> or <code>defaultGripVacuum</code> is outside the range defined by <code>minVacuum</code> and <code>maxVacuum</code> .

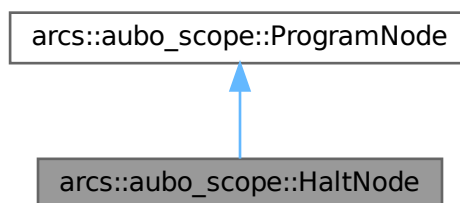
该类的文档由以下文件生成:

- `include/aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h`

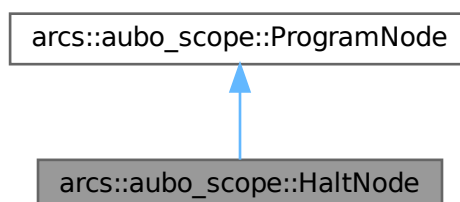
11.61 arcs::aubo_scope::HaltNode 类参考

```
#include <halt_node.h>
```

类 arcs::aubo_scope::HaltNode 继承关系图:



arcs::aubo_scope::HaltNode 的协作图:



Public 成员函数

- [HaltNode](#) (HaltNode &f)
- [HaltNode](#) (HaltNode &&f)
- virtual [~HaltNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [HaltNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数**Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)**

- [ProgramNode](#) ()

11.61.1 详细描述

在文件 [halt_node.h](#) 第 10 行定义.

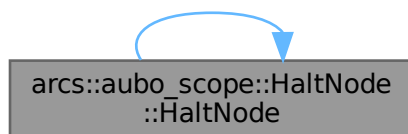
11.61.2 构造及析构函数说明**HaltNode() [1/3]**

```
arcs::aubo_scope::HaltNode::HaltNode (
    HaltNode & f)
```

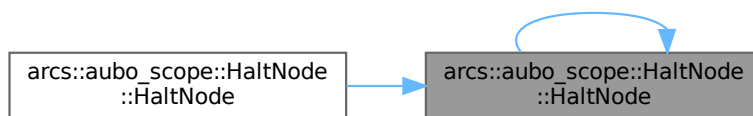
引用了 [HaltNode\(\)](#).

被这些函数引用 [HaltNode\(\)](#) , 以及 [HaltNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

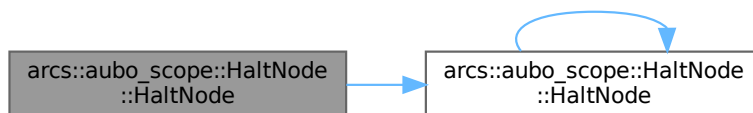


HaltNode() [2/3]

```
arcs::aubo_scope::HaltNode::HaltNode (
    HaltNode && f)
```

引用了 [HaltNode\(\)](#).

函数调用图:



~HaltNode()

```
virtual arcs::aubo_scope::HaltNode::~~HaltNode () [virtual]
```

HaltNode() [3/3]

```
arcs::aubo_scope::HaltNode::HaltNode () [private]
```

11.61.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [halt_node.h](#) 第 18 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.61.4 类成员变量说明

d_

```
void* arcs::aubo_scope::HaltNode::d_ { nullptr } [private]
```

在文件 [halt_node.h](#) 第 20 行定义.

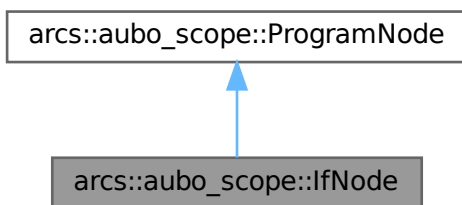
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/halt_node.h](#)

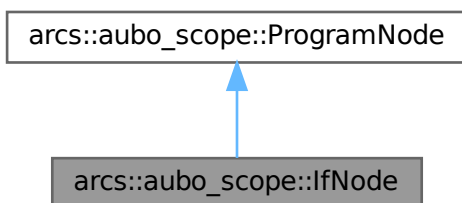
11.62 arcs::aubo_scope::IfNode 类参考

```
#include <if_node.h>
```

类 arcs::aubo_scope::IfNode 继承关系图:



arcs::aubo_scope::IfNode 的协作图:



Public 成员函数

- [IfNode](#) (IfNode &f)
- [IfNode](#) (IfNode &&f)
- virtual [~IfNode](#) ()
- void [setExpression](#) (ExpressionPtr expression)
设置节点使用的表达式。
- ExpressionPtr [getExpression](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [IfNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.62.1 详细描述

在文件 [if_node.h](#) 第 11 行定义.

11.62.2 构造及析构函数说明

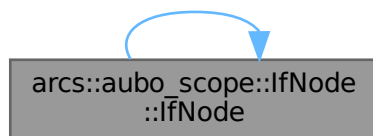
[IfNode](#)() [1/3]

```
arcs::aubo_scope::IfNode::IfNode (
    IfNode & f)
```

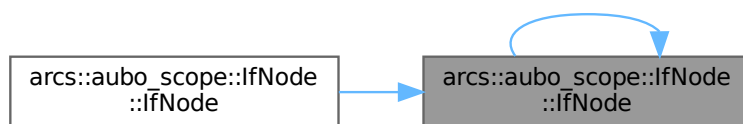
引用了 [IfNode](#)().

被这些函数引用 [IfNode](#)() , 以及 [IfNode](#)().

函数调用图:



这是这个函数的调用关系图:

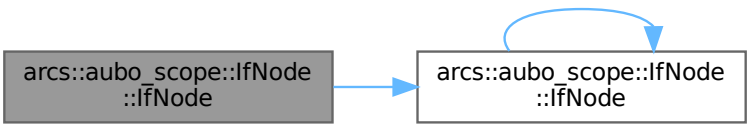


[IfNode](#)() [2/3]

```
arcs::aubo_scope::IfNode::IfNode (
    IfNode && f)
```

引用了 [IfNode](#)().

函数调用图:



~IfNode()
`virtual arcs::aubo_scope::IfNode::~IfNode () [virtual]`

IfNode() [3/3]
`arcs::aubo_scope::IfNode::IfNode () [private]`

11.62.3 成员函数说明

getExpression()
`ExpressionPtr arcs::aubo_scope::IfNode::getExpression ()`

返回
在此 If 节点中求值的表达式

setExpression()
`void arcs::aubo_scope::IfNode::setExpression (`
`ExpressionPtr expression)`
设置节点使用的表达式。

参数

<i>expression</i>	将在 If 节点中求值的表达式

异常

<i>IllegalStateException</i>	如果在 UndoableChanges 范围外调用

11.62.4 友元及相关符号说明

DataSwitch
`friend class DataSwitch [friend]`
在文件 [if_node.h](#) 第 44 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.62.5 类成员变量说明

d_

```
void* arcs::aubo_scope::IfNode::d_ { nullptr } [private]
```

在文件 [if_node.h](#) 第 46 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/if_node.h](#)

11.63 arcs::aubo_scope::InputIoFilter 结构体参考

```
#include <digital_io.h>
```

Public 成员函数

- [bool operator\(\)](#) (IoPtr io)

11.63.1 详细描述

在文件 [digital_io.h](#) 第 80 行定义.

11.63.2 成员函数说明

[operator\(\)](#)()

```
bool arcs::aubo_scope::InputIoFilter::operator() (
    IoPtr io) [inline]
```

在文件 [digital_io.h](#) 第 82 行定义.

该结构体的文档由以下文件生成:

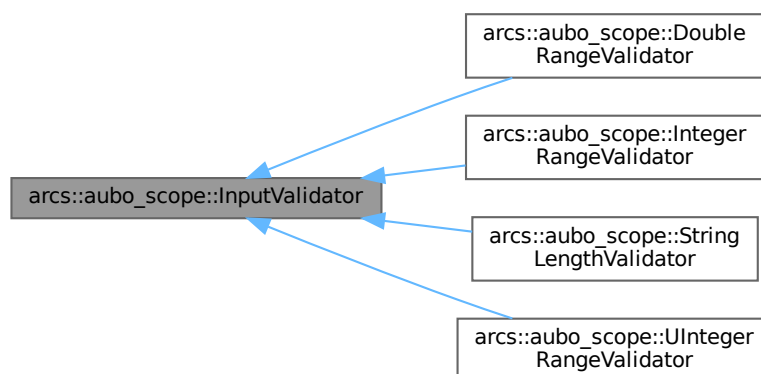
- [include/aubo_caps/domain/io/digital_io.h](#)

11.64 arcs::aubo_scope::InputValidator 结构体参考

输入验证器表示由InputValidationFactory 创建的输入验证器接口。该工厂提供一组标准输入验证器，可用于验证用户使用虚拟键盘/小键盘输入的内容。

```
#include <input_validator.h>
```

类 arcs::aubo_scope::InputValidator 继承关系图:



Public 成员函数

- virtual bool `isValid` (std::string value)=0
验证输入值是否有效。
- virtual std::string `getMessage` (std::string value)=0
当值无效时返回有意义的提示信息。

11.64.1 详细描述

输入验证器表示由 `InputValidationFactory` 创建的输入验证器接口。该工厂提供一组标准输入验证器，可用于验证用户使用虚拟键盘/小键盘输入的内容。

如果 `InputValidationFactory` 中可用的标准验证器不能满足您的需求，您可以自由实现自己的自定义验证器。

参数

<code><T></code>	表示正在验证的输入数据类型的接口（通用）类型参数（例如 <code>Integer</code> 或 <code>Double</code> ）。

在文件 `input_validator.h` 第 41 行定义。

11.64.2 成员函数说明

`getMessage()`

```
virtual std::string arcs::aubo_scope::InputValidator::getMessage (
    std::string value) [pure virtual]
```

当值无效时返回有意义的提示信息。

参数

<i>value</i>	无效的值，如果合理可包含在消息中

返回

提示消息

在 `arcs::aubo_scope::DoubleRangeValidator`, `arcs::aubo_scope::IntegerRangeValidator`, `arcs::aubo_scope::StringLeng`, 以及 `arcs::aubo_scope::UIntegerRangeValidator` 内被实现。

`isValid()`

```
virtual bool arcs::aubo_scope::InputValidator::isValid (
    std::string value) [pure virtual]
```

验证输入值是否有效。

参数

<i>value</i>	需要验证的值

返回

如果值有效则返回 `true`

在 [arcs::aubo_scope::DoubleRangeValidator](#), [arcs::aubo_scope::IntegerRangeValidator](#), [arcs::aubo_scope::StringLengthValidator](#), 以及 [arcs::aubo_scope::UIntegerRangeValidator](#) 内被实现.

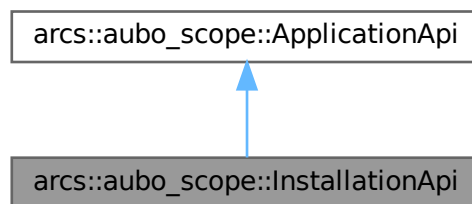
该结构体的文档由以下文件生成:

- `include/aubo_caps/domain/user_interface/input_validator.h`

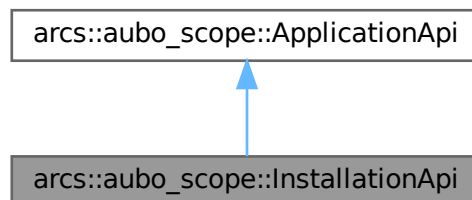
11.65 arcs::aubo_scope::InstallationApi 类参考

```
#include <installation_api.h>
```

类 `arcs::aubo_scope::InstallationApi` 继承关系图:



`arcs::aubo_scope::InstallationApi` 的协作图:



Public 成员函数

- [InstallationApi](#) (`InstallationApi &f`)
- [InstallationApi](#) (`InstallationApi &&f`)
- `virtual ~InstallationApi ()`
- `FunctionModelPtr` [getFunctionModel \(\)](#)
获取向 *AuboScope* 添加脚本功能的接口
- `TcpContributionModelPtr` [getTcpContributionModel \(\)](#)
获取向 *AuboScope* 贡献 *TCP* 的接口
- `FeatureContributionModelPtr` [getFeatureContributionModel \(\)](#)
获取向 *AuboScope* 贡献特征的接口
- `PayloadContributionModelPtr` [getPayloadContributionModel \(\)](#)

Public 成员函数继承自 `arcs::aubo_scope::ApplicationApi`

- `ApplicationApi` (`ApplicationApi &f`)
- `ApplicationApi` (`ApplicationApi &&f`)
- `virtual ~ApplicationApi` ()
- `IoModelPtr` `getIoModel` ()
IoModel 获取IO 模型接口, 用于操作机器人的输入和输出。
- `FeatureModelPtr` `getFeatureModel` ()
FeatureModel 获取特征模型接口, 用于操作当前安装中的特征。
- `TcpModelPtr` `getTcpModel` ()
TcpModel 获取TCP 模型接口, 用于访问当前安装中的TCP。
- `PayloadModelPtr` `getPayloadModel` ()
PayloadModel 获取负载模型接口, 用于访问当前 *AuboScope* 安装中的负载。
- `VariableModelPtr` `getVariableModel` ()
VariableModel 获取变量模型接口, 用于操作变量。
- `ValueFactoryPtr` `getValueFactory` ()
ValueFactory 获取值工厂接口, 用于访问能够创建值对象的各种工厂。
- `template<typename T>`
`std::shared_ptr< T >` `getInstallationNode` ()
 此方法用于获取指定的 *InstallationNodeContribution* 实例。
- `InstallationNodeContributionPtr` `getInstallationNode` (`const char *typeidOfClass`)
- `DeviceManagerPtr` `getDeviceManager` ()
DeviceManager 此方法用于获取指定的 *DeviceManager* 实例。
- `aubo_sdk::RobotProxyPtr` `getRobotProxy` ()
 获取机器人代理, 提供与机器人底层SDK 的交互接口。

Private 成员函数

- `InstallationApi` ()

Private 属性

- `void * d_` { `nullptr` }

友元

- `class DataSwitch`

额外继承的成员函数

Protected 成员函数继承自 `arcs::aubo_scope::ApplicationApi`

- `ApplicationApi` ()

11.65.1 详细描述

在文件 `installation_api.h` 第 31 行定义.

11.65.2 构造及析构函数说明

`InstallationApi()` [1/3]

```
arcs::aubo_scope::InstallationApi::InstallationApi (  
    InstallationApi & f)
```

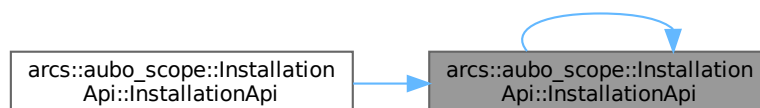
引用了 `InstallationApi()`.

被这些函数引用 `InstallationApi()`, 以及 `InstallationApi()`.

函数调用图:



这是这个函数的调用关系图:

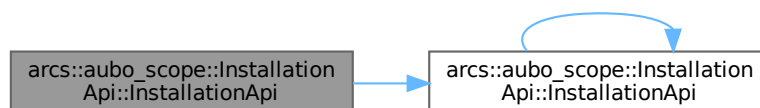


InstallationApi() [2/3]

```
arcs::aubo_scope::InstallationApi::InstallationApi (  
    InstallationApi && f)
```

引用了 [InstallationApi\(\)](#).

函数调用图:



~InstallationApi()

```
virtual arcs::aubo_scope::InstallationApi::~~InstallationApi () [virtual]
```

InstallationApi() [3/3]

```
arcs::aubo_scope::InstallationApi::InstallationApi () [private]
```

11.65.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [installation_api.h](#) 第 98 行定义.
 引用了 [DataSwitch](#).
 被这些函数引用 [DataSwitch](#).

11.65.4 类成员变量说明

`d_`

`void* arcs::aubo_scope::InstallationApi::d_ { nullptr } [private]`
 在文件 [installation_api.h](#) 第 100 行定义.
 该类的文档由以下文件生成:

- [include/aubo_caps/domain/installation/installation_api.h](#)

11.66 arcs::aubo_scope::InstallationApiProvider 类参考

安装 API 提供者提供与安装节点相关的功能和服务访问。这包括 AuboScope 内部已有的功能，以及在 AuboScope 中创建要使用的额外模型元素。

```
#include <installation_api_provider.h>
```

Public 成员函数

- [InstallationApiProvider](#) ([InstallationApiProvider](#) &f)
- [InstallationApiProvider](#) ([InstallationApiProvider](#) &&f)
- [virtual ~InstallationApiProvider](#) ()
- [SystemApiPtr](#) [getSystemApi](#) ()
提供对系统相关功能的访问
- [UserInterfaceApiPtr](#) [getUserInterfaceApi](#) ()
提供对用户界面和终端用户交互相关功能的访问
- [InstallationApiPtr](#) [getInstallationApi](#) ()
提供对安装相关功能的访问

Private 成员函数

- [InstallationApiProvider](#) ()

Private 属性

- `void * d\_ { nullptr }`

友元

- `class DataSwitch`

11.66.1 详细描述

安装 API 提供者提供与安装节点相关的功能和服务访问。这包括 AuboScope 内部已有的功能，以及在 AuboScope 中创建要使用的额外模型元素。

在文件 [installation_api_provider.h](#) 第 26 行定义.

11.66.2 构造及析构函数说明

InstallationApiProvider() [1/3]

```
arcs::aubo_scope::InstallationApiProvider::InstallationApiProvider (  
    InstallationApiProvider & f)
```

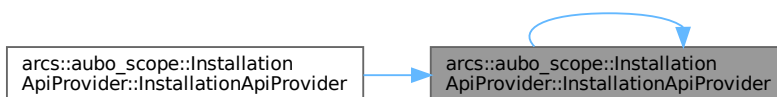
引用了 [InstallationApiProvider\(\)](#).

被这些函数引用 [InstallationApiProvider\(\)](#) , 以及 [InstallationApiProvider\(\)](#).

函数调用图:



这是这个函数的调用关系图:



InstallationApiProvider() [2/3]

```
arcs::aubo_scope::InstallationApiProvider::InstallationApiProvider (
    InstallationApiProvider && f)
```

引用了 [InstallationApiProvider\(\)](#).

函数调用图:



~InstallationApiProvider()

```
virtual arcs::aubo_scope::InstallationApiProvider::~~InstallationApiProvider () [virtual]
```

InstallationApiProvider() [3/3]

```
arcs::aubo_scope::InstallationApiProvider::InstallationApiProvider () [private]
```

11.66.3 成员函数说明

getInstallationApi()

```
InstallationApiPtr arcs::aubo_scope::InstallationApiProvider::getInstallationApi ()
```

提供对安装相关功能的访问

返回

InstallationAPI 实例

getSystemApi()

SystemApiPtr arcs::aubo_scope::InstallationApiProvider::getSystemApi ()
提供对系统相关功能的访问

返回

SystemAPI 实例

getUserInterfaceApi()

UserInterfaceApiPtr arcs::aubo_scope::InstallationApiProvider::getUserInterfaceApi ()
提供对用户界面和终端用户交互相关功能的访问

返回

UserInterfaceAPI 实例

11.66.4 友元及相关符号说明

DataSwitch

friend class DataSwitch [friend]
在文件 [installation_api_provider.h](#) 第 77 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.66.5 类成员变量说明

d_

void* arcs::aubo_scope::InstallationApiProvider::d_ { nullptr } [private]
在文件 [installation_api_provider.h](#) 第 79 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/contribution/installation/installation_api_provider.h](#)

11.67 arcs::aubo_scope::InstallationContributionConfiguration 类参考

```
#include <installation_contribution_configuration.h>
```

Public 成员函数

- [virtual ~InstallationContributionConfiguration \(\)=default](#)

11.67.1 详细描述

在文件 [installation_contribution_configuration.h](#) 第 11 行定义.

11.67.2 构造及析构函数说明

~InstallationContributionConfiguration()

virtual arcs::aubo_scope::InstallationContributionConfiguration::~~InstallationContributionConfiguration ()
[virtual], [default]

该类的文档由以下文件生成:

- [include/aubo_caps/contribution/installation/installation_contribution_configuration.h](#)

11.68 arcs::aubo_scope::InstallationCreationContext 类参考

安装创建上下文提供安装节点创建时所在上下文的相关信息。

```
#include <installation_creation_context.h>
```

Public 类型

- enum [NodeCreationType](#) { [NEW](#) , [LOAD](#) }

Public 成员函数

- [InstallationCreationContext](#) ([InstallationCreationContext](#) &f)
- [InstallationCreationContext](#) ([InstallationCreationContext](#) &&f)
- [~InstallationCreationContext](#) ()
- [NodeCreationType](#) [getNodeCreationType](#) ()
- std::string [getCreatedVersion](#) ()
- std::string [getLastSavedVersion](#) ()

Private 成员函数

- [InstallationCreationContext](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.68.1 详细描述

安装创建上下文提供安装节点创建时所在上下文的相关信息。

在文件 [installation_creation_context.h](#) 第 22 行定义.

11.68.2 成员枚举类型说明

NodeCreationType

```
enum arcs::aubo_scope::InstallationCreationContext::NodeCreationType
```

枚举值

NEW	
LOAD	

在文件 [installation_creation_context.h](#) 第 25 行定义.

11.68.3 构造及析构函数说明

InstallationCreationContext() [1/3]

```
arcs::aubo_scope::InstallationCreationContext::InstallationCreationContext (  
    InstallationCreationContext & f)
```

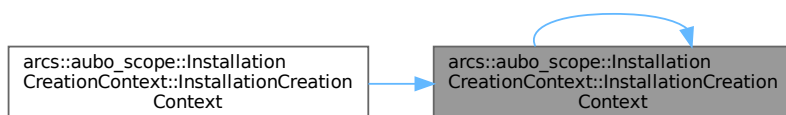
引用了 [InstallationCreationContext\(\)](#).

被这些函数引用 [InstallationCreationContext\(\)](#) , 以及 [InstallationCreationContext\(\)](#).

函数调用图:



这是这个函数的调用关系图:

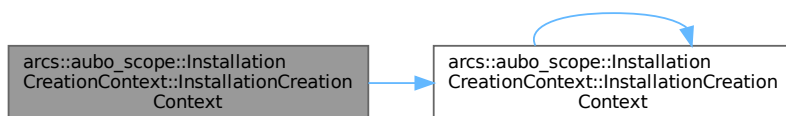


InstallationCreationContext() [2/3]

```
arcs::aubo_scope::InstallationCreationContext::InstallationCreationContext (
    InstallationCreationContext && f)
```

引用了 [InstallationCreationContext\(\)](#).

函数调用图:



~InstallationCreationContext()

```
arcs::aubo_scope::InstallationCreationContext::~InstallationCreationContext ()
```

InstallationCreationContext() [3/3]

```
arcs::aubo_scope::InstallationCreationContext::InstallationCreationContext () [private]
```

11.68.4 成员函数说明

getCreatedVersion()

```
std::string arcs::aubo_scope::InstallationCreationContext::getCreatedVersion ()
```

getLastSavedVersion()

```
std::string arcs::aubo_scope::InstallationCreationContext::getLastSavedVersion ()
```

getNodeCreationType()

```
NodeCreationType arcs::aubo_scope::InstallationCreationContext::getNodeCreationType ()
```

11.68.5 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
在文件 installation_creation_context.h 第 41 行定义.
引用了 DataSwitch.
被这些函数引用 DataSwitch.
```

11.68.6 类成员变量说明

d_

```
void* arcs::aubo_scope::InstallationCreationContext::d_ { nullptr } [private]
在文件 installation_creation_context.h 第 43 行定义.
该类的文档由以下文件生成:
```

- include/aubo_caps/contribution/installation/installation_creation_context.h

11.69 arcs::aubo_scope::InstallationNodeContribution 类参考

安装节点贡献安装节点贡献的 API。

```
#include <installation_node_contribution.h>
```

Public 成员函数

- virtual [~InstallationNodeContribution](#) ()=default
- virtual void [openView](#) ()=0
当终端用户每次在安装标签页中打开此 *AuboCap* 贡献时调用。
- virtual void [closeView](#) ()=0
当终端用户每次在安装标签页中退出此 *AuboCap* 贡献时调用。
- virtual void [generateScript](#) (ScriptWriterPtr script_writer)=0
定义添加到程序启动时执行的脚本代码开头的脚本代码。

11.69.1 详细描述

安装节点贡献安装节点贡献的 API。

在文件 [installation_node_contribution.h](#) 第 20 行定义。

11.69.2 构造及析构函数说明

~InstallationNodeContribution()

```
virtual arcs::aubo_scope::InstallationNodeContribution::~~InstallationNodeContribution () [virtual], [default]
```

11.69.3 成员函数说明

closeView()

```
virtual void arcs::aubo_scope::InstallationNodeContribution::closeView () [pure virtual]
当终端用户每次在安装标签页中退出此 AuboCap 贡献时调用。
```

generateScript()

```
virtual void arcs::aubo_scope::InstallationNodeContribution::generateScript (
    ScriptWriterPtr script_writer) [pure virtual]
```

定义添加到程序启动时执行的脚本代码开头的脚本代码。

参数

<i>script_writer</i>	用于向程序脚本代码前言添加脚本命令

openView()

```
virtual void arcs::aubo_scope::InstallationNodeContribution::openView () [pure virtual]
```

当终端用户每次在安装标签页中打开此 AuboCap 贡献时调用。

该类的文档由以下文件生成:

- include/aubo_caps/contribution/installation/[installation_node_contribution.h](#)

11.70 arcs::aubo_scope::InstallationNodeService 类参考

安装节点服务定义了可在 AuboScope 中定义和添加安装节点及对应界面（用户界面基于 Qt）所需的 API。

```
#include <installation_node_service.h>
```

Public 成员函数

- virtual [~InstallationNodeService](#) ()=default
- virtual std::string [getTitle](#) ()=0
- virtual std::string [getIcon](#) ()=0
- virtual void [configureContribution](#) (InstallationContributionConfigurationPtr configuration)=0
此方法在服务注册后调用一次。修改 *configuration* 参数来配置您的贡献。配置对象将已经具有匹配大多数用例的默认属性值。
- virtual InstallationNodeViewPtr [createView](#) (ViewApiProviderPtr api_provider)=0
创建一个新的视图实例，实现安装节点画面的 UI。在加载或创建新安装时调用一次。
- virtual InstallationNodeContributionPtr [createInstallationNode](#) (InstallationApiProviderPtr api_provider, InstallationNodeViewPtr view, DataModelPtr model, InstallationCreationContextPtr context)=0

11.70.1 详细描述

安装节点服务定义了可在 AuboScope 中定义和添加安装节点及对应界面（用户界面基于 Qt）所需的 API。

在文件 [installation_node_service.h](#) 第 28 行定义。

11.70.2 构造及析构函数说明**~InstallationNodeService()**

```
virtual arcs::aubo_scope::InstallationNodeService::~~InstallationNodeService () [virtual], [default]
```

11.70.3 成员函数说明**configureContribution()**

```
virtual void arcs::aubo_scope::InstallationNodeService::configureContribution (
    InstallationContributionConfigurationPtr configuration) [pure virtual]
```

此方法在服务注册后调用一次。修改 `configuration` 参数来配置您的贡献。配置对象将已经具有匹配大多数用例的默认属性值。

`ContributionConfiguration` 对象的值将在方法调用后立即读取一次。稍后更改值将不会产生任何效果，因此不要存储对配置对象的引用。

如果默认值合适，请将此方法留空。

参数

<i>configuration</i>	具有默认值的可修改贡献配置

createInstallationNode()

```
virtual InstallationNodeContributionPtr arcs::aubo_scope::InstallationNodeService::createInstallationNode (
    InstallationApiProviderPtr api_provider,
    InstallationNodeViewPtr view,
    DataModelPtr model,
    InstallationCreationContextPtr context) [pure virtual]
```

创建一个新的安装节点实例。

返回的节点必须使用提供的数据模型对象来检索和存储其中包含的数据。加载现有安装时，将忽略安装节点构造函数对提供的数据模型所做的所有修改。数据模型对象在同一 AuboCap 贡献的所有安装节点之间共享。

参数

<i>api_provider</i>	提供对 AuboScope 中与安装节点相关的功能和服务访问
<i>view</i>	由 <code>createView</code> 创建的视图
<i>model</i>	新安装节点实例的所有配置数据存储和检索的数据模型对象
<i>context</i>	创建此节点的上下文

返回

新构建的安装节点贡献实例

createView()

```
virtual InstallationNodeViewPtr arcs::aubo_scope::InstallationNodeService::createView (
    ViewApiProviderPtr api_provider) [pure virtual]
```

创建一个新的视图实例，实现安装节点画面的 UI。在加载或创建新安装时调用一次。

参数

<i>api_provider</i>	提供对 AuboScope 中与用户界面和终端用户交互相关的功能和服务访问

返回

视图

getIcon()

```
virtual std::string arcs::aubo_scope::InstallationNodeService::getIcon () [pure virtual]
```

返回

此安装贡献显示的图标。在启动时调用一次。

getTitle()

```
virtual std::string arcs::aubo_scope::InstallationNodeService::getTitle () [pure virtual]
```

参数

<i>locale</i>	AuboScope 的当前语言环境，可用于支持多种语言的标题

返回

在安装标签页左侧导航中显示的文本以及对应安装节点画面的标题。在启动时调用一次。

该类的文档由以下文件生成:

- include/aubo_caps/contribution/installation/installation_node_service.h

11.71 arcs::aubo_scope::InstallationNodeView 类参考

安装节点视图安装节点视图的 API，定义了对应 [InstallationNodeContribution](#) 的 UI。

```
#include <installation_node_view.h>
```

Public 成员函数

- virtual [~InstallationNodeView](#) ()=default
- virtual void [buildUi](#) ([WidgetHandle](#) panel, [InstallationNodeContributionPtr](#) contribution)=0
在提供的面板上为对应的 [InstallationNodeContribution](#) 构建 UI。使用贡献进行逻辑和数据模型操作。

11.71.1 详细描述

安装节点视图安装节点视图的 API，定义了对应 [InstallationNodeContribution](#) 的 UI。

参数

<code><C></code>	表示此视图实现 UI 的 InstallationNodeContribution 类型的（通用）类型参数

在文件 [installation_node_view.h](#) 第 32 行定义。

11.71.2 构造及析构函数说明**~InstallationNodeView()**

```
virtual arcs::aubo_scope::InstallationNodeView::~~InstallationNodeView () [virtual], [default]
```

11.71.3 成员函数说明**buildUi()**

```
virtual void arcs::aubo_scope::InstallationNodeView::buildUi (  
    WidgetHandle panel,  
    InstallationNodeContributionPtr contribution) [pure virtual]
```

在提供的面板上为对应的 [InstallationNodeContribution](#) 构建 UI。使用贡献进行逻辑和数据模型操作。

参数

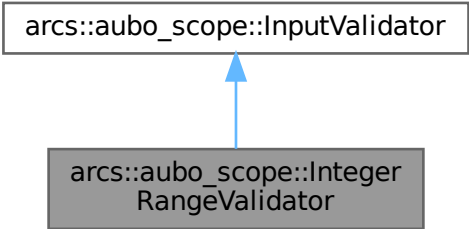
<i>panel</i>	构建 UI 的面板
<i>contribution</i>	对应的安装节点贡献

该类的文档由以下文件生成:

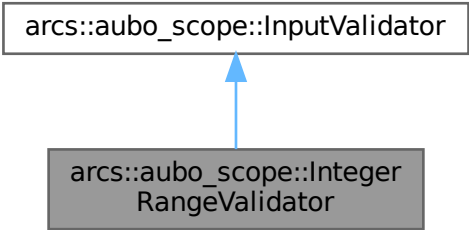
- include/aubo_caps/contribution/installation/[installation_node_view.h](#)

11.72 arcs::aubo_scope::IntegerRangeValidator 结构体参考

```
#include <input_validator.h>
类 arcs::aubo_scope::IntegerRangeValidator 继承关系图:
```



arcs::aubo_scope::IntegerRangeValidator 的协作图:



Public 成员函数

- [IntegerRangeValidator](#) (int min_value, int max_value)
- [IntegerRangeValidator](#) (std::function< std::pair< int, int >()> range_func)
- bool [isValid](#) (std::string value) override
验证输入值是否有效。
- std::string [getMessage](#) (std::string value) override
当值无效时返回有意义的提示信息。

Private 成员函数

- void `updateValue()`

Private 属性

- int `min_value_` { INT_MIN }
- int `max_value_` { INT_MAX }
- std::function< std::pair< int, int >()> `range_func_` { nullptr }

11.72.1 详细描述

在文件 `input_validator.h` 第 77 行定义.

11.72.2 构造及析构函数说明**IntegerRangeValidator()** [1/2]

```
arcs::aubo_scope::IntegerRangeValidator::IntegerRangeValidator (
    int min_value,
    int max_value)
```

IntegerRangeValidator() [2/2]

```
arcs::aubo_scope::IntegerRangeValidator::IntegerRangeValidator (
    std::function< std::pair< int, int >()> range_func)
```

11.72.3 成员函数说明**getMessage()**

```
std::string arcs::aubo_scope::IntegerRangeValidator::getMessage (
    std::string value) [override], [virtual]
```

当值无效时返回有意义的提示信息。

参数

<i>value</i>	无效的值，如果合理可包含在消息中

返回

提示信息

实现了 `arcs::aubo_scope::InputValidator`.

isValid()

```
bool arcs::aubo_scope::IntegerRangeValidator::isValid (
    std::string value) [override], [virtual]
```

验证输入值是否有效。

参数

<i>value</i>	需要验证的值

返回

如果值有效则返回 `true`

实现了 `arcs::aubo_scope::InputValidator`.

updateValue()

```
void arcs::aubo_scope::IntegerRangeValidator::updateValue () [private]
```

11.72.4 类成员变量说明

max_value__

```
int arcs::aubo_scope::IntegerRangeValidator::max_value_ { INT_MAX } [private]
```

在文件 [input_validator.h](#) 第 92 行定义.

min_value__

```
int arcs::aubo_scope::IntegerRangeValidator::min_value_ { INT_MIN } [private]
```

在文件 [input_validator.h](#) 第 91 行定义.

range_func__

```
std::function<std::pair<int, int>()> arcs::aubo_scope::IntegerRangeValidator::range_func_ { nullptr } [private]
```

在文件 [input_validator.h](#) 第 94 行定义.

该结构体的文档由以下文件生成:

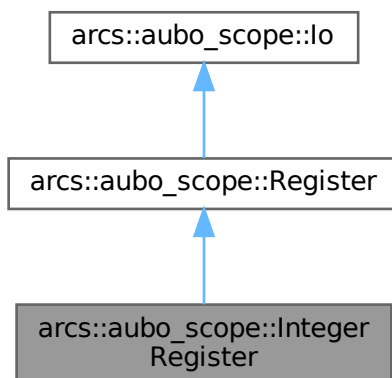
- [include/aubo_caps/domain/user_interface/input_validator.h](#)

11.73 arcs::aubo_scope::IntegerRegister 类参考

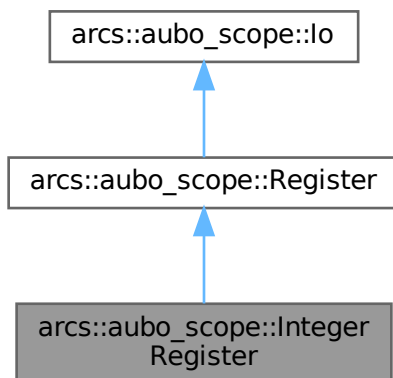
This interface provides support for integer registers.

```
#include <integer_register.h>
```

类 arcs::aubo_scope::IntegerRegister 继承关系图:



arcs::aubo_scope::IntegerRegister 的协作图:



Public 成员函数

- virtual `~IntegerRegister ()`=default

Public 成员函数继承自 `arcs::aubo_scope::Register`

- virtual `~Register ()`=default

Public 成员函数继承自 `arcs::aubo_scope::Io`

- `Io (Io &f)`
- `Io (Io &&f)`
- virtual `~Io ()`
- `std::string getDisplayName () const`
获取当前安装中I/O的用户自定义名称
- `std::string getDefaultName () const`
获取I/O的机器人默认名称
- `std::string getValueStr () const`
获取I/O的当前读数值（字符串形式）
- `IoType getType () const`
获取I/O类型
- `InterfaceType getInterfaceType () const`
获取此I/O所属的接口类型
- `int getAction () const`
获取I/O的动作编号
- `bool isInput () const`
判断I/O是否支持读取值
- `bool isResolvable () const`
某些I/O（如MODBUS I/O和模拟工具输入）在某些情况下可能不可用。此方法用于判断I/O是否可用。
- `size_t hashCode ()`

额外继承的成员函数

Public 类型继承自 `arcs::aubo_scope::Io`

- enum `IoType` : int {
 DIGITAL , ANALOG , INTEGER , BOOLEAN ,
 DOUBLE }
- enum `InterfaceType` : int {
 STANDARD , TOOL , CONFIGURABLE , LINKIO ,
 MODBUS , GENERAL_PURPOSE , PROFINET , ETHERNET_IP }

Protected 成员函数继承自 `arcs::aubo_scope::Io`

- `Io ()`

11.73.1 详细描述

This interface provides support for integer registers.
在文件 `integer_register.h` 第 14 行定义.

11.73.2 构造及析构函数说明

`~IntegerRegister()`

```
virtual arcs::aubo_scope::IntegerRegister::~~IntegerRegister () [virtual], [default]
```

该类的文档由以下文件生成:

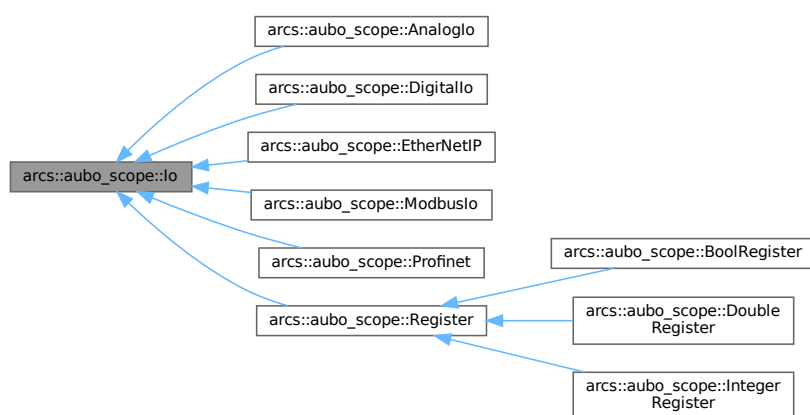
- `include/aubo_caps/domain/io/integer_register.h`

11.74 `arcs::aubo_scope::Io` 类参考

IO 基础接口这是表示 AuboScope 中所有可用 I/O 类型的基础接口。

```
#include <io.h>
```

类 `arcs::aubo_scope::Io` 继承关系图:



Public 类型

- enum `IoType` : int {
 DIGITAL , ANALOG , INTEGER , BOOLEAN ,
 DOUBLE }

- enum `InterfaceType` : int {
`STANDARD` , `TOOL` , `CONFIGURABLE` , `LINKIO` ,
`MODBUS` , `GENERAL_PURPOSE` , `PROFINET` , `ETHERNET_IP` }

Public 成员函数

- `Io` (Io &f)
- `Io` (Io &&f)
- virtual `~Io` ()
- std::string `getDisplayname` () const
 获取当前安装中 I/O 的用户自定义名称
- std::string `getDefaultName` () const
 获取 I/O 的机器人默认名称
- std::string `getValueStr` () const
 获取 I/O 的当前读数值 (字符串形式)
- `IoType` `getType` () const
 获取 I/O 类型
- `InterfaceType` `getInterfaceType` () const
 获取此 I/O 所属的接口类型
- int `getAction` () const
 获取 I/O 的动作编号
- bool `isInput` () const
 判断 I/O 是否支持读取值
- bool `isResolvable` () const
 某些 I/O (如 MODBUS I/O 和模拟工具输入) 在某些情况下可能不可用。此方法用于判断 I/O 是否可用。
- size_t `hashCode` ()

Protected 成员函数

- `Io` ()

Private 属性

- void * `d_` { nullptr }

友元

- class `DataSwitch`

11.74.1 详细描述

IO 基础接口这是表示 AuboScope 中所有可用 I/O 类型的基础接口。
 在文件 `io.h` 第 22 行定义。

11.74.2 成员枚举类型说明

InterfaceType

```
enum arcs::aubo_scope::Io::InterfaceType : int
```

枚举值

STANDARD	
TOOL	
CONFIGURABLE	

LINKIO	
MODBUS	
GENERAL_PURPOSE	
PROFINET	
ETHERNET_IP	

在文件 [io.h](#) 第 33 行定义.

IoType

```
enum arcs::aubo_scope::Io::IoType : int
```

枚举值

DIGITAL	
ANALOG	
INTEGER	
BOOLEAN	
DOUBLE	

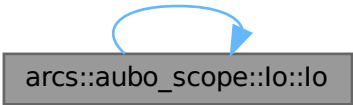
在文件 [io.h](#) 第 25 行定义.

11.74.3 构造及析构函数说明

Io() [1/3]

```
arcs::aubo_scope::Io::Io (  
    Io & f)
```

引用了 [Io\(\)](#).
被这些函数引用 [Io\(\)](#) , 以及 [Io\(\)](#).
函数调用图:



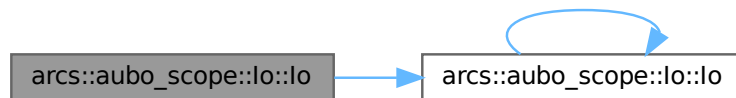
这是这个函数的调用关系图:



Io() [2/3]

```
arcs::aubo_scope::Io::Io (
    Io && f)
```

引用了 [Io\(\)](#).
函数调用图:



~Io()

```
virtual arcs::aubo_scope::Io::~~Io () [virtual]
```

Io() [3/3]

```
arcs::aubo_scope::Io::Io () [protected]
```

11.74.4 成员函数说明

getAction()

```
int arcs::aubo_scope::Io::getAction () const
```

获取I/O 的动作编号

返回

I/O 的动作编号。

getDefaultName()

```
std::string arcs::aubo_scope::Io::getDefaultName () const
```

获取I/O 的机器人默认名称

返回

I/O 的默认名称。

getDisplayName()

```
std::string arcs::aubo_scope::Io::getDisplayName () const
```

获取当前安装中I/O 的用户自定义名称

返回

I/O 的显示名称。

getInterfaceType()

```
InterfaceType arcs::aubo_scope::Io::getInterfaceType () const
```

获取此I/O 所属的接口类型

返回

接口类型枚举。

getType()

```
IoType arcs::aubo_scope::Io::getType () const
```

获取I/O 类型

返回

I/O 类型枚举。

getValueStr()

```
std::string arcs::aubo_scope::Io::getValueStr () const
```

获取I/O 的当前读数值（字符串形式）

返回

I/O 的当前读数。

hashCode()

```
size_t arcs::aubo_scope::Io::hashCode ()
```

isInput()

```
bool arcs::aubo_scope::Io::isInput () const
```

判断I/O 是否支持读取值

返回

如果I/O 支持读取返回 `true`, 否则返回 `false`。

isResolvable()

```
bool arcs::aubo_scope::Io::isResolvable () const
```

某些I/O（如MODBUS I/O 和模拟工具输入）在某些情况下可能不可用。此方法用于判断 I/O 是否可用。

返回

如果I/O 可解析返回 `true`, 否则返回 `false`。

11.74.5 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [io.h](#) 第 158 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.74.6 类成员变量说明

d_

```
void* arcs::aubo_scope::Io::d_ { nullptr } [private]
```

在文件 [io.h](#) 第 159 行定义.

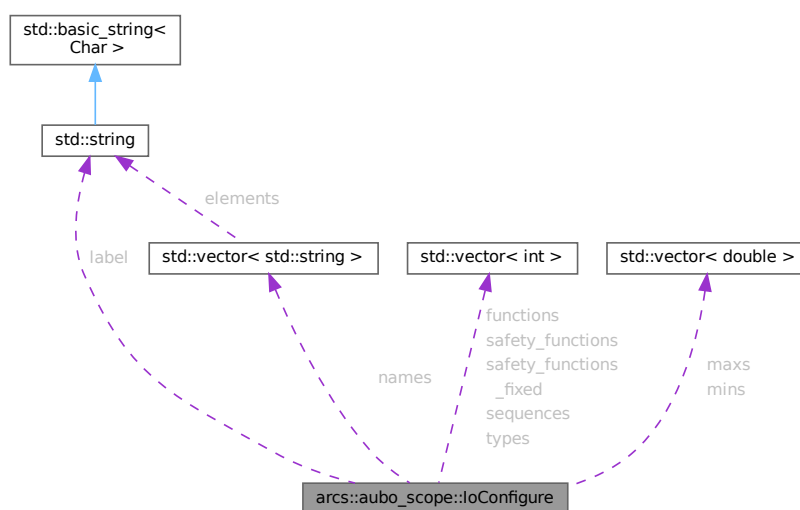
该类的文档由以下文件生成:

- [include/aubo_caps/domain/io/io.h](#)

11.75 arcs::aubo_scope::IoConfigure 结构体参考

```
#include <system_api.h>
```

arcs::aubo_scope::IoConfigure 的协作图:



Public 属性

- [std::string label](#)
- [int num](#)
- [std::vector< std::string > names](#)
- [std::vector< int > sequences](#)
- [std::vector< int > functions](#)
- [std::vector< int > safety_functions](#)
- [std::vector< int > safety_functions_fixed](#)
- [bool function_lock { false }](#)
- [std::vector< int > types](#)
- [std::vector< double > mins](#)
- [std::vector< double > maxs](#)

11.75.1 详细描述

在文件 [system_api.h](#) 第 18 行定义.

11.75.2 类成员变量说明

function_lock

```
bool arcs::aubo_scope::IoConfigure::function_lock { false }
```

在文件 [system_api.h](#) 第 27 行定义.

functions

```
std::vector<int> arcs::aubo_scope::IoConfigure::functions
```

在文件 [system_api.h](#) 第 24 行定义.

label

```
std::string arcs::aubo_scope::IoConfigure::label
```

在文件 [system_api.h](#) 第 20 行定义.

maxs

```
std::vector<double> arcs::aubo_scope::IoConfigure::maxs
```

在文件 [system_api.h](#) 第 30 行定义.

mins

```
std::vector<double> arcs::aubo_scope::IoConfigure::mins
```

在文件 [system_api.h](#) 第 29 行定义.

names

```
std::vector<std::string> arcs::aubo_scope::IoConfigure::names
```

在文件 [system_api.h](#) 第 22 行定义.

num

```
int arcs::aubo_scope::IoConfigure::num
```

在文件 [system_api.h](#) 第 21 行定义.

safety_functions

```
std::vector<int> arcs::aubo_scope::IoConfigure::safety_functions
```

在文件 [system_api.h](#) 第 25 行定义.

safety_functions_fixed

```
std::vector<int> arcs::aubo_scope::IoConfigure::safety_functions_fixed
```

在文件 [system_api.h](#) 第 26 行定义.

sequences

```
std::vector<int> arcs::aubo_scope::IoConfigure::sequences
```

在文件 [system_api.h](#) 第 23 行定义.

types

```
std::vector<int> arcs::aubo_scope::IoConfigure::types
```

在文件 [system_api.h](#) 第 28 行定义.
该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/system_api.h](#)

11.76 arcs::aubo_scope::IoModel 类参考

IO 模型提供从当前机器人安装中获取 I/O 的方法。

```
#include <io_model.h>
```

Public 成员函数

- [IoModel](#) (IoModel &f)
- [IoModel](#) (IoModel &&f)
- [~IoModel](#) ()
- std::vector< IoPtr > [getAll](#) ()
获取所有可用的 I/O 集合
- IoPtr [getIo](#) (const std::string &name)
[Io](#) 根据名称获取 I/O
- std::vector< IoPtr > [getIos](#) (std::function< bool(IoPtr)> filter)
使用过滤器获取 I/O 子集

Private 成员函数

- [IoModel](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.76.1 详细描述

IO 模型提供从当前机器人安装中获取 I/O 的方法。
在文件 [io_model.h](#) 第 24 行定义。

11.76.2 构造及析构函数说明

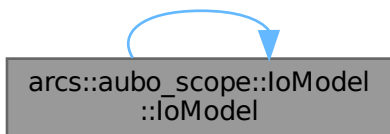
IoModel() [1/3]

```
arcs::aubo_scope::IoModel::IoModel (  
    IoModel & f)
```

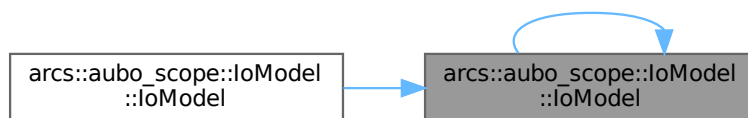
引用了 [IoModel\(\)](#).

被这些函数引用 [IoModel\(\)](#) , 以及 [IoModel\(\)](#).

函数调用图:



这是这个函数的调用关系图:

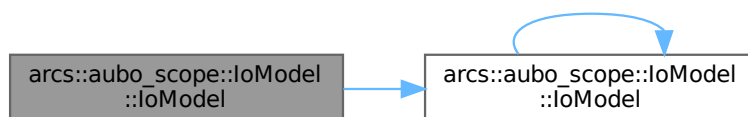


IoModel() [2/3]

```
arcs::aubo_scope::IoModel::IoModel (
    IoModel && f)
```

引用了 [IoModel\(\)](#).

函数调用图:



~IoModel()

```
arcs::aubo_scope::IoModel::~~IoModel ()
```

IoModel() [3/3]

```
arcs::aubo_scope::IoModel::IoModel () [private]
```

11.76.3 成员函数说明

getAll()

```
std::vector< IoPtr > arcs::aubo_scope::IoModel::getAll ()
```

获取所有可用的I/O 集合

返回

可用I/O 的集合。

getIo()

```
IoPtr arcs::aubo_scope::IoModel::getIo (
    const std::string & name)
```

[Io](#) 根据名称获取I/O

参数

<i>name</i>	I/O 名称。

返回

对应名称的I/O。

getIos()

```
std::vector< IntPtr > arcs::aubo_scope::IoModel::getIos (
    std::function< bool(IntPtr)> filter)
```

使用过滤器获取I/O 子集

参数

<i>filter</i>	过滤器。

返回

被过滤器接受的I/O 集合。

11.76.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [io_model.h](#) 第 77 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.76.5 类成员变量说明**d_**

```
void* arcs::aubo_scope::IoModel::d_ { nullptr } [private]
```

在文件 [io_model.h](#) 第 79 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/io/io_model.h](#)

11.77 arcs::aubo_scope::KeyboardManager 类参考

```
#include <keyboard_manager.h>
```

Public 成员函数

- [KeyboardManager](#) ([KeyboardManager](#) &f)
- [KeyboardManager](#) ([KeyboardManager](#) &&f)
- virtual [~KeyboardManager](#) ()
- ARCS_DEPRECATED void [registerStandardKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)

registerStandardKeyboard 注册标准键盘

- void [registerStandardKeyboard](#) (QWidget *wd, std::function< ExpressionPtr(void)> init_cb, std::function< void(ExpressionPtr)> result_cb, const std::shared_ptr< [InputValidator](#) > &validator=nullptr)

registerStandardKeyboard

- ARCS_DEPRECATED void [registerStringKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)

注册字符串键盘

- void [registerStringKeyboard](#) (QWidget *wd, std::function< ExpressionPtr(void)> init_cb, std::function< void(ExpressionPtr)> result_cb, const std::shared_ptr< [InputValidator](#) > &validator=nullptr)
- ARCS_DEPRECATED void [registerPasswordKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)

注册密码键盘

- void [registerPasswordKeyboard](#) (QWidget *wd, std::function< ExpressionPtr(void)> init_cb, std::function< void(ExpressionPtr)> result_cb, const std::shared_ptr< [InputValidator](#) > &validator=nullptr)
- ARCS_DEPRECATED void [registerCodeEditorKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)

注册多行代码编辑键盘

- void [registerCodeEditorKeyboard](#) (QWidget *wd, std::function< ExpressionPtr(void)> init_cb, std::function< void(ExpressionPtr)> result_cb, const std::shared_ptr< [InputValidator](#) > &validator=nullptr)
- void [registerNumberKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerDoubleKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerPositiveDoubleKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerIntegerKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerPositiveIntegerKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerIPAddressKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerExpressionKeyboard](#) (QWidget *wd, std::function< ExpressionPtr(void)> init_cb=nullptr, std::function< void(ExpressionPtr result_value)> result_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr)

Private 成员函数

- [KeyboardManager](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.77.1 详细描述

在文件 `keyboard_manager.h` 第 18 行定义.

11.77.2 构造及析构函数说明

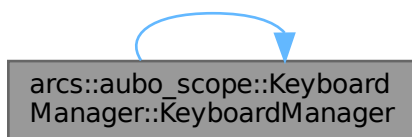
KeyboardManager() [1/3]

```
arcs::aubo_scope::KeyboardManager::KeyboardManager (  
    KeyboardManager & f)
```

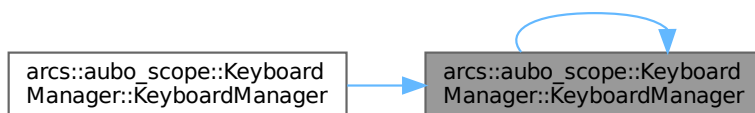
引用了 `KeyboardManager()`.

被这些函数引用 `KeyboardManager()` , 以及 `KeyboardManager()`.

函数调用图:



这是这个函数的调用关系图:

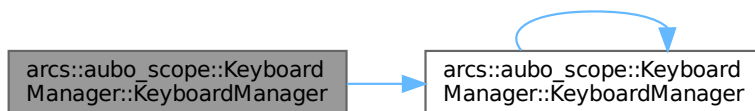


KeyboardManager() [2/3]

```
arcs::aubo_scope::KeyboardManager::KeyboardManager (  
    KeyboardManager && f)
```

引用了 `KeyboardManager()`.

函数调用图:



~KeyboardManager()

```
virtual arcs::aubo_scope::KeyboardManager::~~KeyboardManager () [virtual]
```

KeyboardManager() [3/3]

```
arcs::aubo_scope::KeyboardManager::KeyboardManager () [private]
```

11.77.3 成员函数说明**registerCodeEditorKeyboard()** [1/2]

```
void arcs::aubo_scope::KeyboardManager::registerCodeEditorKeyboard (
    QWidget * wd,
    std::function< ExpressionPtr(void)> init_cb,
    std::function< void(ExpressionPtr)> result_cb,
    const std::shared_ptr< InputValidator > & validator = nullptr)
```

registerCodeEditorKeyboard() [2/2]

```
ARCS_DEPRECATED void arcs::aubo_scope::KeyboardManager::registerCodeEditorKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

注册多行代码编辑键盘

registerDoubleKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerDoubleKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

registerExpressionKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerExpressionKeyboard (
    QWidget * wd,
    std::function< ExpressionPtr(void)> init_cb = nullptr,
    std::function< void(ExpressionPtr result_value)> result_cb = nullptr,
    std::shared_ptr< InputValidator > validator = nullptr)
```

registerIntegerKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerIntegerKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

registerIPAddressKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerIPAddressKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

registerNumberKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerNumberKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

registerPasswordKeyboard() [1/2]

```
void arcs::aubo_scope::KeyboardManager::registerPasswordKeyboard (
    QWidget * wd,
    std::function< ExpressionPtr(void)> init_cb,
    std::function< void(ExpressionPtr)> result_cb,
    const std::shared_ptr< InputValidator > & validator = nullptr)
```

registerPasswordKeyboard() [2/2]

```
ARCS_DEPRECATED void arcs::aubo_scope::KeyboardManager::registerPasswordKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

注册密码键盘

registerPositiveDoubleKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerPositiveDoubleKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

registerPositiveIntegerKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerPositiveIntegerKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

registerStandardKeyboard() [1/2]

```
void arcs::aubo_scope::KeyboardManager::registerStandardKeyboard (
    QWidget * wd,
    std::function< ExpressionPtr(void)> init_cb,
    std::function< void(ExpressionPtr)> result_cb,
    const std::shared_ptr< InputValidator > & validator = nullptr)
```

registerStandardKeyboard

参数

<i>wd</i>	当前要书写的控件，比如 ui->lineEdit
<i>init_cb</i>	当前控件存储的内容以表达式指针的形式表示 将指针指向的内容发送到键盘
<i>result_cb</i>	将当前键盘的内容的 Expression 存储 然后以 string 的格式显示到控件
<i>validator</i>	键盘的过滤器

registerStandardKeyboard() [2/2]

```
ARCS_DEPRECATED void arcs::aubo_scope::KeyboardManager::registerStandardKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
registerStandardKeyboard 注册标准键盘
```

参数

<i>wd</i>	当前要书写的控件，比如 ui->lineEdit
<i>init_cb</i>	初始 edit 自带的字符串输出到键盘
<i>validator</i>	键盘的过滤器
<i>result_cb</i>	把当前键盘中的值发送到 edit

registerStringKeyboard() [1/2]

```
void arcs::aubo_scope::KeyboardManager::registerStringKeyboard (
    QWidget * wd,
    std::function< ExpressionPtr(void)> init_cb,
    std::function< void(ExpressionPtr)> result_cb,
    const std::shared_ptr< InputValidator > & validator = nullptr)
```

registerStringKeyboard() [2/2]

```
ARCS_DEPRECATED void arcs::aubo_scope::KeyboardManager::registerStringKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
注册字符串键盘
```

11.77.4 友元及相关符号说明

DataSwitch

friend class DataSwitch [friend]
 在文件 [keyboard_manager.h](#) 第 109 行定义。
 引用了 [DataSwitch](#).
 被这些函数引用 [DataSwitch](#).

11.77.5 类成员变量说明

d_

```
void* arcs::aubo_scope::KeyboardManager::d_ { nullptr } [private]
在文件 keyboard\_manager.h 第 111 行定义。  

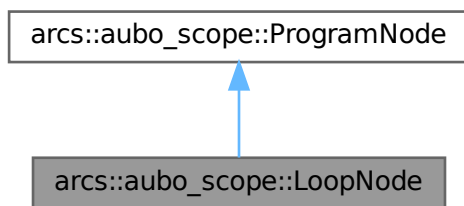
该类的文档由以下文件生成:
```

- [include/aubo_caps/domain/user_interface/keyboard_manager.h](#)

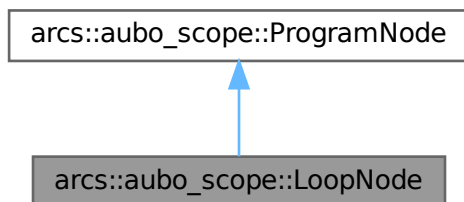
11.78 arcs::aubo_scope::LoopNode 类参考

```
#include <loop_node.h>
```

类 arcs::aubo_scope::LoopNode 继承关系图:



arcs::aubo_scope::LoopNode 的协作图:



Public 类型

- enum ConfigType { ALWAYS , COUNTER , EXPRESSION }

The configuration type used to determine which type of configuration this instance is.

Public 成员函数

- LoopNode (LoopNode &f)
- LoopNode (LoopNode &&f)
- virtual ~LoopNode ()
- ConfigType getConfigType ()
This method returns the type of configuration.
- void setConfigType (ConfigType type)
- void setLoopCount (int loop_count)
Creates a loop counter configuration.
- int getLoopCount ()
- void setExpression (ExpressionPtr expression)
Create a loop expression configuration.
- ExpressionPtr getExpression ()
- bool isEvaluateContinously ()
- void setEvaluateContinously (bool cont)
- void setLoopCountVariable (VariablePtr var)
- VariablePtr getLoopCountVariable ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *Script←Writer::writeChildren*。

Private 成员函数

- [LoopNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.78.1 详细描述

在文件 [loop_node.h](#) 第 12 行定义.

11.78.2 成员枚举类型说明

ConfigType

```
enum arcs::aubo\_scope::LoopNode::ConfigType
```

The configuration type used to determine which type of configuration this instance is.

枚举值

ALWAYS	Looping infinitely has been selected. The config instance can be cast to AlwaysLoopNodeConfig
COUNTER	Looping a certain number of times has been selected. The config instance can be cast to CounterLoopNodeConfig
EXPRESSION	Looping as long as the evaluation of an expression is true has been selected. The config instance can be cast to ExpressionLoopNodeConfig

在文件 [loop_node.h](#) 第 19 行定义.

11.78.3 构造及析构函数说明

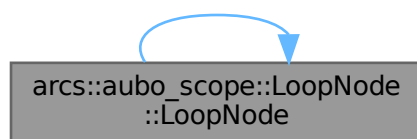
LoopNode() [1/3]

```
arcs::aubo\_scope::LoopNode::LoopNode (  
    LoopNode & f)
```

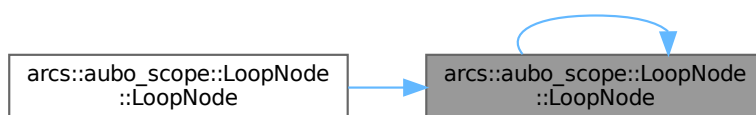
引用了 [LoopNode\(\)](#).

被这些函数引用 [LoopNode\(\)](#) , 以及 [LoopNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

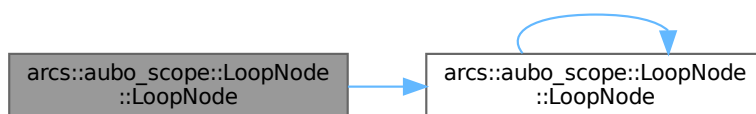


LoopNode() [2/3]

```
arcs::aubo_scope::LoopNode::LoopNode (  
    LoopNode && f)
```

引用了 [LoopNode\(\)](#).

函数调用图:



~LoopNode()

```
virtual arcs::aubo_scope::LoopNode::~~LoopNode () [virtual]
```

LoopNode() [3/3]

```
arcs::aubo_scope::LoopNode::LoopNode () [private]
```

11.78.4 成员函数说明

getConfigType()

```
ConfigType arcs::aubo_scope::LoopNode::getConfigType ()
```

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

返回

the type of this config.

getExpression()

```
ExpressionPtr arcs::aubo_scope::LoopNode::getExpression ()
```

getLoopCount()

```
int arcs::aubo_scope::LoopNode::getLoopCount ()
```

getLoopCountVariable()

```
VariablePtr arcs::aubo_scope::LoopNode::getLoopCountVariable ()
```

isEvaluateContinously()

```
bool arcs::aubo_scope::LoopNode::isEvaluateContinously ()
```

返回

true if the expression will be evaluated continuously during the looping (i.e. the loop can be interrupted anytime during its execution), otherwise **false** and the expression will be evaluated once per loop iteration.

setConfigType()

```
void arcs::aubo_scope::LoopNode::setConfigType (
    ConfigType type)
```

setEvaluateContinously()

```
void arcs::aubo_scope::LoopNode::setEvaluateContinously (
    bool cont)
```

setExpression()

```
void arcs::aubo_scope::LoopNode::setExpression (
    ExpressionPtr expression)
```

Create a loop expression configuration.

参数

<i>expression</i>	the expression whose evaluation must be true to continue looping, not null .
<i>evaluateContinuously</i>	if true the expression will be evaluated continuously during the looping (i.e. the loop can be interrupted anytime during its execution), otherwise false and the expression will be evaluated once per loop iteration.

返回

the configuration.

setLoopCount()

```
void arcs::aubo_scope::LoopNode::setLoopCount (
    int loop_count)
```

Creates a loop counter configuration.

参数

<i>loopCount</i>	the number of iterations through the loop node.
<i>errorHandler</i>	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid loop count value.

返回

the configuration.

setLoopCountVariable()

```
void arcs::aubo_scope::LoopNode::setLoopCountVariable (
    VariablePtr var)
```

返回

the variable for storing the loop count

11.78.5 友元及相关符号说明

DataSwitch

friend class DataSwitch [friend]
在文件 [loop_node.h](#) 第 108 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.78.6 类成员变量说明

d_

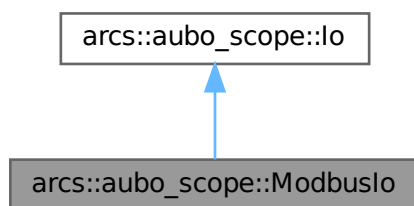
void* arcs::aubo_scope::LoopNode::d_ { nullptr } [private]
在文件 [loop_node.h](#) 第 110 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/loop_node.h](#)

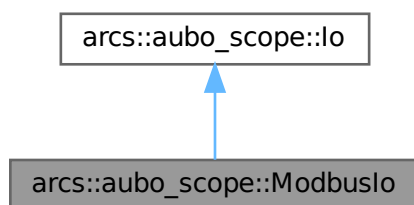
11.79 arcs::aubo_scope::ModbusIo 类参考

Modbus IO 接口此接口提供对 MODBUS I/O 的支持。
`#include <modbus_io.h>`

类 arcs::aubo_scope::ModbusIo 继承关系图:



arcs::aubo_scope::ModbusIo 的协作图:



Public 成员函数

- [ModbusIo](#) (ModbusIo &f)
- [ModbusIo](#) (ModbusIo &&f)
- [~ModbusIo](#) ()
- std::string [getIpAddress](#) ()
获取此MODBUS I/O 的IP 地址
- std::string [getSignalAddress](#) ()
获取此MODBUS I/O 的信号地址
- bool [isSlave](#) ()
判断此MODBUS I/O 是否为从站
- bool [setValue](#) (int value)
设置MODBUS 信号的整数值。
- int [getValue](#) ()
获取MODBUS 信号的整数值。

Public 成员函数继承自 [arcs::aubo_scope::Io](#)

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- virtual [~Io](#) ()
- std::string [getDisplayname](#) () const
获取当前安装中I/O 的用户自定义名称

- `std::string getDefaultName () const`
获取I/O 的机器人默认名称
- `std::string getValueStr () const`
获取I/O 的当前读数值（字符串形式）
- `IoType getType () const`
获取I/O 类型
- `InterfaceType getInterfaceType () const`
获取此I/O 所属的接口类型
- `int getAction () const`
获取I/O 的动作编号
- `bool isInput () const`
判断I/O 是否支持读取值
- `bool isResolvable () const`
某些I/O（如MODBUS I/O 和模拟工具输入）在某些情况下可能不可用。此方法用于判断 I/O 是否可用。
- `size_t hashCode ()`

Private 成员函数

- `ModbusIo ()`

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数

Public 类型继承自 `arcs::aubo_scope::Io`

- `enum IoType : int {
DIGITAL , ANALOG , INTEGER , BOOLEAN ,
DOUBLE }`
- `enum InterfaceType : int {
STANDARD , TOOL , CONFIGURABLE , LINKIO ,
MODBUS , GENERAL_PURPOSE , PROFINET , ETHERNET_IP }`

Protected 成员函数继承自 `arcs::aubo_scope::Io`

- `Io ()`

11.79.1 详细描述

Modbus IO 接口此接口提供对 MODBUS I/O 的支持。
在文件 `modbus_io.h` 第 21 行定义。

11.79.2 构造及析构函数说明

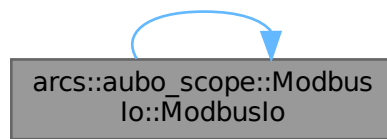
ModbusIo() [1/3]

```
arcs::aubo_scope::ModbusIo::ModbusIo (
    ModbusIo & f)
```

引用了 `ModbusIo()`。

被这些函数引用 `ModbusIo()`，以及 `ModbusIo()`。

函数调用图:



这是这个函数的调用关系图:

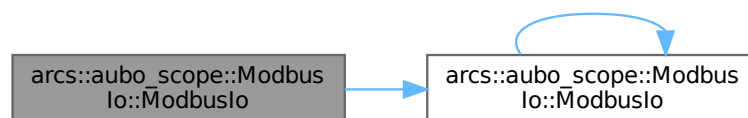


ModbusIo() [2/3]

```
arcs::aubo_scope::ModbusIo::ModbusIo (
    ModbusIo && f)
```

引用了 [ModbusIo\(\)](#).

函数调用图:



~ModbusIo()

```
arcs::aubo_scope::ModbusIo::~~ModbusIo ()
```

ModbusIo() [3/3]

```
arcs::aubo_scope::ModbusIo::ModbusIo () [private]
```

11.79.3 成员函数说明

getIpAddress()

```
std::string arcs::aubo_scope::ModbusIo::getIpAddress ()
```

获取此MODBUS I/O 的IP 地址

返回

IP 地址。

getSignalAddress()

```
std::string arcs::aubo_scope::ModbusIo::getSignalAddress ()
```

获取此MODBUS I/O 的信号地址

返回

信号地址。

getValue()

```
int arcs::aubo_scope::ModbusIo::getValue ()
```

获取MODBUS 信号的整数值。

返回

MODBUS 信号的整数值。

isSlave()

```
bool arcs::aubo_scope::ModbusIo::isSlave ()
```

判断此MODBUS I/O 是否为从站

返回

是否为从站。

setValue()

```
bool arcs::aubo_scope::ModbusIo::setValue (  
    int value)
```

设置MODBUS 信号的整数值。

参数

<i>value</i>	要设置的整数值。

返回

信号下发成功返回 `true`, 否则返回 `false`。

11.79.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [modbus_io.h](#) 第 97 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.79.5 类成员变量说明

d_

```
void* arcs::aubo_scope::ModbusIo::d_ { nullptr } [private]
```

在文件 [modbus_io.h](#) 第 99 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/io/modbus_io.h](#)

11.80 arcs::aubo_scope::ModbusIoFilter 结构体参考

```
#include <modbus_io.h>
```

Public 成员函数

- [ModbusIoFilter](#) ([Io::IoType](#) type, bool is_input)
- bool [operator\(\)](#) ([IoPtr](#) io)

Private 属性

- [Io::IoType](#) type_
- bool [is_input_](#)

11.80.1 详细描述

在文件 [modbus_io.h](#) 第 102 行定义.

11.80.2 构造及析构函数说明

ModbusIoFilter()

```
arcs::aubo_scope::ModbusIoFilter::ModbusIoFilter (
    Io::IoType type,
    bool is\_input) [inline]
```

在文件 [modbus_io.h](#) 第 104 行定义.

引用了 [is_input_](#) , 以及 [type_](#).

11.80.3 成员函数说明

operator()()

```
bool arcs::aubo_scope::ModbusIoFilter::operator() (
    IoPtr io) [inline]
```

在文件 [modbus_io.h](#) 第 108 行定义.

引用了 [is_input_](#) , 以及 [type_](#).

11.80.4 类成员变量说明

is_input_

```
bool arcs::aubo_scope::ModbusIoFilter::is_input_ [private]
```

在文件 [modbus_io.h](#) 第 116 行定义.

被这些函数引用 [ModbusIoFilter\(\)](#) , 以及 [operator\(\)](#).

type_

```
Io::IoType arcs::aubo_scope::ModbusIoFilter::type_ [private]
```

在文件 [modbus_io.h](#) 第 115 行定义.

被这些函数引用 [ModbusIoFilter\(\)](#) , 以及 [operator\(\)](#).

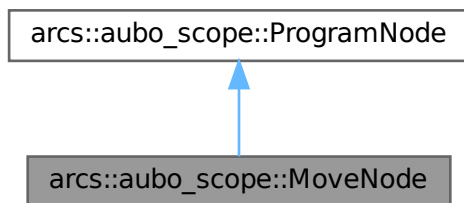
该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/io/modbus_io.h](#)

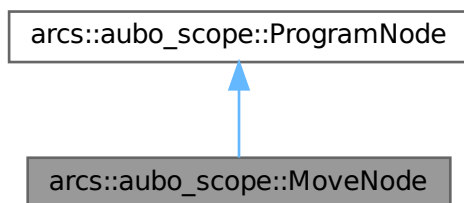
11.81 arcs::aubo_scope::MoveNode 类参考

```
#include <move_node.h>
```

类 arcs::aubo_scope::MoveNode 继承关系图:



arcs::aubo_scope::MoveNode 的协作图:



Public 类型

- enum `ConfigType` : int { `MOVE_J` = 0 , `MOVE_L` , `MOVE_S` }
- enum `TcpSelectionType` : int { `USE_ACTIVE_TCP` = 0 , `IGNORE_ACTIVE_TCP` , `MOVE_TCP` }
- enum `FeatureSelectionType` : int { `UNRESOLVED_FEATURE` , `FEATURE` }

Public 成员函数

- `MoveNode` (MoveNode &f)
- `MoveNode` (MoveNode &&f)
- virtual `~MoveNode` ()
- `ConfigType` `getConfigType` ()
返回配置类型。适当转换此实例以访问特定的 *getter*。
- void `setConfigType` (`ConfigType` type)
- bool `getUseJoints` ()
- void `setUseJoints` (bool use)
- void `setJointSpeed` (double joint_speed)
- double `getJointSpeed` ()
- void `setJointAcceleration` (double joint_acceleration)
- double `getJointAcceleration` ()

- void [setTCPSelection](#) (TCPPtr tcp_selection)
- TCPPtr [getTCPSelection](#) ()
- void [setToolSpeed](#) (double tool_speed)
- double [getToolSpeed](#) ()
- void [setToolAcceleration](#) (double tool_acceleration)
- double [getToolAcceleration](#) ()
- void [setFeature](#) (FeaturePtr feature)
- void [setFeature](#) (FeatureVariablePtr feature_variable)
- FeaturePtr [getFeature](#) ()
- void [setBlend](#) (double blend)
- double [getBlend](#) ()
- void [setTcpSelectionType](#) (TcpSelectionType type)
- TcpSelectionType [getTcpSelectionType](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [MoveNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.81.1 详细描述

在文件 [move_node.h](#) 第 13 行定义.

11.81.2 成员枚举类型说明

ConfigType

```
enum arcs::aubo\_scope::MoveNode::ConfigType : int
```

枚举值

MOVE_J	Movement type is MoveJ. The config instance can be cast to MoveJMoveNodeConfig.
MOVE_L	Movement type is MoveL. The config instance can be cast to MoveLMoveNodeConfig.

MOVE_S	

在文件 `move_node.h` 第 16 行定义.

FeatureSelectionType

```
enum arcs::aubo_scope::MoveNode::FeatureSelectionType : int
```

枚举值

UNRESOLVED_FEATURE	The selected feature is no longer available (e.g. because the user has removed the feature in the Installation or the needed Installation is not currently loaded)
FEATURE	Feature is selected. The Selection instance can be cast to <code>MoveNodeFeature</code>

在文件 `move_node.h` 第 62 行定义.

TcpSelectionType

```
enum arcs::aubo_scope::MoveNode::TcpSelectionType : int
```

枚举值

USE_ACTIVE_TCP	Motions are adjusted to the currently active TCP . The TCP is determined during runtime of the program.
IGNORE_ACTIVE_TCP	No TCP is used and the motion under this Move command will be with respect to the tool output flange.
MOVE_TCP	A specific TCP is selected. The <code>TcpSelection</code> instance can be cast to <code>MoveNodeTCP</code> .

在文件 `move_node.h` 第 40 行定义.

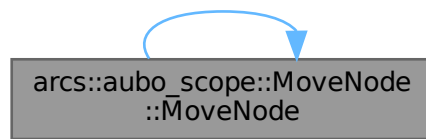
11.81.3 构造及析构造函数说明

MoveNode() [1/3]

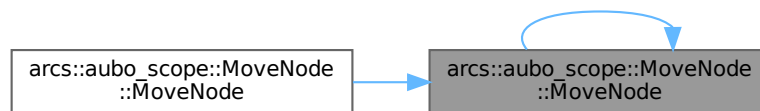
```
arcs::aubo_scope::MoveNode::MoveNode (  
    MoveNode & f)
```

引用了 [MoveNode\(\)](#).
被这些函数引用 [MoveNode\(\)](#) , 以及 [MoveNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

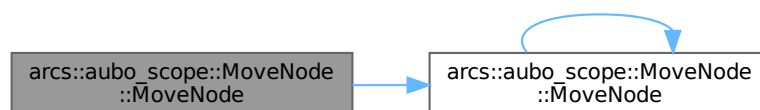


MoveNode() [2/3]

```
arcs::aubo_scope::MoveNode::MoveNode (  
    MoveNode && f)
```

引用了 [MoveNode\(\)](#).

函数调用图:



~MoveNode()

```
virtual arcs::aubo_scope::MoveNode::~~MoveNode () [virtual]
```

MoveNode() [3/3]

```
arcs::aubo_scope::MoveNode::MoveNode () [private]
```

11.81.4 成员函数说明

getBlend()

```
double arcs::aubo_scope::MoveNode::getBlend ()
```

getConfigType()

`ConfigType` arcs::aubo_scope::MoveNode::getConfigType ()
 返回配置类型。适当转换此实例以访问特定的 getter。

返回

此配置的类型

getFeature()

`FeaturePtr` arcs::aubo_scope::MoveNode::getFeature ()

getJointAcceleration()

`double` arcs::aubo_scope::MoveNode::getJointAcceleration ()

getJointSpeed()

`double` arcs::aubo_scope::MoveNode::getJointSpeed ()

getTCPSelection()

`TCPPtr` arcs::aubo_scope::MoveNode::getTCPSelection ()

getTcpSelectionType()

`TcpSelectionType` arcs::aubo_scope::MoveNode::getTcpSelectionType ()

getToolAcceleration()

`double` arcs::aubo_scope::MoveNode::getToolAcceleration ()

getToolSpeed()

`double` arcs::aubo_scope::MoveNode::getToolSpeed ()

getUseJoints()

`bool` arcs::aubo_scope::MoveNode::getUseJoints ()

setBlend()

`void` arcs::aubo_scope::MoveNode::setBlend (
 `double` *blend*)

为 MoveP 配置设置交融参数。
 交融参数将用于机械臂运动中所有路点之间的过渡。

参数

<i>blend</i>	此 Move 节点运动中使用的共享交融值

setConfigType()

`void` arcs::aubo_scope::MoveNode::setConfigType (
 `ConfigType` *type*)

setFeature() [1/2]

```
void arcs::aubo_scope::MoveNode::setFeature (  
    FeaturePtr feature)
```

为 MoveL 配置设置坐标系。

该坐标系将用于机械臂从上一个位置经过 Move 节点下的路点的运动。

参数

<i>feature</i>	此运动的共享坐标系

setFeature() [2/2]

```
void arcs::aubo_scope::MoveNode::setFeature (  
    FeatureVariablePtr feature_variable)
```

setJointAcceleration()

```
void arcs::aubo_scope::MoveNode::setJointAcceleration (  
    double joint_acceleration)
```

为 MoveJ 配置设置关节加速度参数。

该参数适用于机械臂从上一个位置经过 Move 节点下的路点的运动。

参数

<i>joint_acceleration</i>	要使用的共享关节加速度

setJointSpeed()

```
void arcs::aubo_scope::MoveNode::setJointSpeed (  
    double joint_speed)
```

为 MoveJ 配置设置关节速度参数。

该参数适用于机械臂从上一个位置经过 Move 节点下的路点的运动。

参数

<i>joint_speed</i>	要达到的共享关节速度

setTCPSelection()

```
void arcs::aubo_scope::MoveNode::setTCPSelection (  
    TCPPtr tcp_selection)
```

为 MoveJ 配置设置 TCP 选择。

在执行机械臂运动之前，TCP 将被设置为活动 TCP。

参数

<i>tcp_selection</i>	要应用的 TCP 设置

setTcpSelectionType()

```
void arcs::aubo_scope::MoveNode::setTcpSelectionType (
    TcpSelectionType type)
```

setToolAcceleration()

```
void arcs::aubo_scope::MoveNode::setToolAcceleration (
    double tool_acceleration)
```

为 MoveL 配置设置工具加速度参数。

该参数适用于机械臂从上一个位置经过 Move 节点下的路点的运动。

参数

<i>tool_acceleration</i>	要使用的共享工具加速度

setToolSpeed()

```
void arcs::aubo_scope::MoveNode::setToolSpeed (
    double tool_speed)
```

为 MoveL 配置设置工具速度参数。

该参数适用于机械臂从上一个位置经过 Move 节点下的路点的运动。

参数

<i>tool_speed</i>	要达到的共享工具速度

setUseJoints()

```
void arcs::aubo_scope::MoveNode::setUseJoints (
    bool use)
```

11.81.5 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [move_node.h](#) 第 218 行定义。

引用了 [DataSwitch](#)。

被这些函数引用 [DataSwitch](#)。

11.81.6 类成员变量说明**d_**

```
void* arcs::aubo_scope::MoveNode::d_ { nullptr } [private]
```

在文件 [move_node.h](#) 第 220 行定义。

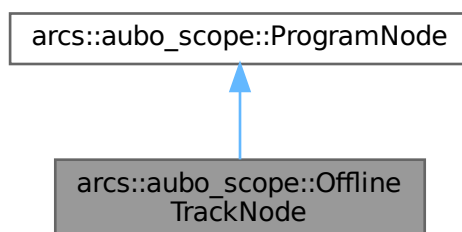
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/move_node.h](#)

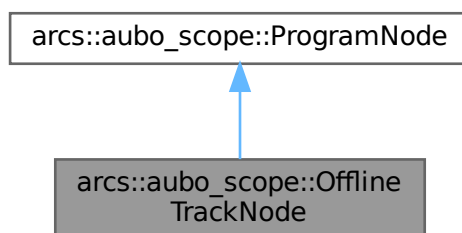
11.82 arcs::aubo_scope::OfflineTrackNode 类参考

```
#include <offline_track_node.h>
```

类 arcs::aubo_scope::OfflineTrackNode 继承关系图:



arcs::aubo_scope::OfflineTrackNode 的协作图:



Public 成员函数

- [OfflineTrackNode](#) (OfflineTrackNode &f)
- [OfflineTrackNode](#) (OfflineTrackNode &&f)
- virtual [~OfflineTrackNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `Script←Writer::writeChildren`。

Private 成员函数

- [OfflineTrackNode](#) ()

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数**Protected 成员函数继承自 `arcs::aubo_scope::ProgramNode`**

- `ProgramNode ()`

11.82.1 详细描述

在文件 `offline_track_node.h` 第 10 行定义.

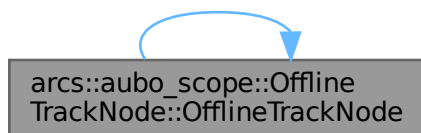
11.82.2 构造及析构函数说明**OfflineTrackNode() [1/3]**

```
arcs::aubo_scope::OfflineTrackNode::OfflineTrackNode (
    OfflineTrackNode & f)
```

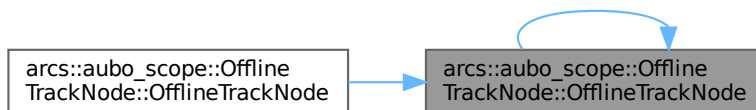
引用了 `OfflineTrackNode()`.

被这些函数引用 `OfflineTrackNode()` , 以及 `OfflineTrackNode()`.

函数调用图:



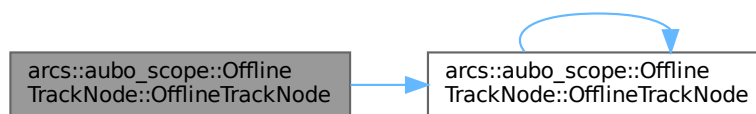
这是这个函数的调用关系图:

**OfflineTrackNode() [2/3]**

```
arcs::aubo_scope::OfflineTrackNode::OfflineTrackNode (
    OfflineTrackNode && f)
```

引用了 `OfflineTrackNode()`.

函数调用图:



~OfflineTrackNode()

```
virtual arcs::aubo_scope::OfflineTrackNode::~~OfflineTrackNode () [virtual]
```

OfflineTrackNode() [3/3]

```
arcs::aubo_scope::OfflineTrackNode::OfflineTrackNode () [private]
```

11.82.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [offline_track_node.h](#) 第 18 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.82.4 类成员变量说明

d_

```
void* arcs::aubo_scope::OfflineTrackNode::d_ { nullptr } [private]
```

在文件 [offline_track_node.h](#) 第 20 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/offline_track_node.h](#)

11.83 arcs::aubo_scope::OutputIoFilter 结构体参考

```
#include <digital_io.h>
```

Public 成员函数

- [bool operator\(\)](#) (IoPtr io)

11.83.1 详细描述

在文件 [digital_io.h](#) 第 85 行定义.

11.83.2 成员函数说明

operator()()

```
bool arcs::aubo_scope::OutputIoFilter::operator() (
    IoPtr io) [inline]
```

在文件 [digital_io.h](#) 第 87 行定义.

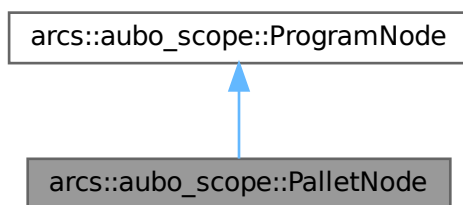
该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/io/digital_io.h](#)

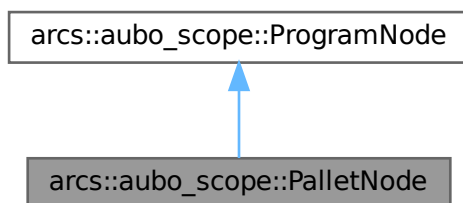
11.84 arcs::aubo_scope::PalletNode 类参考

```
#include <pallet_node.h>
```

类 arcs::aubo_scope::PalletNode 继承关系图:



arcs::aubo_scope::PalletNode 的协作图:



Public 成员函数

- [PalletNode](#) (PalletNode &f)
- [PalletNode](#) (PalletNode &&f)
- virtual [~PalletNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [PalletNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode\(\)](#)

11.84.1 详细描述

在文件 [pallet_node.h](#) 第 10 行定义.

11.84.2 构造及析构函数说明

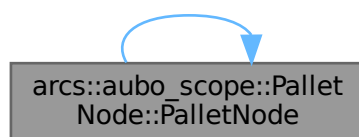
[PalletNode\(\)](#) [1/3]

```
arcs::aubo_scope::PalletNode::PalletNode (  
    PalletNode & f)
```

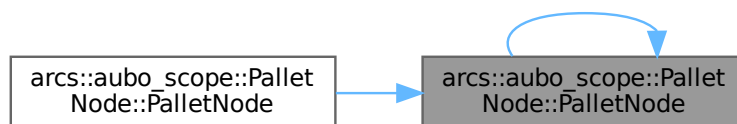
引用了 [PalletNode\(\)](#).

被这些函数引用 [PalletNode\(\)](#) , 以及 [PalletNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

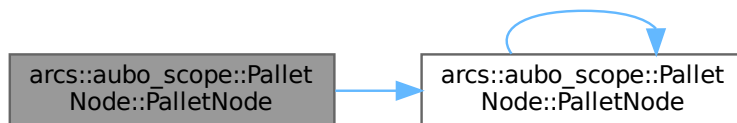


[PalletNode\(\)](#) [2/3]

```
arcs::aubo_scope::PalletNode::PalletNode (  
    PalletNode && f)
```

引用了 [PalletNode\(\)](#).

函数调用图:



~PalletNode()

```
virtual arcs::aubo_scope::PalletNode::~~PalletNode () [virtual]
```

PalletNode() [3/3]

```
arcs::aubo_scope::PalletNode::PalletNode () [private]
```

11.84.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [pallet_node.h](#) 第 18 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.84.4 类成员变量说明

d_

```
void* arcs::aubo_scope::PalletNode::d_ { nullptr } [private]
```

在文件 [pallet_node.h](#) 第 20 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/pallet_node.h](#)

11.85 arcs::aubo_scope::Payload 类参考

获取负载参数API 接口

```
#include <payload.h>
```

Public 类型

- enum [ParametersWarning](#) { [NO_WARNING](#) , [MASS_0KG](#) , [RATED_MASS_EXCEEDED](#) , [MASS_AND_CENTER_OF_GRAVITY_EXCEEDED](#) }

Public 成员函数

- [Payload](#) (Payload &f)
- [Payload](#) (Payload &&f)
- [~Payload](#) ()
- double [getMass](#) () const
- std::vector< double > [getCenterOfGravity](#) () const

Gets the payload's center of gravity (CoG), also referred to as center of mass.

- `std::vector< double > getInertiaMatrix () const`
- `std::string getDisplayName () const`
- `bool isResolvable ()`
- `bool isUsable ()`
- `ParametersWarning getParametersWarning ()`
- `size_t hashCode ()`

Private 成员函数

- `Payload ()`

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

11.85.1 详细描述

获取负载参数API 接口
在文件 `payload.h` 第 16 行定义.

11.85.2 成员枚举类型说明

ParametersWarning

```
enum arcs::aubo_scope::Payload::ParametersWarning
```

This enum represents the different types of warnings that the parameters of a payload can have.

A payload can be usable, but may have a value for a payload parameter or may have a combination of (some of the) payload parameters, that could affect performance of the robot, if the payload is used (i.e., if the parameters are applied).

枚举值

NO_WARNING	None of the parameters of the payload or the combination of payload parameters has any warnings.
MASS_0KG	The payload has a mass of 0 kg which is most likely not intended/valid. Note that the value for the payload mass should be the total mass of the payload attached to the tool output flange of the robot.
RATED_MASS_EXCEEDED	The mass of the payload exceeds the rated maximum supported payload mass of the robot. Using the payload could affect performance of the robot.
MASS_AND_CENTER_OF_GRAVITY_EXCEEDED	The combination of the mass and the center of gravity of could affect the performance of the robot, if the payload is used.

在文件 `payload.h` 第 34 行定义.

11.85.3 构造及析构函数说明

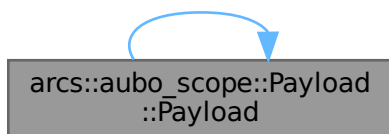
Payload() [1/3]

```
arcs::aubo_scope::Payload::Payload (  
    Payload & f)
```

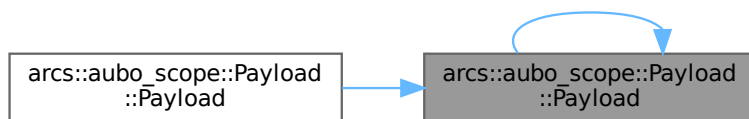
引用了 [Payload\(\)](#).

被这些函数引用 [Payload\(\)](#) , 以及 [Payload\(\)](#).

函数调用图:



这是这个函数的调用关系图:

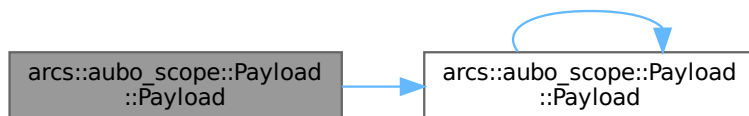


Payload() [2/3]

```
arcs::aubo_scope::Payload::Payload (  
    Payload && f)
```

引用了 [Payload\(\)](#).

函数调用图:



~Payload()

```
arcs::aubo_scope::Payload::~Payload ()
```

Payload() [3/3]

```
arcs::aubo_scope::Payload::Payload () [private]
```

11.85.4 成员函数说明

getCenterOfGravity()

```
std::vector< double > arcs::aubo_scope::Payload::getCenterOfGravity () const
```

Gets the payload's center of gravity (CoG), also referred to as center of mass.

It is defined as the offset between the center of the tool output flange and the center of gravity of the attached payload.

返回

The center of gravity (CoG) of the payload relative to the center of the tool output flange

getDisplayName()

```
std::string arcs::aubo_scope::Payload::getDisplayName () const
```

Gets the name of the payload displayed in the AuboScope UI.

Note: The payload can change name at any time, and thus the name must not be used to uniquely identify a specific payload. The payload name can change in the following situations:

- The payload is renamed by the end user
- The name gets translated (depending on the selected language in AuboScope)
- The name is changed between versions of the AuboCap that added the payload

返回

The name of the payload displayed in AuboScope

getInertiaMatrix()

```
std::vector< double > arcs::aubo_scope::Payload::getInertiaMatrix () const
```

Gets the inertia matrix of the payload. The inertia matrix is defined in a coordinate system with center at the payload's center of gravity (CoG) and the axes aligned with the tool output flange coordinate system (axes).

Note: The inertia matrix specified by the end user can be invalid. The [isUsable](#) method can be used determine, if the payload can be used (i.e., has a valid inertia matrix).

返回

The inertia matrix of the payload

getMass()

```
double arcs::aubo_scope::Payload::getMass () const
```

返回

The total mass of the payload (attached to the tool output flange of the robot)

getParametersWarning()

```
ParametersWarning arcs::aubo_scope::Payload::getParametersWarning ()
```

Use this method to check if the parameters of this payload have any warnings.

A payload can be usable, but may have a value for a payload parameter or may have a combination of (some of the) payload parameters, that could affect performance of the robot, if the payload is used (i.e., if the parameters are applied).

An unresolvable payload (see [isResolvable\(\)](#)) will have no warning for its parameters (i.e. [ParametersWarning#NO_W](#) is returned).

返回

The warning (if any) for the parameters of this payload.

hashCode()

```
size_t arcs::aubo_scope::Payload::hashCode ()
```

isResolvable()

```
bool arcs::aubo_scope::Payload::isResolvable ()
```

A payload cannot be guaranteed to be present in AuboScope. This method can be used to determine, if the payload is present.

The payload will not be present, if the end user loads a different installation which does not contain the payload, or if the payload is removed by the end user or the AuboCap that added the payload.

Note: Storing an undefined payload in a program node contribution's (a [ProgramNodeContribution](#) instance) data model ([DataModel](#) interface) will automatically make the program node (and the program) undefined.

返回

true if this payload is present in AuboScope, otherwise **false**.

isUsable()

```
bool arcs::aubo_scope::Payload::isUsable ()
```

Use this method to determine if the payload can be used. An unresolvable payload (see the [isResolvable](#) method) will always be unusable.

A payload can have an invalid inertia matrix, where all the individual values of the matrix are within the valid range, but the combination of values is invalid.

Note: Storing an unusable payload in a program node contribution's (a [ProgramNodeContribution](#) instance) data model ([DataModel](#) interface) will automatically make the program node (and the program) undefined.

返回

true, if the payload is usable, otherwise **false**. **false** will always be returned when the payload is unresolvable (see the [isResolvable](#) method).

11.85.5 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [payload.h](#) 第 189 行定义。
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.85.6 类成员变量说明

d_

```
void* arcs::aubo_scope::Payload::d_ { nullptr } [private]
```

在文件 [payload.h](#) 第 191 行定义.

该类的文档由以下文件生成:

- include/aubo_caps/domain/payload/[payload.h](#)

11.86 arcs::aubo_scope::PayloadContributionModel 类参考

动态负载管理

```
#include <payload_contribution_model.h>
```

Public 成员函数

- [PayloadContributionModel](#) ([PayloadContributionModel](#) &f)
- [PayloadContributionModel](#) ([PayloadContributionModel](#) &&f)
- [~PayloadContributionModel](#) ()
- [PayloadPtr](#) [addPayload](#) (const std::string &id_key, const std::string &suggested_name, double mass, const std::vector< double > &center_of_gravity)
Payload
- [PayloadPtr](#) [addPayload](#) (const std::string &id_key, const std::string &suggested_name, double mass, const std::vector< double > &center_of_gravity, const std::vector< double > &inertia↵Matrix)
Payload
- [PayloadPtr](#) [getPayload](#) (const std::string &id_key)
Payload 返回此前由此 *AuboCap* 使用相同 *idKey* 标识符添加的负载。使用此方法验证负载是否存在于当前安装中。
- void [updatePayload](#) (const std::string &id_key, double mass, const std::vector< double > &center↵_of_gravity)
Payload
- void [updatePayload](#) (const std::string id_key, double mass, const std::vector< double > &center↵_of_gravity, const std::vector< double > &inertia_matrix)
更新此 *AuboCap* 已添加负载的质量、重心 (CoG) 和惯性矩阵。
- void [removePayload](#) (const std::string &id_key)
从此 *AuboCap* 中移除已添加的负载。使用该负载的程序节点将变为未定义状态, 因为负载不再可解析。

Private 成员函数

- [PayloadContributionModel](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.86.1 详细描述

动态负载管理

在文件 [payload_contribution_model.h](#) 第 17 行定义.

11.86.2 构造及析构造函数说明

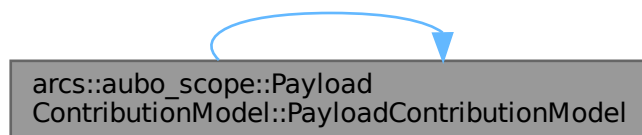
PayloadContributionModel() [1/3]

```
arcs::aubo_scope::PayloadContributionModel::PayloadContributionModel (
    PayloadContributionModel & f)
```

引用了 [PayloadContributionModel\(\)](#).

被这些函数引用 [PayloadContributionModel\(\)](#) , 以及 [PayloadContributionModel\(\)](#).

函数调用图:



这是这个函数的调用关系图:



PayloadContributionModel() [2/3]

```
arcs::aubo_scope::PayloadContributionModel::PayloadContributionModel (
    PayloadContributionModel && f)
```

引用了 [PayloadContributionModel\(\)](#).

函数调用图:



~PayloadContributionModel()

```
arcs::aubo_scope::PayloadContributionModel::~~PayloadContributionModel ()
```

PayloadContributionModel() [3/3]

```
arcs::aubo_scope::PayloadContributionModel::PayloadContributionModel () [private]
```


11.86.3 成员函数说明

addPayload() [1/2]

```
PayloadPtr arcs::aubo_scope::PayloadContributionModel::addPayload (
    const std::string & id_key,
    const std::string & suggested_name,
    double mass,
    const std::vector< double > & center_of_gravity)
```

Payload

向AuboScope 的当前安装中添加一个负载。

添加后终端用户可选择该负载，其他AuboCap 贡献（通过PayloadModel接口）也可使用。该负载不可被终端用户或其他 aubo_studio 插件修改。

系统将根据指定的负载质量和重心自动计算惯性矩阵的默认估算值。如果有负载惯性矩阵的实际数据，可以使用addPayload(String, String, Mass, Position, InertiaMatrix)方法替代（可获得更好的机器人手臂性能）。

注意：指定的负载质量必须是连接到机器人工具输出法兰的负载总质量。

负载质量和重心坐标(CoG)的有效范围可通过RobotLimits接口获取,该接口可通过RobotModel#getRobotLimits()访问。

参数

<i>idKey</i>	用于标识此负载的键，不可为 null 或空字符串。该键仅对此AuboCap 有效，只需在此AuboCap 中唯一，无需对其他 aubo_studio 插件”全局”唯一。
<i>suggestedName</i>	建议的负载名称，不可为 null 或空字符串。有效名称必须匹配正则表达式[a-zA-Z][a-zA-Z0-9_]{0,14}，总共 15 个字符。最终名称可从返回的负载实例获取。
<i>mass</i>	连接到机器人工具输出法兰的负载总质量，不可为 null。使用SimpleValueFactory#createMass(double, Mass.Unit)创建。
<i>centerOfGravity</i>	负载的重心（CoG），也称为质心。定义为工具输出法兰中心与所连接负载重心之间的偏移量。不可为 null。使用PositionFactory 接口创建。

返回

在AuboScope 中创建并注册的负载。

异常

<i>PayloadAlreadyAddedException</i>	如果之前已使用相同的 idKey 标识符添加过负载。使用getPayload(String)检查负载是否已添加到当前安装中。使用updatePayload(String, Mass, Position)或updatePayload(String, Mass, P更新负载。
<i>IllegalPayloadNameException</i>	如果建议的名称不匹配所需的正则表达式。
<i>IllegalMassException</i>	如果指定的质量不在AuboScope 定义的有效范围内。
<i>IllegalCenterOfGravityException</i>	如果指定重心的任何值不在AuboScope 定义的有效范围内。

addPayload() [2/2]

```
PayloadPtr arcs::aubo_scope::PayloadContributionModel::addPayload (
    const std::string & id_key,
    const std::string & suggested_name,
    double mass,
```

```
const std::vector< double > & center_of_gravity,
const std::vector< double > & inertiaMatrix)
```

Payload

向AuboScope 的当前安装中添加一个包含惯性矩阵的负载。

添加后终端用户可选择该负载，其他AuboCap 贡献（通过PayloadModel接口）也可使用。该负载不可被终端用户或其他 aubo_studio 插件修改。

如果没有负载惯性矩阵的数据,请改用addPayload(String, String, Mass, Position, InertiaMatrix)方法。

注意：指定的负载质量必须是连接到机器人工具输出法兰的负载总质量。

负载质量和重心坐标(CoG)的有效范围可通过RobotLimits接口获取,该接口可通过RobotModel#getRobotLimits()访问。

参数

<i>idKey</i>	用于标识此负载的键，不可为 null 或空字符串。该键仅对此AuboCap 有效。
<i>suggestedName</i>	建议的负载名称，不可为 null 或空字符串。
<i>mass</i>	负载总质量，不可为 null。
<i>centerOfGravity</i>	负载的重心（CoG），不可为 null。
<i>inertiaMatrix</i>	负载的惯性矩阵，定义在以负载重心为中心的坐标系中，坐标轴与工具输出法兰坐标系对齐。不可为 null。

返回

在AuboScope 中创建并注册的负载。

异常

<i>PayloadAlreadyAddedException</i>	如果之前已使用相同的 <i>idKey</i> 标识符添加过负载。
<i>IllegalPayloadNameException</i>	如果建议的名称不匹配所需的正则表达式。
<i>IllegalMassException</i>	如果指定的质量不在有效范围内。
<i>IllegalCenterOfGravityException</i>	如果指定重心的任何值不在有效范围内。
<i>IllegalInertiaMatrixException</i>	如果指定惯性矩阵的任何值不在有效范围内。

getPayload()

```
PayloadPtr arcs::aubo_scope::PayloadContributionModel::getPayload (
    const std::string & id_key)
```

Payload 返回此前由此AuboCap 使用相同 *idKey* 标识符添加的负载。使用此方法验证负载是否存在于当前安装中。

参数

<i>idKey</i>	用于标识此负载的键，不可为 null 或空字符串。

返回

此前由此AuboCap 添加的负载。如果当前安装中不存在（指定 idKey 标识符的）负载，则返回 null。

removePayload()

```
void arcs::aubo_scope::PayloadContributionModel::removePayload (
    const std::string & id_key)
```

从此AuboCap 中移除已添加的负载。使用该负载的程序节点将变为未定义状态，因为负载不再可解析。

参数

idKey	添加负载时使用的标识符键，不可为 null 或空字符串。

异常

PayloadNotFoundException	如果当前安装中不存在对应 idKey 标识符的负载。

updatePayload() [1/2]

```
void arcs::aubo_scope::PayloadContributionModel::updatePayload (
    const std::string & id_key,
    double mass,
    const std::vector< double > & center_of_gravity)
```

Payload

更新此AuboCap 已添加负载的质量和重心（CoG）。

注意: 如果已为负载指定了惯性矩阵, 调用此方法将用基于新的负载质量和重心计算的默认估算值替换(覆盖)它。为避免覆盖现有惯性矩阵(或更新现有惯性矩阵),请改用updatePayload(String, Mass, Position, InertiaMatrix)方法。

参数

idKey	用于标识负载的键，不可为 null 或空字符串。调用此方法前必须已使用相同的键添加过负载。
mass	负载总质量，不可为 null。使用SimpleValueFactory#createMass(double, Mass.Unit)创建。
centerOfGravity	负载的重心，不可为 null。使用PositionFactory接口创建。

异常

PayloadNotFoundException	如果当前安装中不存在对应 idKey 标识符的负载。
IllegalMassException	如果指定的质量不在AuboScope 定义的有效范围内。

<i>IllegalCenterOfGravityException</i>	如果指定重心的任何值不在有效范围内。

updatePayload() [2/2]

```
void arcs::aubo_scope::PayloadContributionModel::updatePayload (
    const std::string id_key,
    double mass,
    const std::vector< double > & center_of_gravity,
    const std::vector< double > & inertia_matrix)
```

更新此AuboCap 已添加负载的质量、重心（CoG）和惯性矩阵。

参数

<i>idKey</i>	用于标识负载的键，不可为 null 或空字符串。调用此方法前必须已使用相同的键添加过负载。
<i>mass</i>	负载总质量，不可为 null。使用SimpleValueFactory#createMass(double, Mass.Unit)创建。
<i>centerOfGravity</i>	负载的重心，不可为 null。使用PositionFactory接口创建。
<i>inertiaMatrix</i>	负载的惯性矩阵，不可为 null。

异常

<i>PayloadNotFoundException</i>	如果当前安装中不存在对应 idKey 标识符的负载。
<i>IllegalMassException</i>	如果指定的质量不在有效范围内。
<i>IllegalCenterOfGravityException</i>	如果指定重心的任何值不在有效范围内。
<i>IllegalInertiaMatrixException</i>	如果指定惯性矩阵的任何值不在有效范围内。

11.86.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
在文件 payload\_contribution\_model.h 第 412 行定义.
引用了 DataSwitch.
被这些函数引用 DataSwitch.
```

11.86.5 类成员变量说明

d_

```
void* arcs::aubo_scope::PayloadContributionModel::d_ { nullptr } [private]
在文件 payload\_contribution\_model.h 第 414 行定义.
该类的文档由以下文件生成:
```

- [include/aubo_caps/domain/payload/payload_contribution_model.h](#)

11.87 arcs::aubo_scope::PayloadModel 类参考

获取负载参数API 接口

```
#include <payload_model.h>
```

Public 成员函数

- [PayloadModel](#) (PayloadModel &f)
- [PayloadModel](#) (PayloadModel &&f)
- virtual [~PayloadModel](#) ()
- std::vector< PayloadPtr > [getPayloads](#) () const
getPayloads
- PayloadPtr [getPayload](#) (const std::string &name) const
Payload
- PayloadPtr [getDefaultPayload](#) () const
Payload
- PayloadPtr [getCurrentPayload](#) () const
Payload

Private 成员函数

- [PayloadModel](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.87.1 详细描述

获取负载参数API 接口

在文件 [payload_model.h](#) 第 18 行定义.

11.87.2 构造及析构函数说明

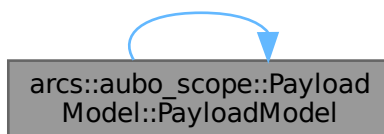
PayloadModel() [1/3]

```
arcs::aubo_scope::PayloadModel::PayloadModel (  
    PayloadModel & f)
```

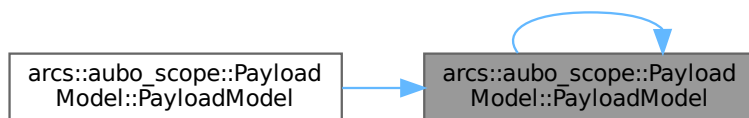
引用了 [PayloadModel\(\)](#).

被这些函数引用 [PayloadModel\(\)](#) , 以及 [PayloadModel\(\)](#).

函数调用图:



这是这个函数的调用关系图:

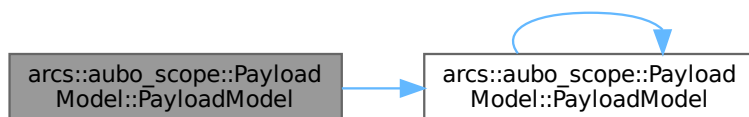


PayloadModel() [2/3]

```
arcs::aubo_scope::PayloadModel::PayloadModel (
    PayloadModel && f)
```

引用了 [PayloadModel\(\)](#).

函数调用图:



~PayloadModel()

```
virtual arcs::aubo_scope::PayloadModel::~~PayloadModel () [virtual]
```

PayloadModel() [3/3]

```
arcs::aubo_scope::PayloadModel::PayloadModel () [private]
```

11.87.3 成员函数说明

getCurrentPayload()

```
PayloadPtr arcs::aubo_scope::PayloadModel::getCurrentPayload () const
Payload
getCurrentPayload
```

返回

getDefaultPayload()

```
PayloadPtr arcs::aubo_scope::PayloadModel::getDefaultPayload () const
Payload
getDefaultPayload
```

返回

getPayload()

```
PayloadPtr arcs::aubo_scope::PayloadModel::getPayload (
    const std::string & name) const
```

[Payload](#)

getPayload

参数

<i>name</i>	

返回

getPayloads()

```
std::vector< PayloadPtr > arcs::aubo_scope::PayloadModel::getPayloads () const
getPayloads
```

返回

A collection of all the payloads from the current installation in AuboScope, including user-defined payloads and payloads added by aubo_studio plugins.

11.87.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [payload_model.h](#) 第 60 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.87.5 类成员变量说明

d_

```
void* arcs::aubo_scope::PayloadModel::d_ { nullptr } [private]
```

在文件 [payload_model.h](#) 第 62 行定义.

该类的文档由以下文件生成:

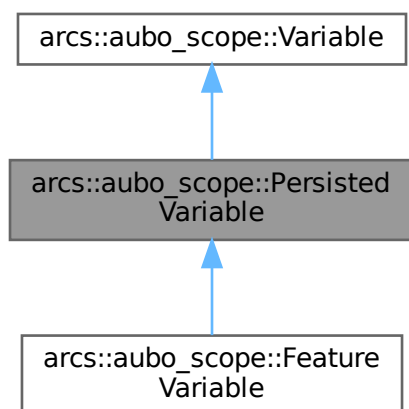
- include/aubo_caps/domain/payload/[payload_model.h](#)

11.88 arcs::aubo_scope::PersistedVariable 类参考

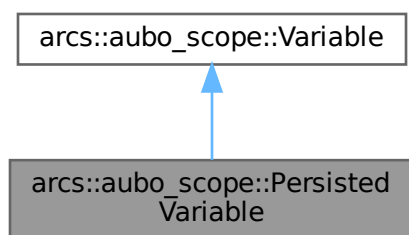
ingroup [ApplicationApi](#) This interface represents variables stored with the installation.

```
#include <persisted_variable.h>
```

类 `arcs::aubo_scope::PersistedVariable` 继承关系图:



`arcs::aubo_scope::PersistedVariable` 的协作图:



Public 成员函数

- `PersistedVariable` (`PersistedVariable &f`)
- `PersistedVariable` (`PersistedVariable &&f`)
- `virtual ~PersistedVariable` ()
- `bool isResolvable` ()

持久化变量无法保证始终存在。当终端用户加载不包含该变量的不同安装或删除该变量时，持久化变量将不可用。如果变量无法解析，存储了 `PersistedVariable` 的程序节点将变为未定义。

Public 成员函数继承自 `arcs::aubo_scope::Variable`

- `Variable` (`Variable &f`)
- `Variable` (`Variable &&f`)
- `virtual ~Variable` ()
- `Type getType` ()
获取变量类型
- `std::string getDisplayName` ()

获取变量显示名称。注意：变量可随时被重命名。

- bool [equals](#) (VariablePtr object)
判断是否引用相同的内部变量
- size_t [hashCode](#) ()
获取哈希码
- std::string [toString](#) ()
获取变量的字符串表示

Protected 成员函数

- [PersistedVariable](#) ()

Protected 成员函数继承自 [arcs::aubo_scope::Variable](#)

- [Variable](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Public 类型继承自 [arcs::aubo_scope::Variable](#)

- enum [Type](#) : int { [GLOBAL](#) , [VALUE_PERSISTED](#) , [FEATURE](#) }
The variable type used to determine which type of variable this instance is.

11.88.1 详细描述

ingroup [ApplicationApi](#) This interface represents variables stored with the installation. These variables are defined in the installation and will exist even when a robot is turned off and on. 在文件 [persisted_variable.h](#) 第 16 行定义.

11.88.2 构造及析构函数说明

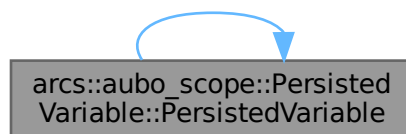
PersistedVariable() [1/3]

```
arcs::aubo_scope::PersistedVariable::PersistedVariable (
    PersistedVariable & f)
```

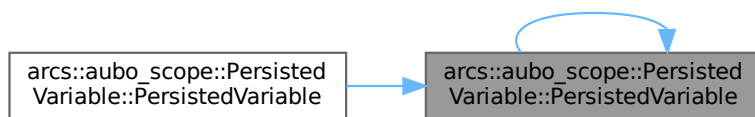
引用了 [PersistedVariable\(\)](#).

被这些函数引用 [PersistedVariable\(\)](#) , 以及 [PersistedVariable\(\)](#).

函数调用图:



这是这个函数的调用关系图:

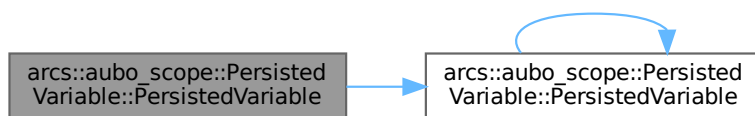


PersistedVariable() [2/3]

```
arcs::aubo_scope::PersistedVariable::PersistedVariable (
    PersistedVariable && f)
```

引用了 [PersistedVariable\(\)](#).

函数调用图:



~PersistedVariable()

```
virtual arcs::aubo_scope::PersistedVariable::~~PersistedVariable () [virtual]
```

PersistedVariable() [3/3]

```
arcs::aubo_scope::PersistedVariable::PersistedVariable () [protected]
```

11.88.3 成员函数说明

isResolvable()

```
bool arcs::aubo_scope::PersistedVariable::isResolvable ()
```

持久化变量无法保证始终存在。当终端用户加载不包含该变量的不同安装或删除该变量时，持久化变量将不可用。如果变量无法解析，存储了PersistedVariable 的程序节点将变为未定义。

返回

如果持久化变量可解析（即存在于当前安装中）返回 `true`。

11.88.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [persisted_variable.h](#) 第 53 行定义。

引用了 [DataSwitch](#)。

被这些函数引用 [DataSwitch](#)。

11.88.5 类成员变量说明

d_

```
void* arcs::aubo_scope::PersistedVariable::d_ { nullptr } [private]
```

在文件 [persisted_variable.h](#) 第 54 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/variable/persisted_variable.h](#)

11.89 arcs::aubo_scope::PersistedVariableFilter 结构体参考

```
#include <persisted_variable.h>
```

Public 成员函数

- [bool operator\(\)](#) (VariablePtr var)

11.89.1 详细描述

在文件 [persisted_variable.h](#) 第 57 行定义.

11.89.2 成员函数说明

[operator\(\)](#)()

```
bool arcs::aubo_scope::PersistedVariableFilter::operator() (
    VariablePtr var) [inline]
```

在文件 [persisted_variable.h](#) 第 59 行定义.

引用了 [arcs::aubo_scope::Variable::VALUE_PERSISTED](#).

该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/variable/persisted_variable.h](#)

11.90 arcs::aubo_scope::PopupManager 类参考

```
#include <popup_manager.h>
```

Public 成员函数

- [PopupManager](#) (PopupManager &f)
- [PopupManager](#) (PopupManager &&f)
- [virtual ~PopupManager](#) ()
- [void notifyPopupShown](#) ()
- [void notifyPopupHidden](#) ()
- [int popupCount](#) () const
- [bool isPopupCountZero](#) () const
- [void setPopupCountChangedCallback](#) (std::function< void(bool)> callback)

Private 成员函数

- [PopupManager](#) ()

Private 属性

- [void * d_](#) { nullptr }

友元

- [struct DefaultSingletonTraits< PopupManager >](#)

11.90.1 详细描述

在文件 `popup_manager.h` 第 16 行定义.

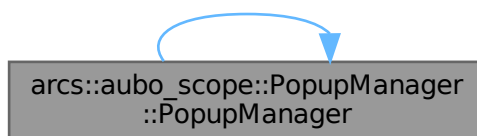
11.90.2 构造及析构函数说明

PopupMenu() [1/3]

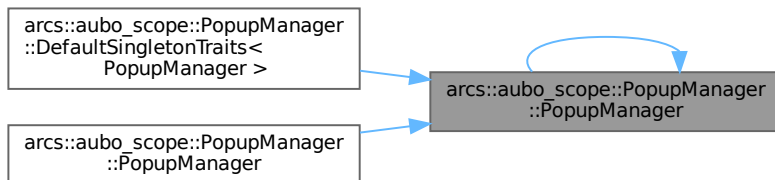
```
arcs::aubo_scope::PopupMenu::PopupMenu (  
    PopupManager & f)
```

引用了 `PopupMenu()`.

被这些函数引用 `DefaultSingletonTraits< PopupManager >, PopupManager()`, 以及 `PopupMenu()`.
函数调用图:



这是这个函数的调用关系图:



PopupMenu() [2/3]

```
arcs::aubo_scope::PopupMenu::PopupMenu (  
    PopupManager && f)
```

引用了 `PopupMenu()`.

函数调用图:



~PopupManager()

```
virtual arcs::aubo_scope::PopupManager::~~PopupManager () [virtual]
```

PopupManager() [3/3]

```
arcs::aubo_scope::PopupManager::PopupManager () [private]
```

11.90.3 成员函数说明**isPopupCountZero()**

```
bool arcs::aubo_scope::PopupManager::isPopupCountZero () const
```

notifyPopupHidden()

```
void arcs::aubo_scope::PopupManager::notifyPopupHidden ()
```

notifyPopupShown()

```
void arcs::aubo_scope::PopupManager::notifyPopupShown ()
```

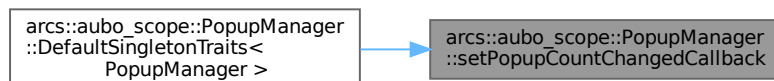
popupCount()

```
int arcs::aubo_scope::PopupManager::popupCount () const
```

setPopupCountChangedCallback()

```
void arcs::aubo_scope::PopupManager::setPopupCountChangedCallback (
    std::function< void(bool)> callback)
```

被这些函数引用 [DefaultSingletonTraits< PopupManager >](#).
这是这个函数的调用关系图:

**11.90.4 友元及相关符号说明****DefaultSingletonTraits< PopupManager >**

```
friend struct DefaultSingletonTraits< PopupManager > [friend]
```

在文件 [popup_manager.h](#) 第 36 行定义.

引用了 [PopupManager\(\)](#) , 以及 [setPopupCountChangedCallback\(\)](#).

11.90.5 类成员变量说明**d_**

```
void* arcs::aubo_scope::PopupManager::d_ { nullptr } [private]
```

在文件 [popup_manager.h](#) 第 41 行定义.

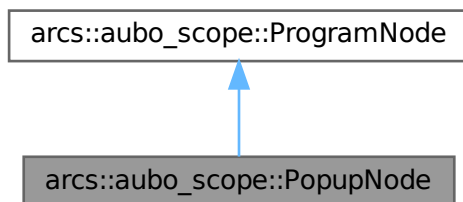
该类的文档由以下文件生成:

- [include/aubo_caps/domain/user_interface/popup_manager.h](#)

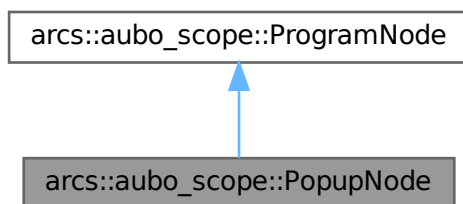
11.91 arcs::aubo_scope::PopupNode 类参考

#include <popup_node.h>

类 arcs::aubo_scope::PopupNode 继承关系图:



arcs::aubo_scope::PopupNode 的协作图:



Public 类型

- enum `Type` { `MESSAGE` , `WARNING` , `ERROR` }

Public 成员函数

- `PopupNode` (`PopupNode` &f)
- `PopupNode` (`PopupNode` &&f)
- virtual `~PopupNode` ()
- void `setMessage` (std::string message)
- std::string `getMessage` ()
- void `setType` (`Type` type)
- `Type` `getType` () const

Public 成员函数继承自 arcs::aubo_scope::ProgramNode

- `ProgramNode` (`ProgramNode` &f)
- `ProgramNode` (`ProgramNode` &&f)
- virtual `~ProgramNode` ()
- `ProgramNodeContributionPtr` `getProgramNodeContribution` ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `ScriptWriter::writeChildren`。

Private 成员函数

- [PopupNode\(\)](#)

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数**Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)**

- [ProgramNode\(\)](#)

11.91.1 详细描述

在文件 [popup_node.h](#) 第 11 行定义.

11.91.2 成员枚举类型说明**Type**

```
enum arcs::aubo\_scope::PopupNode::Type
```

枚举值

MESSAGE	
WARNING	
ERROR	

在文件 [popup_node.h](#) 第 14 行定义.

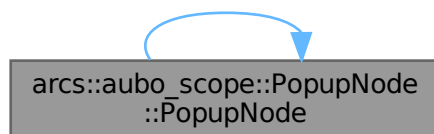
11.91.3 构造及析构函数说明**PopupNode() [1/3]**

```
arcs::aubo\_scope::PopupNode::PopupNode (  
    PopupNode & f)
```

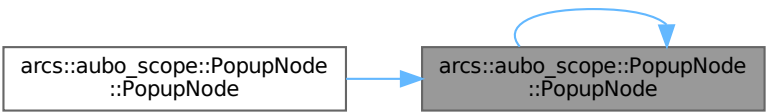
引用了 [PopupNode\(\)](#).

被这些函数引用 [PopupNode\(\)](#) , 以及 [PopupNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:



PopupNode() [2/3]

```
arcs::aubo_scope::PopupNode::PopupNode (  
    PopupNode && f)  
引用了 PopupNode\(\).  
函数调用图:
```



~PopupNode()

```
virtual arcs::aubo_scope::PopupNode::~~PopupNode () [virtual]
```

PopupNode() [3/3]

```
arcs::aubo_scope::PopupNode::PopupNode () [private]
```

11.91.4 成员函数说明

getMessage()

```
std::string arcs::aubo_scope::PopupNode::getMessage ()
```

getType()

```
Type arcs::aubo_scope::PopupNode::getType () const
```

setMessage()

```
void arcs::aubo_scope::PopupNode::setMessage (  
    std::string message)
```

参数

<i>message</i>	the message in the pop up for the end user

返回

this node

异常

<i>IllegalStateException</i>	if called from a Qt-based AuboCap outside of an UndoableChanges scope (see also UndoRedoManager)

setType()

```
void arcs::aubo_scope::PopupNode::setType (  
    Type type)
```

参数

<i>type</i>	the type of pop up

返回

this node

异常

<i>IllegalStateException</i>	if called from a Qt-based AuboCap outside of an UndoableChanges scope (see also UndoRedoManager)

11.91.5 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [popup_node.h](#) 第 48 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.91.6 类成员变量说明

d_

```
void* arcs::aubo_scope::PopupNode::d_ { nullptr } [private]
```

在文件 [popup_node.h](#) 第 50 行定义.
该类的文档由以下文件生成:

- include/aubo_caps/domain/program/nodes/builtin/[popup_node.h](#)

11.92 arcs::aubo_scope::PrepareToStartScrewdriverParameters 类参考

```
#include <screw_driver_configuration.h>
```

Public 成员函数

- virtual `~PrepareToStartScrewdriverParameters()`=default

11.92.1 详细描述

This interface provides parameters relevant for generating script code for the Prepare To Start Screwdriver capability (registered using `ScrewdriverCapabilities#registerPrepareToStartScrewdriverCapability(ScriptCodeGenerator)`).

These parameters are passed when AuboScope calls the implementation of the `ScriptCodeGenerator#generateScript` method responsible for the generating the script code.

The interface is intentionally empty, but could have parameters in future releases.

在文件 `screw_driver_configuration.h` 第 61 行定义.

11.92.2 构造及析构函数说明

`~PrepareToStartScrewdriverParameters()`

```
virtual arcs::aubo_scope::PrepareToStartScrewdriverParameters::~~PrepareToStartScrewdriverParameters () [virtual],
[default]
```

该类的文档由以下文件生成:

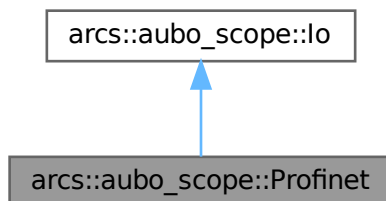
- `include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h`

11.93 arcs::aubo_scope::Profinet 类参考

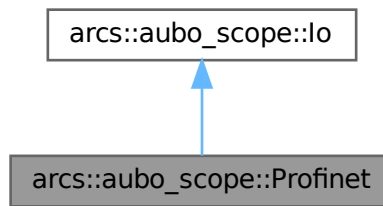
PROFINET IO 接口此接口提供对 PROFINET I/O 的支持。

```
#include <profinet.h>
```

类 `arcs::aubo_scope::Profinet` 继承关系图:



arcs::aubo_scope::Profinet 的协作图:



Public 成员函数

- [Profinet](#) (Profinet &f)
- [Profinet](#) (Profinet &&f)
- [~Profinet](#) ()
- [int getByteOffset](#) ()
获取此PROFINET I/O 的字节偏移量
- [int getBitOffset](#) ()
获取此PROFINET I/O 的位偏移量
- [bool isSlave](#) ()
判断此PROFINET I/O 是否为从站
- [bool setValue](#) (int value)
设置PROFINET 信号的整数值。如果PROFINET 信号是数字量, 则 *false* 用 0 表示, *true* 用所有非 0 数字表示。
- [int getValue](#) ()
获取PROFINET 信号的整数值。如果PROFINET 信号是数字量, 则 *false* 用 0 表示, *true* 用值 1 表示。

Public 成员函数继承自 [arcs::aubo_scope::Io](#)

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- [virtual ~Io](#) ()
- [std::string getDisplayName](#) () const
获取当前安装中I/O 的用户自定义名称
- [std::string getDefaultName](#) () const
获取I/O 的机器人默认名称
- [std::string getValueStr](#) () const
获取I/O 的当前读数值 (字符串形式)
- [IoType getType](#) () const
获取I/O 类型
- [InterfaceType getInterfaceType](#) () const
获取此I/O 所属的接口类型
- [int getAction](#) () const
获取I/O 的动作编号
- [bool isInput](#) () const
判断I/O 是否支持读取值
- [bool isResolvable](#) () const
某些I/O (如MODBUS I/O 和模拟工具输入) 在某些情况下可能不可用。此方法用于判断 I/O 是否可用。
- [size_t hashCode](#) ()

Private 成员函数

- [Profinet](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数**Public 类型继承自 [arcs::aubo_scope::Io](#)**

- enum [IoType](#) : int {
 [DIGITAL](#) , [ANALOG](#) , [INTEGER](#) , [BOOLEAN](#) ,
 [DOUBLE](#) }
- enum [InterfaceType](#) : int {
 [STANDARD](#) , [TOOL](#) , [CONFIGURABLE](#) , [LINKIO](#) ,
 [MODBUS](#) , [GENERAL_PURPOSE](#) , [PROFINET](#) , [ETHERNET_IP](#) }

Protected 成员函数继承自 [arcs::aubo_scope::Io](#)

- [Io](#) ()

11.93.1 详细描述

PROFINET IO 接口此接口提供对 PROFINET I/O 的支持。
在文件 [profinet.h](#) 第 21 行定义.

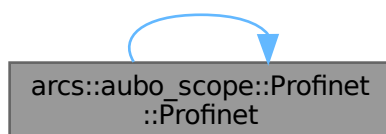
11.93.2 构造及析构函数说明**Profinet()** [1/3]

```
arcs::aubo_scope::Profinet::Profinet (
    Profinet & f)
```

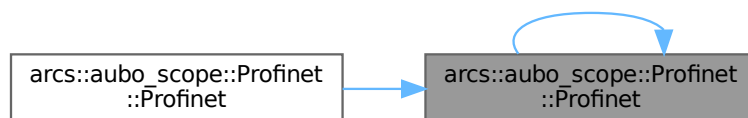
引用了 [Profinet\(\)](#).

被这些函数引用 [Profinet\(\)](#) , 以及 [Profinet\(\)](#).

函数调用图:



这是这个函数的调用关系图:

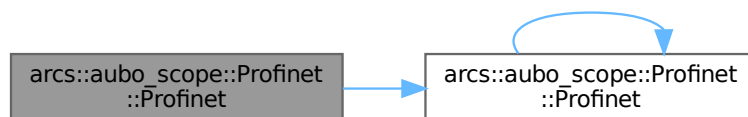


`Profinet()` [2/3]

```
arcs::aubo_scope::Profinet::Profinet (  
    Profinet && f)
```

引用了 [Profinet\(\)](#).

函数调用图:



`~Profinet()`

```
arcs::aubo_scope::Profinet::~~Profinet ()
```

`Profinet()` [3/3]

```
arcs::aubo_scope::Profinet::Profinet () [private]
```

11.93.3 成员函数说明

`getBitOffset()`

```
int arcs::aubo_scope::Profinet::getBitOffset ()  
获取此PROFINET I/O 的位偏移量
```

返回

位偏移量。

`getByteOffset()`

```
int arcs::aubo_scope::Profinet::getByteOffset ()  
获取此PROFINET I/O 的字节偏移量
```

返回

字节偏移量。

getValue()

```
int arcs::aubo_scope::Profinet::getValue ()
```

获取PROFINET 信号的整数值。如果PROFINET 信号是数字量，则 `false` 用 0 表示，`true` 用值 1 表示。

返回

PROFINET 信号的整数值。

isSlave()

```
bool arcs::aubo_scope::Profinet::isSlave ()
```

判断此PROFINET I/O 是否为从站

返回

是否为从站。

setValue()

```
bool arcs::aubo_scope::Profinet::setValue (
    int value)
```

设置PROFINET 信号的整数值。如果PROFINET 信号是数字量，则 `false` 用 0 表示，`true` 用所有非 0 数字表示。

参数

<i>value</i>	要设置的整数值。

返回

如果PROFINET 信号已下发到控制器则返回 `true`，如果信号未下发（例如控制器未运行）则返回 `false`。注意PROFINET 信号必须支持写入，请查阅Aubo PROFINET 协议。

11.93.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [profinet.h](#) 第 109 行定义。
引用了 [DataSwitch](#)。
被这些函数引用 [DataSwitch](#)。

11.93.5 类成员变量说明**d_**

```
void* arcs::aubo_scope::Profinet::d_ { nullptr } [private]
```

在文件 [profinet.h](#) 第 111 行定义。
该类的文档由以下文件生成:

- [include/aubo_caps/domain/io/profinet.h](#)

11.94 arcs::aubo_scope::ProfinetFilter 结构体参考

```
#include <profinet.h>
```

Public 成员函数

- [ProfinetFilter](#) ([Io::IoType](#) type, bool is_input)
- bool [operator\(\)](#) (IoPtr io)

Private 属性

- [Io::IoType](#) type_
- bool [is_input_](#)

11.94.1 详细描述

在文件 [profinet.h](#) 第 114 行定义.

11.94.2 构造及析构造函数说明**ProfinetFilter()**

```
arcs::aubo_scope::ProfinetFilter::ProfinetFilter (
    Io::IoType type,
    bool is_input) [inline]
```

在文件 [profinet.h](#) 第 116 行定义.

引用了 [is_input_](#), 以及 [type_](#).

11.94.3 成员函数说明**operator()()**

```
bool arcs::aubo_scope::ProfinetFilter::operator() (
    IoPtr io) [inline]
```

在文件 [profinet.h](#) 第 120 行定义.

引用了 [is_input_](#), 以及 [type_](#).

11.94.4 类成员变量说明**is_input_**

```
bool arcs::aubo_scope::ProfinetFilter::is_input_ [private]
```

在文件 [profinet.h](#) 第 128 行定义.

被这些函数引用 [operator\(\)\(\)](#), 以及 [ProfinetFilter\(\)](#).

type_

```
Io::IoType arcs::aubo_scope::ProfinetFilter::type_ [private]
```

在文件 [profinet.h](#) 第 127 行定义.

被这些函数引用 [operator\(\)\(\)](#), 以及 [ProfinetFilter\(\)](#).

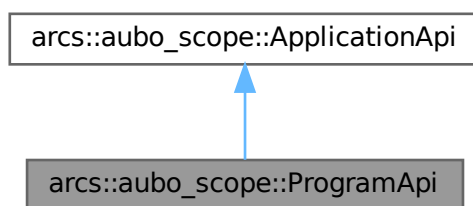
该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/io/profinet.h](#)

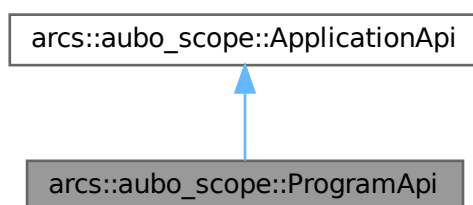
11.95 arcs::aubo_scope::ProgramApi 类参考

```
#include <program_api.h>
```

类 `arcs::aubo_scope::ProgramApi` 继承关系图:



`arcs::aubo_scope::ProgramApi` 的协作图:



Public 成员函数

- `ProgramApi` (`ProgramApi &f`)
- `ProgramApi` (`ProgramApi &&f`)
- `virtual ~ProgramApi ()`
- `ProgramModelPtr` `getProgramModel ()`
ProgramModel
- `WaypointModelPtr` `getWaypointModel ()`
WaypointModel
- `UndoRedoManagerPtr` `getUndoRedoManager ()`
UndoRedoManager

Public 成员函数继承自 `arcs::aubo_scope::ApplicationApi`

- `ApplicationApi` (`ApplicationApi &f`)
- `ApplicationApi` (`ApplicationApi &&f`)
- `virtual ~ApplicationApi ()`
- `IoModelPtr` `getIoModel ()`
IoModel 获取 *IO* 模型接口, 用于操作机器人的输入和输出。
- `FeatureModelPtr` `getFeatureModel ()`
FeatureModel 获取特征模型接口, 用于操作当前安装中的特征。
- `TcpModelPtr` `getTcpModel ()`
TcpModel 获取 *TCP* 模型接口, 用于访问当前安装中的 *TCP*。

- PayloadModelPtr [getPayloadModel](#) ()
PayloadModel 获取负载模型接口，用于访问当前 *AuboScope* 安装中的负载。
- VariableModelPtr [getVariableModel](#) ()
VariableModel 获取变量模型接口，用于操作变量。
- ValueFactoryPtr [getValueFactory](#) ()
ValueFactory 获取值工厂接口，用于访问能够创建值对象的各种工厂。
- `template<typename T>`
`std::shared_ptr< T >` [getInstallationNode](#) ()
此方法用于获取指定的 *InstallationNodeContribution* 实例。
- InstallationNodeContributionPtr [getInstallationNode](#) (const char *typeidOfClass)
- DeviceManagerPtr [getDeviceManager](#) ()
DeviceManager 此方法用于获取指定的 *DeviceManager* 实例。
- `aubo_sdk::RobotProxyPtr` [getRobotProxy](#) ()
获取机器人代理，提供与机器人底层 *SDK* 的交互接口。

Private 成员函数

- [ProgramApi](#) ()

Private 属性

- `void * d_` { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ApplicationApi](#)

- [ApplicationApi](#) ()

11.95.1 详细描述

在文件 [program_api.h](#) 第 30 行定义.

11.95.2 构造及析构函数说明

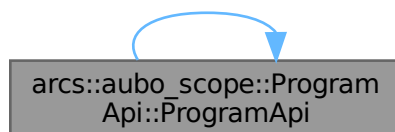
[ProgramApi](#)() [1/3]

```
arcs::aubo_scope::ProgramApi::ProgramApi (  
    ProgramApi & f)
```

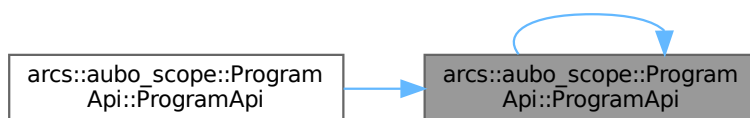
引用了 [ProgramApi](#)() .

被这些函数引用 [ProgramApi](#)() , 以及 [ProgramApi](#)() .

函数调用图:



这是这个函数的调用关系图:

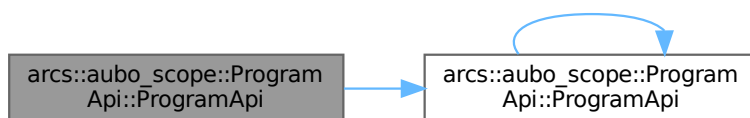


ProgramApi() [2/3]

```
arcs::aubo_scope::ProgramApi::ProgramApi (
    ProgramApi && f)
```

引用了 [ProgramApi\(\)](#).

函数调用图:



~ProgramApi()

```
virtual arcs::aubo_scope::ProgramApi::~~ProgramApi () [virtual]
```

ProgramApi() [3/3]

```
arcs::aubo_scope::ProgramApi::ProgramApi () [private]
```

11.95.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [program_api.h](#) 第 75 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.95.4 类成员变量说明

d_

```
void* arcs::aubo_scope::ProgramApi::d_ { nullptr } [private]
```

在文件 [program_api.h](#) 第 77 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/program_api.h](#)

11.96 arcs::aubo_scope::ProgramApiProvider 类参考

程序 API 提供者提供与程序节点相关的功能和服务访问。这包括 AuboScope 内部已有的功能，以及在 AuboScope 中创建要使用的额外模型元素。

```
#include <program_api_provider.h>
```

Public 成员函数

- [ProgramApiProvider](#) (ProgramApiProvider &f)
- [ProgramApiProvider](#) (ProgramApiProvider &&f)
- virtual [~ProgramApiProvider](#) ()
- SystemApiPtr [getSystemApi](#) ()
提供对系统相关功能的访问
- UserInterfaceApiPtr [getUserInterfaceApi](#) ()
提供对用户界面和终端用户交互相关功能的访问
- ProgramApiPtr [getProgramApi](#) ()
提供对程序相关功能的访问

Private 成员函数

- [ProgramApiProvider](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.96.1 详细描述

程序 API 提供者提供与程序节点相关的功能和服务访问。这包括 AuboScope 内部已有的功能，以及在 AuboScope 中创建要使用的额外模型元素。

在文件 [program_api_provider.h](#) 第 26 行定义.

11.96.2 构造及析构函数说明

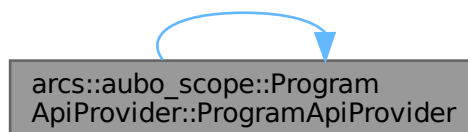
ProgramApiProvider() [1/3]

```
arcs::aubo_scope::ProgramApiProvider::ProgramApiProvider (  
    ProgramApiProvider & f)
```

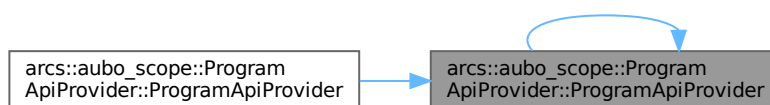
引用了 [ProgramApiProvider\(\)](#).

被这些函数引用 [ProgramApiProvider\(\)](#) , 以及 [ProgramApiProvider\(\)](#).

函数调用图:



这是这个函数的调用关系图:

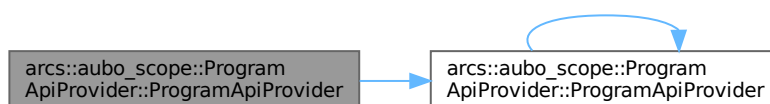


ProgramApiProvider() [2/3]

```
arcs::aubo_scope::ProgramApiProvider::ProgramApiProvider (
    ProgramApiProvider && f)
```

引用了 [ProgramApiProvider\(\)](#).

函数调用图:



~ProgramApiProvider()

```
virtual arcs::aubo_scope::ProgramApiProvider::~~ProgramApiProvider () [virtual]
```

ProgramApiProvider() [3/3]

```
arcs::aubo_scope::ProgramApiProvider::ProgramApiProvider () [private]
```

11.96.3 成员函数说明

getProgramApi()

```
ProgramApiPtr arcs::aubo_scope::ProgramApiProvider::getProgramApi ()
```

提供对程序相关功能的访问

返回

ProgramAPI 实例

getSystemApi()

```
SystemApiPtr arcs::aubo_scope::ProgramApiProvider::getSystemApi ()
```

提供对系统相关功能的访问

返回

SystemAPI 实例

getUserInterfaceApi()

UserInterfaceApiPtr arcs::aubo_scope::ProgramApiProvider::getUserInterfaceApi ()
 提供对用户界面和终端用户交互相关功能的访问

返回

UserInterfaceAPI 实例

11.96.4 友元及相关符号说明**DataSwitch**

friend class DataSwitch [friend]
 在文件 [program_api_provider.h](#) 第 78 行定义。
 引用了 [DataSwitch](#).
 被这些函数引用 [DataSwitch](#).

11.96.5 类成员变量说明**d_**

void* arcs::aubo_scope::ProgramApiProvider::d_ { nullptr } [private]
 在文件 [program_api_provider.h](#) 第 79 行定义。
 该类的文档由以下文件生成:

- [include/aubo_caps/contribution/program/program_api_provider.h](#)

11.97 arcs::aubo_scope::ProgramContributionConfiguration 类参考

程序贡献配置提供对安装节点贡献属性进行配置的访问。

#include <program_contribution_configuration.h>

Public 成员函数

- [ProgramContributionConfiguration](#) (ProgramContributionConfiguration &f)
- [ProgramContributionConfiguration](#) (ProgramContributionConfiguration &&f)
- virtual [~ProgramContributionConfiguration](#) ()
- void [setChildrenAllowed](#) (bool childrenAllowed)
If set to true it is possible for the program node to have child nodes.
- void [setChildrenAllowedOnly](#) (const std::vector< std::string > &childrenList)
- void [setUserInsertable](#) (bool userInsertable)
If set to false this program node will not appear in the Program Tab in AuboScope and can not be inserted in the program tree by the end user.
- void [setDeprecated](#) (bool deprecated)
If set to true it is not possible for the end user to create new program nodes of this type.
- void [setDisplayOrderId](#) (double displayOrderId)
Set the display order id for this type of program node contribution.
- void [setAllowBreakpointOnNode](#) (bool isBreakpointAllowed)
- void [setAllowBreakpointOnChildNodesInSubtree](#) (bool isBreakpointOnChildNodesAllowed)
- void [setAllowStartFromNode](#) (bool isStartFromNodeAllowed)
- void [setAllowStartFromChildNodesInSubtree](#) (bool isStartFromChildNodesAllowed)
- void [setAllowInsertOnlyIntoNodes](#) (const std::vector< std::string > &parentNodeNames)
Configure parent node of this node.

Private 成员函数

- [ProgramContributionConfiguration](#) ()

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

11.97.1 详细描述

程序贡献配置提供对安装节点贡献属性进行配置访问。

在文件 `program_contribution_configuration.h` 第 26 行定义。

11.97.2 构造及析构函数说明

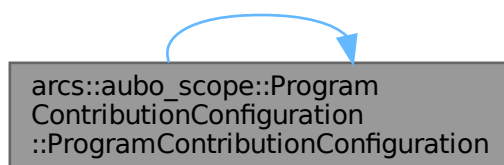
ProgramContributionConfiguration() [1/3]

```
arcs::aubo_scope::ProgramContributionConfiguration::ProgramContributionConfiguration (
    ProgramContributionConfiguration & f)
```

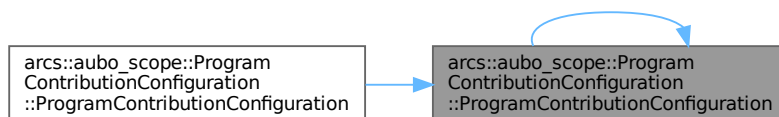
引用了 `ProgramContributionConfiguration()`。

被这些函数引用 `ProgramContributionConfiguration()` , 以及 `ProgramContributionConfiguration()`。

函数调用图:



这是这个函数的调用关系图:

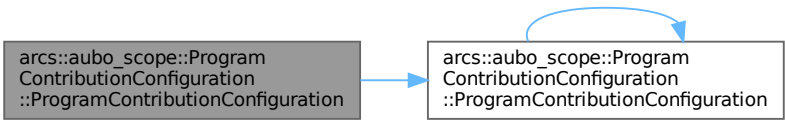


ProgramContributionConfiguration() [2/3]

```
arcs::aubo_scope::ProgramContributionConfiguration::ProgramContributionConfiguration (
    ProgramContributionConfiguration && f)
```

引用了 `ProgramContributionConfiguration()`。

函数调用图:



~ProgramContributionConfiguration()

```
virtual arcs::aubo_scope::ProgramContributionConfiguration::~~ProgramContributionConfiguration () [virtual]
```

ProgramContributionConfiguration() [3/3]

```
arcs::aubo_scope::ProgramContributionConfiguration::ProgramContributionConfiguration () [private]
```

11.97.3 成员函数说明

setAllowBreakpointOnChildNodesInSubtree()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setAllowBreakpointOnChildNodesInSubtree (
    bool isBreakpointOnChildNodesAllowed)
```

Configure whether or not the end user is allowed to set a breakpoint on or single step any of the child nodes in the sub-tree under the AuboCap program node. Note that it will only be possible to set a breakpoint on a child node if the child node itself supports breakpoints.

The default value is `false`.

参数

<i>isBreakpointOnChildNodesAllowed</i>	<code>true</code> if it should be allowed to set a breakpoint on or single step any of the child nodes in the sub-tree under this AuboCap program node. If it should be disallowed, the value must be <code>false</code> .

setAllowBreakpointOnNode()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setAllowBreakpointOnNode (
    bool isBreakpointAllowed)
```

Configure whether or not the end user is allowed to set a breakpoint on or single step the AuboCap program node.

The default value is `true`.

参数

<i>isBreakpointAllowed</i>	<code>true</code> if it should be allowed to set a breakpoint on or single step this AuboCap program node. If it should be disallowed, the value must be <code>false</code> .

setAllowInsertOnlyIntoNodes()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setAllowInsertOnlyIntoNodes (
    const std::vector< std::string > & parentNodeNames)
```

Configure parent node of this node.

End user cannot insert this node to

The default value is "".

参数

<i>parentNodeName</i>	

setAllowStartFromChildNodesInSubtree()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setAllowStartFromChildNodesInSubtree (
    bool isStartFromChildNodesAllowed)
```

Configure whether or not the end user can start the program directly from any selected child node in the sub-tree under the AboCap program node. Note that it will only be possible to start from a selected child node if the child node itself supports starting from the child node.

The default value is **false**.

参数

<i>isStartFromChildNodesAllowed</i>	true if it should be allowed to start the program directly from any child program node. If it should be disallowed, the value must be false ..

setAllowStartFromNode()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setAllowStartFromNode (
    bool isStartFromNodeAllowed)
```

Configure whether or not the end user can start the program from the AboCap program node when it is selected in the program tree.

The default value is **true**.

参数

<i>isStartFromNodeAllowed</i>	true if it should be allowed to start the program directly from this AboCap program node. If it should be disallowed, the value must be false .

setChildrenAllowed()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setChildrenAllowed (
    bool childrenAllowed)
```

If set to **true** it is possible for the program node to have child nodes.

The default value is **false**.

参数

<i>childrenAllowed</i>	determines whether this node can have child nodes

setChildrenAllowedOnly()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setChildrenAllowedOnly (
    const std::vector< std::string > & childrenList)
```

setDeprecated()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setDeprecated (
    bool deprecated)
```

If set to **true** it is not possible for the end user to create new program nodes of this type.

Loading of existing programs will however still be supported for this program node.

The default value is **false**.

参数

<i>deprecated</i>	determines whether this node is deprecated

setDisplayOrderId()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setDisplayOrderId (
    double displayOrderId)
```

Set the display order id for this type of program node contribution.

This makes it possible to specify how all the program node contributions from a AuboCap should be ordered/sorted in AuboScope.

All program node contributions from a AuboCap will be grouped together and sorted (in ascending order) according to their display order id.

The default value is 0.0. If the display order id for all program node contributions in a AuboCap have the default value, the standard built-in sorting in AuboScope will be used.

参数

<i>display← OrderId</i>	Determines the display order of the AuboCap program node type

setUserInsertable()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setUserInsertable (
    bool userInsertable)
```

If set to **false** this program node will not appear in the Program Tab in AuboScope and can not be inserted in the program tree by the end user.

It can only be inserted programmatically using the [ProgramNodeFactory](#). This also means, it can not be inserted using copy/paste of that node directly. It can however be inserted by copy/paste of an ancestor of such a node.

The default value is **true**.

参数

<i>userInsertable</i>	determines whether the end user can insert this node

11.97.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [program_contribution_configuration.h](#) 第 181 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.97.5 类成员变量说明

d_

```
void* arcs::aubo_scope::ProgramContributionConfiguration::d_ { nullptr } [private]
```

在文件 [program_contribution_configuration.h](#) 第 182 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/contribution/program/program_contribution_configuration.h](#)

11.98 arcs::aubo_scope::ProgramCreationContext 类参考

程序创建上下文提供安装节点创建时所在上下文的相关信息。

```
#include <program_creation_context.h>
```

Public 类型

- [enum NodeCreationType { NEW , LOAD }](#)

Public 成员函数

- [ProgramCreationContext \(ProgramCreationContext &f\)](#)
- [ProgramCreationContext \(ProgramCreationContext &&f\)](#)
- [~ProgramCreationContext \(\)](#)
- [NodeCreationType getNodeCreationType \(\)](#)
- [std::string getCreatedVersion \(\)](#)
- [std::string getLastSavedVersion \(\)](#)

Private 成员函数

- [ProgramCreationContext \(\)](#)

Private 属性

- [void * d_ { nullptr }](#)

友元

- [class DataSwitch](#)

11.98.1 详细描述

程序创建上下文提供安装节点创建时所在上下文的相关信息。

在文件 [program_creation_context.h](#) 第 21 行定义.

11.98.2 成员枚举类型说明

NodeCreationType

```
enum arcs::aubo_scope::ProgramCreationContext::NodeCreationType
```

枚举值

NEW	
LOAD	

在文件 [program_creation_context.h](#) 第 24 行定义.

11.98.3 构造及析构造函数说明

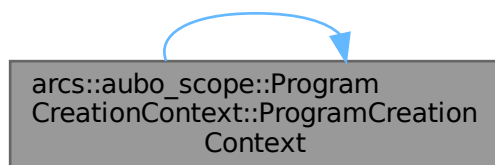
ProgramCreationContext() [1/3]

```
arcs::aubo_scope::ProgramCreationContext::ProgramCreationContext (  
    ProgramCreationContext & f)
```

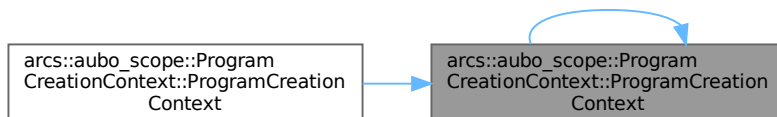
引用了 [ProgramCreationContext\(\)](#).

被这些函数引用 [ProgramCreationContext\(\)](#) , 以及 [ProgramCreationContext\(\)](#).

函数调用图:



这是这个函数的调用关系图:



ProgramCreationContext() [2/3]

```
arcs::aubo_scope::ProgramCreationContext::ProgramCreationContext (  
    ProgramCreationContext && f)
```

引用了 [ProgramCreationContext\(\)](#).

函数调用图:



~ProgramCreationContext()

```
arcs::aubo_scope::ProgramCreationContext::~~ProgramCreationContext ()
```

ProgramCreationContext() [3/3]

```
arcs::aubo_scope::ProgramCreationContext::ProgramCreationContext () [private]
```

11.98.4 成员函数说明

getCreatedVersion()

```
std::string arcs::aubo_scope::ProgramCreationContext::getCreatedVersion ()
```

getLastSavedVersion()

```
std::string arcs::aubo_scope::ProgramCreationContext::getLastSavedVersion ()
```

getNodeCreationType()

```
NodeCreationType arcs::aubo_scope::ProgramCreationContext::getNodeCreationType ()
```

11.98.5 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [program_creation_context.h](#) 第 40 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.98.6 类成员变量说明

d_

```
void* arcs::aubo_scope::ProgramCreationContext::d_ { nullptr } [private]
```

在文件 [program_creation_context.h](#) 第 42 行定义.

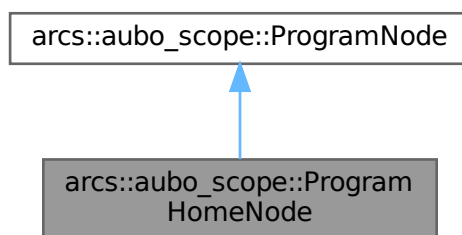
该类的文档由以下文件生成:

- [include/aubo_caps/contribution/program/program_creation_context.h](#)

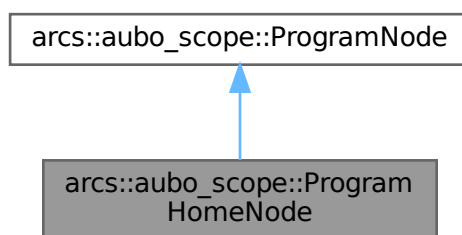
11.99 arcs::aubo_scope::ProgramHomeNode 类参考

```
#include <program_home_node.h>
```

类 arcs::aubo_scope::ProgramHomeNode 继承关系图:



arcs::aubo_scope::ProgramHomeNode 的协作图:



Public 成员函数

- [ProgramHomeNode](#) (ProgramHomeNode &f)
- [ProgramHomeNode](#) (ProgramHomeNode &&f)
- virtual [~ProgramHomeNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [ProgramHomeNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode\(\)](#)

11.99.1 详细描述

在文件 [program_home_node.h](#) 第 12 行定义.

11.99.2 构造及析构函数说明

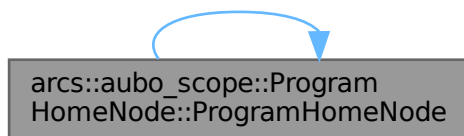
ProgramHomeNode() [1/3]

```
arcs::aubo_scope::ProgramHomeNode::ProgramHomeNode (  
    ProgramHomeNode & f)
```

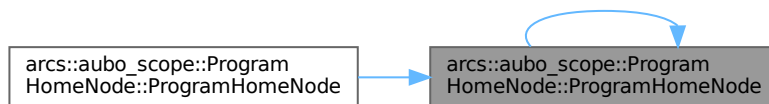
引用了 [ProgramHomeNode\(\)](#).

被这些函数引用 [ProgramHomeNode\(\)](#) , 以及 [ProgramHomeNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

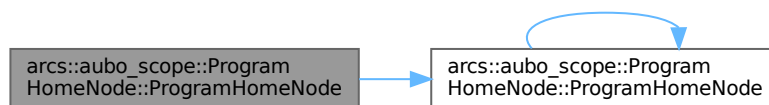


ProgramHomeNode() [2/3]

```
arcs::aubo_scope::ProgramHomeNode::ProgramHomeNode (  
    ProgramHomeNode && f)
```

引用了 [ProgramHomeNode\(\)](#).

函数调用图:



~ProgramHomeNode()

```
virtual arcs::aubo_scope::ProgramHomeNode::~~ProgramHomeNode () [virtual]
```

ProgramHomeNode() [3/3]

```
arcs::aubo_scope::ProgramHomeNode::ProgramHomeNode () [private]
```

11.99.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [program_home_node.h](#) 第 22 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.99.4 类成员变量说明

d_

```
void* arcs::aubo_scope::ProgramHomeNode::d_ { nullptr } [private]
```

在文件 [program_home_node.h](#) 第 24 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/program_home_node.h](#)

11.100 arcs::aubo_scope::ProgramModel 类参考

程序模型

```
#include <program_model.h>
```

Public 成员函数

- [ProgramModel](#) (ProgramModel &f)
- [ProgramModel](#) (ProgramModel &&f)
- virtual [~ProgramModel](#) ()
- ProgramNodeFactoryPtr [getProgramNodeFactory](#) ()
返回一个 [ProgramNodeFactory](#) 用于创建程序节点。
- TreeNodePtr [getTreeNode](#) (ProgramNodeContribution *root)
从 [ProgramNodeContribution](#) 获取 [TreeNode](#) 根节点。从这里可以添加子节点以形成子树。

Private 成员函数

- [ProgramModel](#) ()

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

11.100.1 详细描述

程序模型

概述

此接口使得修改 AudoStudio 插件的子树成为可能。树的结构与节点的内容相分离。`TreeNode` 接口关注树的结构，而`ProgramNode` 接口处理树节点的具体内容。

程序修改入门

`ProgramModel` 包含：

- `getProgramNodeFactory()` - 可以创建新的程序节点（包括 AudoScope 内置节点和 AudoCap 程序节点）。
- `getRootTreeNode(ProgramNodeContribution root)` - 返回给定 AudoCap `ProgramNodeContribution` 的子树

`TreeNode` 接口支持节点的插入和删除。使用它可以获取特定节点的子节点列表。对树节点调用 `lockChildSequence` 会锁定该节点下的直接子节点。即终端用户无法重新排列、删除子节点或将其他节点插入到子节点序列中。

在文件 `program_model.h` 第 58 行定义。

11.100.2 构造及析构函数说明

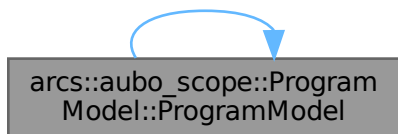
`ProgramModel()` [1/3]

```
arcs::audo_scope::ProgramModel::ProgramModel (
    ProgramModel & f)
```

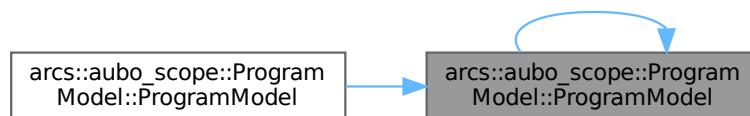
引用了 `ProgramModel()`。

被这些函数引用 `ProgramModel()`，以及 `ProgramModel()`。

函数调用图：



这是这个函数的调用关系图:

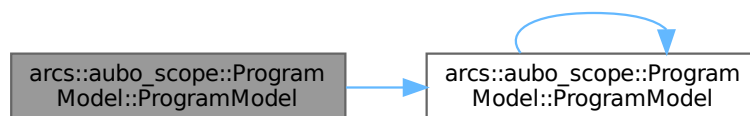


ProgramModel() [2/3]

```
arcs::aubo_scope::ProgramModel::ProgramModel (
    ProgramModel && f)
```

引用了 [ProgramModel\(\)](#).

函数调用图:



~ProgramModel()

```
virtual arcs::aubo_scope::ProgramModel::~~ProgramModel () [virtual]
```

ProgramModel() [3/3]

```
arcs::aubo_scope::ProgramModel::ProgramModel () [private]
```

11.100.3 成员函数说明

getProgramNodeFactory()

```
ProgramNodeFactoryPtr arcs::aubo_scope::ProgramModel::getProgramNodeFactory ()
```

返回一个 [ProgramNodeFactory](#) 用于创建程序节点。

返回

用于创建程序节点的工厂

getTreeNode()

```
TreeNodePtr arcs::aubo_scope::ProgramModel::getTreeNode (
    ProgramNodeContribution * root)
```

从 [ProgramNodeContribution](#) 获取 [TreeNode](#) 根节点。从这里可以添加子节点以形成子树。

参数

<i>root</i>	要作为子树根的 AuboCap 程序节点

返回

[TreeNode](#) 根节点

11.100.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [program_model.h](#) 第 98 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.100.5 类成员变量说明

d_

```
void* arcs::aubo_scope::ProgramModel::d_ { nullptr } [private]
```

在文件 [program_model.h](#) 第 100 行定义.

该类的文档由以下文件生成:

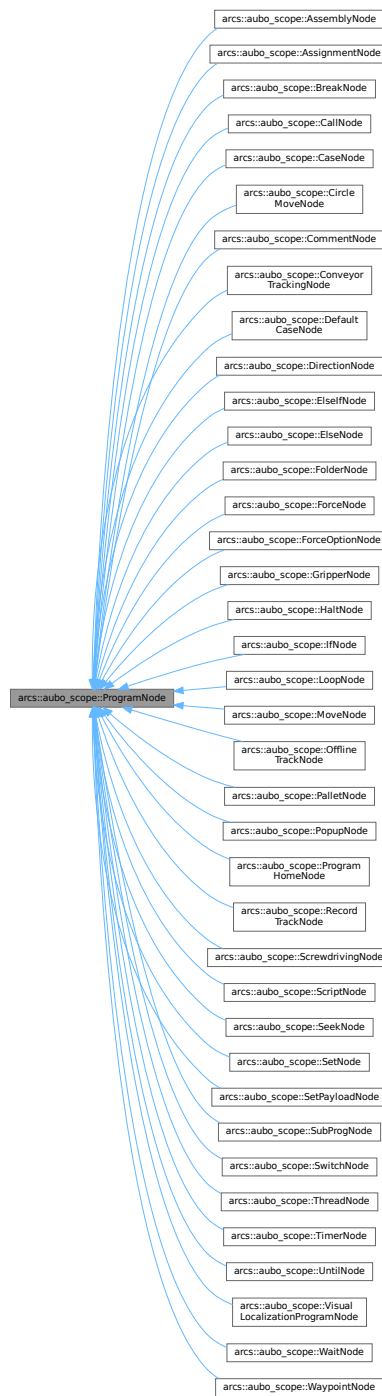
- [include/aubo_caps/domain/program/program_model.h](#)

11.101 arcs::aubo_scope::ProgramNode 类参考

程序节点此接口表示 AuboScope 中的程序节点。

```
#include <program_node.h>
```

类 arcs::aubo_scope::ProgramNode 继承关系图:



Public 成员函数

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `ScriptWriter::writeChildren`。

Protected 成员函数

- `ProgramNode()`

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

11.101.1 详细描述

程序节点此接口表示 AuboScope 中的程序节点。
 程序节点可以通过 `ProgramNodeFactory` 创建。
 在文件 `program_node.h` 第 25 行定义。

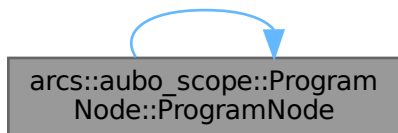
11.101.2 构造及析构函数说明**ProgramNode() [1/3]**

```
arcs::aubo_scope::ProgramNode::ProgramNode (
    ProgramNode & f)
```

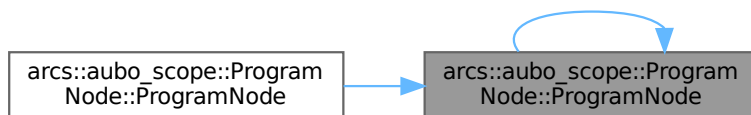
引用了 `ProgramNode()`。

被这些函数引用 `ProgramNode()`，以及 `ProgramNode()`。

函数调用图：



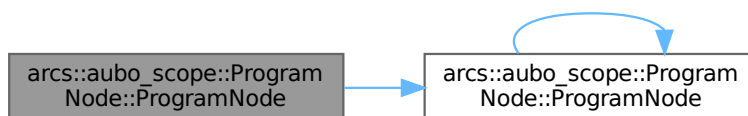
这是这个函数的调用关系图：

**ProgramNode() [2/3]**

```
arcs::aubo_scope::ProgramNode::ProgramNode (
    ProgramNode && f)
```

引用了 `ProgramNode()`。

函数调用图:



~ProgramNode()

```
virtual arcs::aubo_scope::ProgramNode::~~ProgramNode () [virtual]
```

ProgramNode() [3/3]

```
arcs::aubo_scope::ProgramNode::ProgramNode () [protected]
```

11.101.3 成员函数说明

getProgramNodeContribution()

```
ProgramNodeContributionPtr arcs::aubo_scope::ProgramNode::getProgramNodeContribution ()
```

获取程序节点贡献。不推荐用户直接调用 ProgramNodeContribution::generateScript, 应使用 ScriptWriter::writeChildren。

返回

程序节点贡献

11.101.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [program_node.h](#) 第 51 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.101.5 类成员变量说明

d_

```
void* arcs::aubo_scope::ProgramNode::d_ { nullptr } [private]
```

在文件 [program_node.h](#) 第 52 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/program_node.h](#)

11.102 arcs::aubo_scope::ProgramNodeContribution 类参考

程序节点贡献定义了了在 AuboScope 中指定 AuboCap 程序节点所需的 API。

```
#include <program_node_contribution.h>
```

Public 成员函数

- virtual `~ProgramNodeContribution()`=default
- virtual void `openView()`=0
当此节点在程序树中被选中时调用。
- virtual void `closeView()`=0
当此节点在程序树中被取消选中或导航到其他视图时调用。
- virtual std::string `getTitle()`=0
- virtual bool `isDefined()`=0
- virtual void `generateScript` (ScriptWriterPtr script_writer)=0
定义此程序节点在程序中要生成的脚本代码。
- virtual std::string `postAction` (const char *pattern, const std::string &body)
- virtual std::string `getAction` (const char *pattern, const std::string &body)

11.102.1 详细描述

程序节点贡献定义了 `AuboScope` 中指定 `AuboCap` 程序节点所需的 API。
在文件 `program_node_contribution.h` 第 23 行定义。

11.102.2 构造及析构函数说明

`~ProgramNodeContribution()`

```
virtual arcs::aubo_scope::ProgramNodeContribution::~~ProgramNodeContribution () [virtual], [default]
```

11.102.3 成员函数说明

`closeView()`

```
virtual void arcs::aubo_scope::ProgramNodeContribution::closeView () [pure virtual]
```

当此节点在程序树中被取消选中或导航到其他视图时调用。

`generateScript()`

```
virtual void arcs::aubo_scope::ProgramNodeContribution::generateScript (
    ScriptWriterPtr script_writer) [pure virtual]
```

定义此程序节点在程序中要生成的脚本代码。

参数

<i>script_writer</i>	用于添加定义此程序节点行为的脚本命令

`getAction()`

```
virtual std::string arcs::aubo_scope::ProgramNodeContribution::getAction (
    const char * pattern,
    const std::string & body) [inline], [virtual]
```

`getAction` 发送 HTTP GET 请求

参数

<i>pattern</i>	URL 模式
<i>body</i>	请求体

返回

响应内容

在文件 `program_node_contribution.h` 第 153 行定义.

getTitle()

```
virtual std::string arcs::aubo_scope::ProgramNodeContribution::getTitle () [pure virtual]
```

返回

程序树中为此程序节点显示的文本。

异常

<i>IllegalStateException</i>	如果数据模型或子程序子树被修改

isDefined()

```
virtual bool arcs::aubo_scope::ProgramNodeContribution::isDefined () [pure virtual]
```

如果此程序节点完整，应返回 `true`，否则返回 `false`。

未定义的程序节点将在程序树中以黄色标记。只要程序包含一个或多个未定义的程序节点，用户就无法启动程序。

如果此程序节点在其 [DataModel](#) 中存储了不可解析的实体（例如 [PersistedVariable](#) 或 TCP），则该程序节点将被视为未定义。

返回

如果此程序节点完整则返回 `true`，否则返回 `false`。

异常

<i>IllegalStateException</i>	如果数据模型或子程序子树被修改

openView()

```
virtual void arcs::aubo_scope::ProgramNodeContribution::openView () [pure virtual]
```

当此节点在程序树中被选中时调用。

postAction()

```
virtual std::string arcs::aubo_scope::ProgramNodeContribution::postAction (  
    const char * pattern,  
    const std::string & body) [inline], [virtual]
```

postAction 发送 HTTP POST 请求

参数

<i>pattern</i>	URL 模式
<i>body</i>	请求体

返回

响应内容

在文件 `program_node_contribution.h` 第 134 行定义。
该类的文档由以下文件生成:

- `include/aubo_caps/contribution/program/program_node_contribution.h`

11.103 arcs::aubo_scope::ProgramNodeFactory 类参考

程序节点工厂 此接口提供创建不同类型程序节点的方法。

```
#include <program_node_factory.h>
```

Public 成员函数

- `ProgramNodeFactory` (`ProgramNodeFactory &f`)
- `ProgramNodeFactory` (`ProgramNodeFactory &&f`)
- `virtual ~ProgramNodeFactory` ()
- `ProgramNodePtr createUserProgramNode` (`const char *typeid_service`)
创建一个 *AuboCap* 程序节点, 该节点是 *ProgramNodeContribution* 的实例。
- `MoveNodePtr createMoveNode` ()
创建一个默认的移动节点, 其下包含一个默认路点节点。
- `MoveNodePtr createMoveNodeNoTemplate` ()
- `WaypointNodePtr createWaypointNode` ()
创建具有默认路点名称的路点节点 (对应终端用户手动添加新节点时生成的名称)。
- `WaypointNodePtr createWaypointNode` (`const std::string &suggested_name`)
- `UntilNodePtr createUntilNode` ()
创建一个未指定 *until* 类型的 *Until* 节点。
- `DirectionNodePtr createDirectionNode` ()
- `WaitNodePtr createWaitNode` ()
- `SetNodePtr createSetNode` ()
- `SetPayloadNodePtr createSetPayloadNode` ()
- `PopupNodePtr createPopupNode` ()
- `HaltNodePtr createHaltNode` ()
- `CommentNodePtr createCommentNode` ()
- `FolderNodePtr createFolderNode` ()
- `LoopNodePtr createLoopNode` ()
- `AssignmentNodePtr createAssignmentNode` ()
- `IfNodePtr createIfNode` ()
- `ElseIfNodePtr createElseIfNode` ()
- `ElseNodePtr createElseNode` ()
- `CircleMoveNodePtr createCircleMoveNode` ()
- `PalletNodePtr createPalletNode` ()
- `SeekNodePtr createSeekNode` ()
- `ForceNodePtr createForceNode` ()
- `ForceOptionNodePtr createForceOptionNode` ()
- `AssemblyNodePtr createAssemblyNode` ()
- `ScrewdrivingNodePtr createScrewdrivingNode` ()
- `ScriptNodePtr createScriptNode` ()
- `TimerNodePtr createTimerNode` ()
- `SwitchNodePtr createSwitchNode` ()
- `CaseNodePtr createCaseNode` ()
- `DefaultCaseNodePtr createDefaultCaseNode` ()
- `BreakNodePtr createBreakNode` ()
- `SubProgNodePtr createSubProgNode` ()

- CallNodePtr [createCallNode](#) ()
- OfflineTrackNodePtr [createOfflineTrackNode](#) ()
- RecordTrackNodePtr [createRecordTrackNode](#) ()
- ThreadNodePtr [createThreadNode](#) ()
- ProgramHomeNodePtr [createProgramHomeNode](#) ()
- VisualLocalizationProgramNodePtr [createVisualLocalizationProgramNode](#) ()
- ConveyorTrackingNodePtr [createConveyorTrackingNode](#) ()
- GripperNodePtr [createGripperNode](#) (GripperDevicePtr gripperDevice)

Private 成员函数

- [ProgramNodeFactory](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.103.1 详细描述

程序节点工厂此接口提供创建不同类型程序节点的方法。

可以创建内置程序节点（由 AUBO Robots 提供）和 AuboCap 程序节点。大多数方法将创建与终端用户手动添加节点时具有相同默认配置的节点。

请注意，此默认配置可能会因 AuboScope 版本而异。

在文件 [program_node_factory.h](#) 第 70 行定义。

11.103.2 构造及析构函数说明

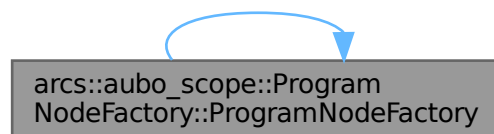
ProgramNodeFactory() [1/3]

```
arcs::aubo_scope::ProgramNodeFactory::ProgramNodeFactory (
    ProgramNodeFactory & f)
```

引用了 [ProgramNodeFactory\(\)](#)。

被这些函数引用 [ProgramNodeFactory\(\)](#)，以及 [ProgramNodeFactory\(\)](#)。

函数调用图：



这是这个函数的调用关系图:



ProgramNodeFactory() [2/3]

```
arcs::aubo_scope::ProgramNodeFactory::ProgramNodeFactory (
    ProgramNodeFactory && f)
```

引用了 [ProgramNodeFactory\(\)](#).

函数调用图:



~ProgramNodeFactory()

```
virtual arcs::aubo_scope::ProgramNodeFactory::~~ProgramNodeFactory () [virtual]
```

ProgramNodeFactory() [3/3]

```
arcs::aubo_scope::ProgramNodeFactory::ProgramNodeFactory () [private]
```

11.103.3 成员函数说明

createAssemblyNode()

```
AssemblyNodePtr arcs::aubo_scope::ProgramNodeFactory::createAssemblyNode ()
```

createAssignmentNode()

```
AssignmentNodePtr arcs::aubo_scope::ProgramNodeFactory::createAssignmentNode ()
```

createBreakNode()

```
BreakNodePtr arcs::aubo_scope::ProgramNodeFactory::createBreakNode ()
```

createCallNode()

```
CallNodePtr arcs::aubo_scope::ProgramNodeFactory::createCallNode ()
```

createCaseNode()

```
CaseNodePtr arcs::aubo_scope::ProgramNodeFactory::createCaseNode ()
```

createCircleMoveNode()

```
CircleMoveNodePtr arcs::aubo_scope::ProgramNodeFactory::createCircleMoveNode ()
```

createCommentNode()

```
CommentNodePtr arcs::aubo_scope::ProgramNodeFactory::createCommentNode ()
```

createConveyorTrackingNode()

```
ConveyorTrackingNodePtr arcs::aubo_scope::ProgramNodeFactory::createConveyorTrackingNode ()
```

createDefaultCaseNode()

```
DefaultCaseNodePtr arcs::aubo_scope::ProgramNodeFactory::createDefaultCaseNode ()
```

createDirectionNode()

```
DirectionNodePtr arcs::aubo_scope::ProgramNodeFactory::createDirectionNode ()
```

createElseIfNode()

```
ElseIfNodePtr arcs::aubo_scope::ProgramNodeFactory::createElseIfNode ()
```

createElseNode()

```
ElseNodePtr arcs::aubo_scope::ProgramNodeFactory::createElseNode ()
```

createFolderNode()

```
FolderNodePtr arcs::aubo_scope::ProgramNodeFactory::createFolderNode ()
```

createForceNode()

```
ForceNodePtr arcs::aubo_scope::ProgramNodeFactory::createForceNode ()
```

createForceOptionNode()

```
ForceOptionNodePtr arcs::aubo_scope::ProgramNodeFactory::createForceOptionNode ()
```

createGripperNode()

```
GripperNodePtr arcs::aubo_scope::ProgramNodeFactory::createGripperNode (  
    GripperDevicePtr gripperDevice)
```

创建新的夹爪程序节点。此节点类型可用于编程所选夹爪设备的夹取和释放动作。
程序节点将使用与终端用户在 AuboScope 中手动添加节点时相同的默认配置创建，即未选择夹爪动作的未定义配置。

参数

<i>gripperDevice</i>	要为其创建夹爪节点的夹爪设备，不能为 <code>null</code> 。创建的夹爪节点将为此设备编程夹爪动作。

返回

新的夹爪程序节点（可使用 [TreeNode](#) 接口插入到程序树中）

异常

<i>UnresolvedDeviceException</i>	当指定的 gripperDevice 设备不可解析时

createHaltNode()

```
HaltNodePtr arcs::aubo_scope::ProgramNodeFactory::createHaltNode ()
```

createIfNode()

```
IfNodePtr arcs::aubo_scope::ProgramNodeFactory::createIfNode ()
```

createLoopNode()

```
LoopNodePtr arcs::aubo_scope::ProgramNodeFactory::createLoopNode ()
```

createMoveNode()

```
MoveNodePtr arcs::aubo_scope::ProgramNodeFactory::createMoveNode ()
```

创建一个默认的移动节点，其下包含一个默认路点节点。

返回

新的移动节点

createMoveNodeNoTemplate()

```
MoveNodePtr arcs::aubo_scope::ProgramNodeFactory::createMoveNodeNoTemplate ()
```

创建一个默认的移动节点，其下不包含路点节点（例如用于插入命名路点）。

注意：移动节点必须至少包含一个路点节点才有效。

返回

新的移动节点

createOfflineTrackNode()

```
OfflineTrackNodePtr arcs::aubo_scope::ProgramNodeFactory::createOfflineTrackNode ()
```

createPalletNode()

```
PalletNodePtr arcs::aubo_scope::ProgramNodeFactory::createPalletNode ()
```

createPopupNode()

```
PopupNodePtr arcs::aubo_scope::ProgramNodeFactory::createPopupNode ()
```

createProgramHomeNode()

```
ProgramHomeNodePtr arcs::aubo_scope::ProgramNodeFactory::createProgramHomeNode ()
```

createRecordTrackNode()

```
RecordTrackNodePtr arcs::aubo_scope::ProgramNodeFactory::createRecordTrackNode ()
```

createScrewdrivingNode()

```
ScrewdrivingNodePtr arcs::aubo_scope::ProgramNodeFactory::createScrewdrivingNode ()
```

createScriptNode()

```
ScriptNodePtr arcs::aubo_scope::ProgramNodeFactory::createScriptNode ()
```

createSeekNode()

```
SeekNodePtr arcs::aubo_scope::ProgramNodeFactory::createSeekNode ()
```

createSetNode()

```
SetNodePtr arcs::aubo_scope::ProgramNodeFactory::createSetNode ()
```

createSetPayloadNode()

```
SetPayloadNodePtr arcs::aubo_scope::ProgramNodeFactory::createSetPayloadNode ()
```

createSubProgNode()

```
SubProgNodePtr arcs::aubo_scope::ProgramNodeFactory::createSubProgNode ()
```

createSwitchNode()

```
SwitchNodePtr arcs::aubo_scope::ProgramNodeFactory::createSwitchNode ()
```

createThreadNode()

```
ThreadNodePtr arcs::aubo_scope::ProgramNodeFactory::createThreadNode ()
```

createTimerNode()

```
TimerNodePtr arcs::aubo_scope::ProgramNodeFactory::createTimerNode ()
```

createUntilNode()

```
UntilNodePtr arcs::aubo_scope::ProgramNodeFactory::createUntilNode ()
```

创建一个未指定 until 类型的 Until 节点。

返回

新的 Until 节点

createUserProgramNode()

```
ProgramNodePtr arcs::aubo_scope::ProgramNodeFactory::createUserProgramNode (  
    const char * typeid_service)
```

创建一个 AuboCap 程序节点，该节点是 [ProgramNodeContribution](#) 的实例。

参数

<i>typeid_service</i>	创建程序节点贡献的服务

返回

由 [ProgramNodeService](#) 创建的表示 [ProgramNodeContribution](#) 的 ProgramNode。如果提供 [ProgramNodeService](#) 的 AuboCap 未安装，则返回 null。

异常

<i>IllegalArgumentException</i>	如果参数 ProgramNodeService 为 <code>null</code> 或未实现 ProgramNodeService

createVisualLocalizationProgramNode()

```
VisualLocalizationProgramNodePtr arcs::aubo_scope::ProgramNodeFactory::createVisualLocalizationProgramNode ( )
```

createWaitNode()

```
WaitNodePtr arcs::aubo_scope::ProgramNodeFactory::createWaitNode ( )
```

createWaypointNode() [1/2]

```
WaypointNodePtr arcs::aubo_scope::ProgramNodeFactory::createWaypointNode ( )
```

创建具有默认路点名称的路点节点（对应终端用户手动添加新节点时生成的名称）。

返回

新的路点节点

createWaypointNode() [2/2]

```
WaypointNodePtr arcs::aubo_scope::ProgramNodeFactory::createWaypointNode (
    const std::string & suggested_name)
```

创建具有建议名称的路点节点。
如果建议名称已被注册，最终名称可能会不同。路点名称在插入程序树时注册。
变量路点将使用变量名称而不是建议名称。

参数

<i>suggested_name</i>	建议的路点名称。有效名称必须匹配正则表达式 <code>[a-zA-Z][a-zA-Z0-9_]{0,14}</code>

返回

新的路点节点

异常

<i>IllegalWaypointNameException</i>	如果名称为非法格式

11.103.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [program_node_factory.h](#) 第 253 行定义。
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.103.5 类成员变量说明

d_

```
void* arcs::aubo_scope::ProgramNodeFactory::d_ { nullptr } [private]
```

在文件 [program_node_factory.h](#) 第 255 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/program_node_factory.h](#)

11.104 arcs::aubo_scope::ProgramNodeService 类参考

程序节点服务定义了向 AuboScope 添加新类型程序节点所需的 API。

```
#include <program_node_service.h>
```

Public 成员函数

- virtual [~ProgramNodeService](#) ()=default
- virtual std::string [getId](#) ()=0
获取此类程序节点的唯一标识符。该标识符将在存储使用此程序节点的程序时使用。
- virtual std::string [getTitle](#) ()=0
- virtual std::string [getIcon](#) ()=0
- virtual void [configureContribution](#) (ProgramContributionConfigurationPtr configuration)=0
配置贡献, 设置程序节点的属性和行为。
- virtual ProgramNodeViewPtr [createView](#) (ViewApiProviderPtr api_provider)=0
创建一个新的视图实例, 实现程序节点画面的 UI。在加载或创建新程序时调用一次。
- virtual ProgramNodeContributionPtr [createNode](#) (ProgramApiProviderPtr api_provider, ProgramNodeViewPtr view, DataModelPtr model, ProgramCreationContextPtr context)=0

11.104.1 详细描述

程序节点服务定义了向 AuboScope 添加新类型程序节点所需的 API。

在文件 [program_node_service.h](#) 第 25 行定义.

11.104.2 构造及析构函数说明

[~ProgramNodeService\(\)](#)

```
virtual arcs::aubo_scope::ProgramNodeService::~~ProgramNodeService () [virtual], [default]
```

11.104.3 成员函数说明

[configureContribution\(\)](#)

```
virtual void arcs::aubo_scope::ProgramNodeService::configureContribution (
    ProgramContributionConfigurationPtr configuration) [pure virtual]
```

配置贡献, 设置程序节点的属性和行为。

参数

<i>configuration</i>	贡献配置对象

createNode()

```
virtual ProgramNodeContributionPtr arcs::aubo_scope::ProgramNodeService::createNode (  
    ProgramApiProviderPtr api_provider,  
    ProgramNodeViewPtr view,  
    DataModelPtr model,  
    ProgramCreationContextPtr context) [pure virtual]
```

创建一个新的程序节点贡献实例。

返回的节点必须使用提供的数据模型对象来检索和存储其中包含的数据。加载或复制/粘贴现有程序节点时，将忽略程序节点构造函数对提供的数据模型所做的所有修改。对模型对象的每次更改都将在程序树中注册为一个单独的撤销/重做事件。

参数

<i>api_provider</i>	提供对 AuboScope 中可用的域数据相关功能的访问
<i>view</i>	程序节点视图
<i>model</i>	新程序节点实例的所有配置数据存储和检索的数据模型对象
<i>context</i>	创建上下文

返回

新构建的程序节点贡献实例

createView()

```
virtual ProgramNodeViewPtr arcs::aubo_scope::ProgramNodeService::createView (  
    ViewApiProviderPtr api_provider) [pure virtual]
```

创建一个新的视图实例，实现程序节点画面的 UI。在加载或创建新程序时调用一次。

参数

<i>api_provider</i>	提供对 AuboScope 中与用户界面和终端用户交互相关的功能和服务访问

返回

视图

getIcon()

```
virtual std::string arcs::aubo_scope::ProgramNodeService::getIcon () [pure virtual]
```

返回

通过此工厂创建的程序节点显示的图标

getId()

```
virtual std::string arcs::aubo_scope::ProgramNodeService::getId () [pure virtual]
```

获取此类程序节点的唯一标识符。该标识符将在存储使用此程序节点的程序时使用。

返回

唯一标识符

getTitle()

```
virtual std::string arcs::aubo_scope::ProgramNodeService::getTitle () [pure virtual]
```

返回

通过此工厂创建的程序节点在结构标签页中显示的文本

该类的文档由以下文件生成:

- include/aubo_caps/contribution/program/[program_node_service.h](#)

11.105 arcs::aubo_scope::ProgramNodeView 类参考

程序节点视图程序节点视图的 API, 定义了对应[ProgramNodeContribution](#) 的 UI。 [ProgramNodeView](#) 实例是多个贡献之间的共享实体。

```
#include <program_node_view.h>
```

Public 成员函数

- virtual [~ProgramNodeView](#) ()=default
- virtual void [buildUi](#) ([WidgetHandle](#) panel, [ContributionProviderPtr](#)< [ProgramNodeContribution](#) > provider)=0

在提供的面板上为对应的 [ProgramNodeContribution](#) 构建 UI。使用贡献进行逻辑和数据模型操作。

11.105.1 详细描述

程序节点视图程序节点视图的 API, 定义了对应[ProgramNodeContribution](#) 的 UI。 [ProgramNodeView](#) 实例是多个贡献之间的共享实体。

在文件 [program_node_view.h](#) 第 25 行定义。

11.105.2 构造及析构函数说明**~ProgramNodeView()**

```
virtual arcs::aubo_scope::ProgramNodeView::~~ProgramNodeView () [virtual], [default]
```

11.105.3 成员函数说明**buildUi()**

```
virtual void arcs::aubo_scope::ProgramNodeView::buildUi (
    WidgetHandle panel,
    ContributionProviderPtr< ProgramNodeContribution > provider) [pure virtual]
```

在提供的面板上为对应的 [ProgramNodeContribution](#) 构建 UI。使用贡献进行逻辑和数据模型操作。

参数

<i>panel</i>	构建 UI 的面板
<i>provider</i>	当前选中的对应程序节点贡献的提供者

该类的文档由以下文件生成:

- include/aubo_caps/contribution/program/[program_node_view.h](#)

11.106 arcs::aubo_scope::ProgramSelectionParameters 类参考

```
#include <screw_driver_configuration.h>
```

Public 成员函数

- virtual `~ProgramSelectionParameters` ()=default
- virtual `ScrewdriverProgram` `getScrewdriverProgram` ()=0
- virtual `ScrewdriverOperationType` `getScrewdriverOperationType` ()=0

11.106.1 详细描述

This interface provides parameters relevant for generating script code for the Program Selection capability (registered using `ScrewdriverCapabilities#registerProgramSelectionCapability(Screwdriver←ProgramListProvider, ScriptCodeGenerator)`).

These parameters are passed when AuboScope calls the implementation of the `ScriptCodeGenerator#generateScript` method responsible for the generating the script code.

在文件 `screw_driver_configuration.h` 第 24 行定义.

11.106.2 构造及析构函数说明

`~ProgramSelectionParameters()`

```
virtual arcs::aubo_scope::ProgramSelectionParameters::~~ProgramSelectionParameters () [virtual], [default]
```

11.106.3 成员函数说明

`getScrewdriverOperationType()`

```
virtual ScrewdriverOperationType arcs::aubo_scope::ProgramSelectionParameters::getScrewdriverOperation←  
Type () [pure virtual]
```

返回

The operation type selected by the end user.

异常

<i>UnsupportedOperationException</i>	if the screwdriver has not registered this capability

`getScrewdriverProgram()`

```
virtual ScrewdriverProgram arcs::aubo_scope::ProgramSelectionParameters::getScrewdriverProgram () [pure  
virtual]
```

返回

The selected screwdriver program to use for the operation.

异常

<i>UnsupportedOperationException</i>	if the screwdriver has not registered this capability

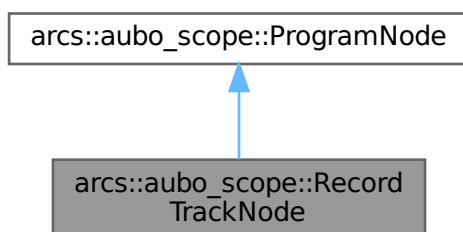
该类的文档由以下文件生成:

- `include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h`

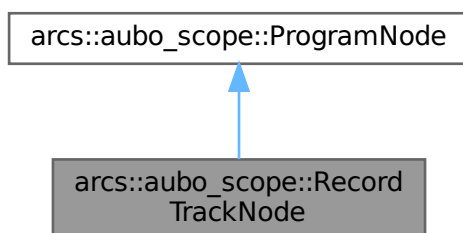
11.107 arcs::aubo_scope::RecordTrackNode 类参考

```
#include <record_track_node.h>
```

类 arcs::aubo_scope::RecordTrackNode 继承关系图:



arcs::aubo_scope::RecordTrackNode 的协作图:



Public 成员函数

- [RecordTrackNode](#) (RecordTrackNode &f)
- [RecordTrackNode](#) (RecordTrackNode &&f)
- virtual [~RecordTrackNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `Script←Writer::writeChildren`。

Private 成员函数

- [RecordTrackNode](#) ()

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数**Protected 成员函数继承自 `arcs::aubo_scope::ProgramNode`**

- `ProgramNode ()`

11.107.1 详细描述

在文件 `record_track_node.h` 第 10 行定义.

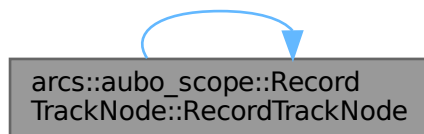
11.107.2 构造及析构函数说明**RecordTrackNode() [1/3]**

```
arcs::aubo_scope::RecordTrackNode::RecordTrackNode (
    RecordTrackNode & f)
```

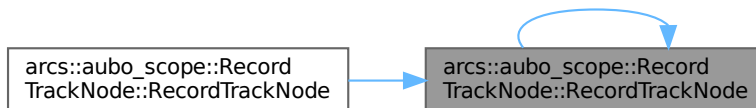
引用了 `RecordTrackNode()`.

被这些函数引用 `RecordTrackNode()` , 以及 `RecordTrackNode()`.

函数调用图:



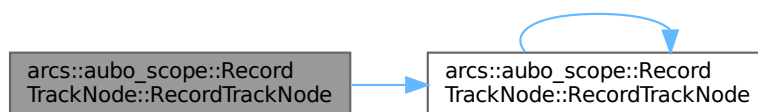
这是这个函数的调用关系图:

**RecordTrackNode() [2/3]**

```
arcs::aubo_scope::RecordTrackNode::RecordTrackNode (
    RecordTrackNode && f)
```

引用了 `RecordTrackNode()`.

函数调用图:



~RecordTrackNode()

```
virtual arcs::aubo_scope::RecordTrackNode::~~RecordTrackNode () [virtual]
```

RecordTrackNode() [3/3]

```
arcs::aubo_scope::RecordTrackNode::RecordTrackNode () [private]
```

11.107.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [record_track_node.h](#) 第 18 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.107.4 类成员变量说明

d_

```
void* arcs::aubo_scope::RecordTrackNode::d_ { nullptr } [private]
```

在文件 [record_track_node.h](#) 第 20 行定义.

该类的文档由以下文件生成:

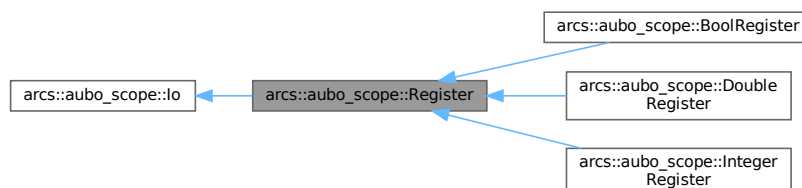
- [include/aubo_caps/domain/program/nodes/builtin/record_track_node.h](#)

11.108 arcs::aubo_scope::Register 类参考

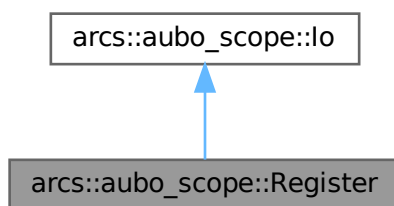
This interface provides support for registers.

```
#include <register.h>
```

类 arcs::aubo_scope::Register 继承关系图:



arcs::aubo_scope::Register 的协作图:



Public 成员函数

- virtual `~Register()` = default

Public 成员函数继承自 `arcs::aubo_scope::Io`

- `Io` (`Io &f`)
- `Io` (`Io &&f`)
- virtual `~Io()`
- `std::string` `getDisplayname()` const
获取当前安装中 I/O 的用户自定义名称
- `std::string` `getDefaultName()` const
获取 I/O 的机器人默认名称
- `std::string` `getValueStr()` const
获取 I/O 的当前读数值 (字符串形式)
- `IoType` `getType()` const
获取 I/O 类型
- `InterfaceType` `getInterfaceType()` const
获取此 I/O 所属的接口类型
- `int` `getAction()` const
获取 I/O 的动作编号
- `bool` `isInput()` const
判断 I/O 是否支持读取值
- `bool` `isResolvable()` const
某些 I/O (如 MODBUS I/O 和模拟工具输入) 在某些情况下可能不可用。此方法用于判断 I/O 是否可用。
- `size_t` `hashCode()`

额外继承的成员函数

Public 类型继承自 `arcs::aubo_scope::Io`

- `enum IoType` : `int` {
 DIGITAL, ANALOG, INTEGER, BOOLEAN,
 DOUBLE }
- `enum InterfaceType` : `int` {
 STANDARD, TOOL, CONFIGURABLE, LINKIO,
 MODBUS, GENERAL_PURPOSE, PROFINET, ETHERNET_IP }

Protected 成员函数继承自 `arcs::aubo_scope::Io`

- `Io()`

11.108.1 详细描述

This interface provides support for registers.
在文件 [register.h](#) 第 14 行定义.

11.108.2 构造及析构造函数说明

`~Register()`

```
virtual arcs::aubo_scope::Register::~Register () [virtual], [default]
```

该类的文档由以下文件生成:

- [include/aubo_caps/domain/io/register.h](#)

11.109 arcs::aubo_scope::RegisteredCapabilities 类参考

```
#include <registered_cpabilities.h>
```

Public 成员函数

- virtual `~RegisteredCapabilities ()=default`
- virtual `WidthCapabilityPtr getWidthCapability ()=0`
- virtual `GripForceCapabilityPtr getGripForceCapability ()=0`
- virtual `SpeedCapabilityPtr getSpeedCapability ()=0`
- virtual `GripVacuumCapabilityPtr getGripVacuumCapability ()=0`

11.109.1 详细描述

Each of the individual grippers/zones available in a multi-gripper device supports all capabilities that has been registered, i.e. all the individual grippers have the exact same set of capabilities. This interface gives access to all registered capabilities, such as the width capability (see [WidthCapability](#)), speed capability (see [SpeedCapability](#)), etc. This enables dynamic adjustment of capability properties, including the value range and default values, exclusively per individual gripper (independently of the other grippers).

This interface is only relevant if the multi-gripper capability has been registered (using the method `double, double, double, Length.Unit`)).

Each capability **must** be registered using the `GripperCapabilities` interface (when `GripperContribution#configure` is called) before they can be accessed using this interface.

NOTE: If an update of the properties of a capability needs to apply for all individual grippers, it is more convenient to perform the adjustment simultaneously for all grippers using the capability instance (e.g. [WidthCapability](#)) returned when the capability was registered (with the `GripperCapabilities` interface).

在文件 [registered_cpabilities.h](#) 第 49 行定义.

11.109.2 构造及析构造函数说明

`~RegisteredCapabilities()`

```
virtual arcs::aubo_scope::RegisteredCapabilities::~RegisteredCapabilities () [virtual], [default]
```

11.109.3 成员函数说明

`getGripForceCapability()`

```
virtual GripForceCapabilityPtr arcs::aubo_scope::RegisteredCapabilities::getGripForceCapability () [pure virtual]
```

Access the grip force capability interface which allows for dynamic adjustments of the capability properties exclusively for this specific individual gripper (independently of the other grippers), including the value range and default value.

This is only valid, if the force capability has been registered (using the method `double, double, Force.Unit`)).

返回

The grip force capability for this individual gripper

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the force capability

getGripVacuumCapability()

```
virtual GripVacuumCapabilityPtr arcs::aubo_scope::RegisteredCapabilities::getGripVacuumCapability () [pure virtual]
```

Access the grip vacuum capability interface which allows for dynamic adjustments of the capability properties exclusively for this specific individual gripper (independently of the other grippers), including the value range and default value.

This is only valid, if the vacuum capability has been registered (using the method `double, double, Pressure.Unit`).

返回

The grip vacuum capability for this individual gripper

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the vacuum capability

getSpeedCapability()

```
virtual SpeedCapabilityPtr arcs::aubo_scope::RegisteredCapabilities::getSpeedCapability () [pure virtual]
```

Access the speed capability interface which allows for dynamic adjustments of the capability properties exclusively for this specific individual gripper (independently of the other grippers), including the value range and default value.

This is only valid, if the speed capability has been registered (using the method `double, double, double, Speed.Unit`).

返回

The speed capability for this individual gripper

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the speed capability

getWidthCapability()

```
virtual WidthCapabilityPtr arcs::aubo_scope::RegisteredCapabilities::getWidthCapability () [pure virtual]
```

Access the width capability interface which allows for dynamic adjustments of the capability properties exclusively for this specific individual gripper (independently of the other grippers), including the value range and default value.

This is only valid, if the width capability has been registered (using the method `double, double, double, Length.Uni`).

返回

The width capability for this individual gripper

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the width capability

该类的文档由以下文件生成:

- include/aubo_caps/contribution/driver/gripper/capability/[registered_cpabilities.h](#)

11.110 arcs::aubo_scope::ReleaseActionParameters 类参考

```
#include <release_action_parameters.h>
```

Public 成员函数

- virtual [~ReleaseActionParameters](#) ()=default
- virtual double [getWidth](#) ()=0
- virtual double [getSpeed](#) ()=0
- virtual bool [isReleaseDetectionEnabled](#) ()=0

11.110.1 详细描述

This interface represents parameters for a release action defined/configured by the end user.

These parameters are used when generating script code for a release action (see [GripperContribution#generateReleaseAction](#) in file [release_action_parameters.h](#) 第 21 行定义).

11.110.2 构造及析构函数说明

[~ReleaseActionParameters\(\)](#)

```
virtual arcs::aubo_scope::ReleaseActionParameters::~ReleaseActionParameters () [virtual], [default]
```

11.110.3 成员函数说明

[getSpeed\(\)](#)

```
virtual double arcs::aubo_scope::ReleaseActionParameters::getSpeed () [pure virtual]
```

Get the selected speed to be used for releasing the gripper.

This is only valid, if the speed capability has been registered (using the method [double, double, double, Speed.Uni](#)

返回

The speed to use for the release action

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the speed capability

[getWidth\(\)](#)

```
virtual double arcs::aubo_scope::ReleaseActionParameters::getWidth () [pure virtual]
```

Get the selected width to be used for releasing the gripper.

This is only valid, if the width capability has been registered (using the method [double, double, double, Length.Uni](#)

返回

The width to use for the release action

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the width capability

isReleaseDetectionEnabled()

```
virtual bool arcs::aubo_scope::ReleaseActionParameters::isReleaseDetectionEnabled () [pure virtual]
```

Use this method to determine whether the release detection option in the Gripper program node is on or off.

Note that this method will report that release detection is off, when script code is to be generated due to the user testing a release configuration of a gripper program or operating the gripper using the toolbar.

This method should only be called, if the release detected feedback capability has been registered (using `GripperFeedbackCapabilities#registerReleaseDetectedCapability(ScriptCodeGenerator)`).

The purpose of this method is to ensure that the appropriate script code for performing a release action is generated when the end user has specified a new payload value. The user-defined payload value will be applied by AuboScope immediately after the generated release action script code has finished executing. Depending on the enablement state of release detection, the generated script should have the following properties:

- *Release detection off*: The script code should not finish earlier than when it is appropriate to apply the new payload value (i.e. when the object has been released). This typically means that the script code should wait a time period corresponding to opening the gripper's "fingers" fully (if the gripper is vacuum operated then long enough to achieve some level of vacuum).
- *Release detection on*: The payload will only be set if the release detection was successful, i.e. it will be applied immediately after a release was detected. This means that it is not necessary for the script code to wait (since the built-in detection timeout functionality will handle the waiting part).

返回

true if the end user has enabled release detection, and **false** if release detection is disabled.

异常

<i>UnsupportedOperationException</i>	if the gripper has not registered the release detected feedback capability

该类的文档由以下文件生成:

- include/aubo_caps/contribution/driver/gripper/[release_action_parameters.h](#)

11.111 arcs::aubo_scope::RenderInterface 类参考

渲染接口提供 3D 渲染相关的显示选项控制, 包括地面、机器人模型、坐标系、轨迹和点等元素的显示/隐藏。

```
#include <render_interface.h>
```

Public 类型

- enum [ShowOptions](#) : uint16_t {
[None](#) = 0 , [Ground](#) = 0x01 , [TargetRobot](#) = 0x02 , [RealRobot](#) = 0x04 ,
[Coordinate](#) = 0x08 , [Track](#) = 0x10 , [Point](#) = 0x20 , [View](#) = 0x40 ,
[Obstacle](#) = 0x80 , [SafetyPlane](#) = 0x0100 , [ToolPosition](#) = 0x0200 , [PointCloud](#) = 0x0400 ,
[All](#) = 0xFFFF }

Public 成员函数

- [RenderInterface](#) (RenderInterface &f)
- [RenderInterface](#) (RenderInterface &&f)
- virtual [~RenderInterface](#) ()
- void [setRenderEmbedded](#) (bool is_embedded, QGridLayout *gly)
设置 *Render* 窗口嵌入/移除 (*isEmbedded*) 布局 *gly* 中
- void [setShowOptions](#) (uint16_t options)
设置显示模式, 控制各种元素的显隐
- void [addPointCloud](#) (const std::string &name, const std::vector< std::vector< double > > &points, const std::vector< double > &color)
添加/修改点云
- void [removePointCloud](#) (const std::string &name)
移除指定名称的点云
- bool [hasPointCloud](#) (const std::string &name)
查找指定名称的点云是否存在
- void [clearPointCloud](#) ()
移除所有的点云
- bool [addWaypoint](#) (void *wp, const std::vector< double > &pose, const std::vector< double > &color={ 0.0, 1.0, 0.0, 1.0 }, bool axis_visible=false)
添加/修改路点
- void [removeWaypoint](#) (void *wp)
移除指定路点
- bool [hasWaypoint](#) (void *wp)
查找指定路点是否存在
- void [clearWaypoints](#) ()
移除所有的路点
- void [setMoveTrack](#) (void *wp, const std::vector< std::vector< double > > &track)
添加/修改指定路点的交融轨迹
- void [setBlendTrack](#) (void *wp, const std::vector< std::vector< double > > &track)
- void [setMoveTrack](#) (void *wp, const std::vector< std::vector< double > > &track, const std::vector< double > &color)
- void [setBlendTrack](#) (void *wp, const std::vector< std::vector< double > > &track, const std::vector< double > &color)
- void [addCoordinate](#) (const std::string &name, const std::vector< double > &pose)
添加/修改坐标轴
- void [removeCoordinate](#) (const std::string &name)
移除指定坐标轴
- bool [hasCoordinate](#) (const std::string &name)
查找指定坐标轴是否存在
- void [addLine](#) (const std::string &name, const std::vector< std::vector< double > > &points, const std::vector< double > &color)
添加/修改线
- void [addLine](#) (const std::string &name, int link_num, const std::vector< double > &pose, const std::vector< std::vector< double > > &points, const std::vector< double > &color)
- void [removeLine](#) (const std::string &name)
移除指定线
- void [clearLines](#) ()
移除所有的线
- void [addPlane](#) (const std::string &name, const std::vector< double > &pose, const std::vector< double > &size, const std::vector< double > &color)
添加/修改平面

- void `addPlane` (const std::string &name, int link_num, const std::vector< double > &pose, const std::vector< double > &size, const std::vector< double > &color)
- void `removePlane` (const std::string &name)
移除指定平面
- void `clearPlanes` ()
移除所有的平面
- void `addCube` (const std::string &name, const std::vector< double > &pose, const std::vector< double > &size, const std::vector< double > &color, int type=0)
添加/修改立方体
- void `addCube` (const std::string &name, int link_num, const std::vector< double > &pose, const std::vector< double > &size, const std::vector< double > &color, int type=0)
- void `addCube` (const std::string &name, const std::vector< double > &pose, float length_min, float length_max, float width_min, float width_max, float height_min, float height_max, const std::vector< double > &color, int type=0)
添加/修改立方体 (带尺寸范围)
- void `addCube` (const std::string &name, int link_num, const std::vector< double > &pose, float length_min, float length_max, float width_min, float width_max, float height_min, float height_max, const std::vector< double > &color, int type=0)
- void `removeCube` (const std::string &name)
移除指定立方体
- void `clearCubes` ()
移除所有的立方体
- int `addMesh` (const std::string &name, int link_num, const std::string &model_path, const std::vector< double > &pose)
addMesh 添加任意模型文件
- std::vector< std::string > `getMeshes` (int link_num)
getMeshes 获取关节 *link_num* 的外设模型名字
- int `removeMesh` (const std::string &name)
removeMesh
- int `clearMeshes` (int link_num)
clearMeshes

Private 成员函数

- `RenderInterface` ()

Private 属性

- void * `d_` { nullptr }

友元

- class `DataSwitch`

11.111.1 详细描述

渲染接口提供 3D 渲染相关的显示选项控制, 包括地面、机器人模型、坐标系、轨迹和点等元素的显示/隐藏。

在文件 `render_interface.h` 第 28 行定义。

11.111.2 成员枚举类型说明

ShowOptions

```
enum arcs::aubo_scope::RenderInterface::ShowOptions : uint16_t
```

枚举值

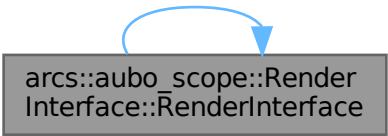
None	
Ground	
TargetRobot	
RealRobot	
Coordinate	
Track	
Point	
View	
Obstacle	
SafetyPlane	
ToolPosition	
PointCloud	
All	

在文件 [render_interface.h](#) 第 31 行定义.

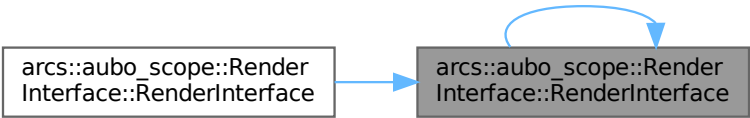
11.111.3 构造及析构函数说明

RenderInterface() [1/3]

```
arcs::aubo_scope::RenderInterface::RenderInterface (  
    RenderInterface & f)  
引用了 RenderInterface\(\).  
被这些函数引用 RenderInterface\(\) , 以及 RenderInterface\(\).  
函数调用图:
```



这是这个函数的调用关系图:

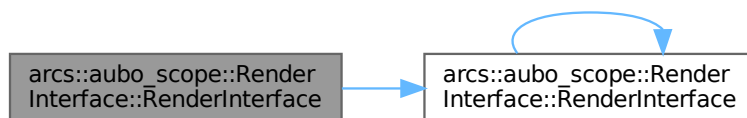


RenderInterface() [2/3]

```
arcs::aubo_scope::RenderInterface::RenderInterface (
    RenderInterface && f)
```

引用了 [RenderInterface\(\)](#).

函数调用图:

**~RenderInterface()**

```
virtual arcs::aubo_scope::RenderInterface::~~RenderInterface () [virtual]
```

RenderInterface() [3/3]

```
arcs::aubo_scope::RenderInterface::RenderInterface () [private]
```

11.111.4 成员函数说明**addCoordinate()**

```
void arcs::aubo_scope::RenderInterface::addCoordinate (
    const std::string & name,
    const std::vector< double > & pose)
```

添加/修改坐标轴

参数

<i>name</i>	坐标轴名称
<i>pose</i>	位姿 x,y,z,rx,ry,rz

addCube() [1/4]

```
void arcs::aubo_scope::RenderInterface::addCube (
    const std::string & name,
    const std::vector< double > & pose,
    const std::vector< double > & size,
    const std::vector< double > & color,
    int type = 0)
```

添加/修改立方体

参数

<i>name</i>	立方体名称
<i>link_num</i>	-1: TCP; 0: 底座; 1~6: 关节;

<i>pose</i>	位置
<i>size</i>	立方体大小（长，宽，高）
<i>color</i>	立方体颜色 {r, g, b, a}, 范围是 0.0~1.0
<i>type</i>	立方体绘制类型 0 代表画 6 个平面，1 代表只画边框线

addCube() [2/4]

```
void arcs::aubo_scope::RenderInterface::addCube (
    const std::string & name,
    const std::vector< double > & pose,
    float length_min,
    float length_max,
    float width_min,
    float width_max,
    float height_min,
    float height_max,
    const std::vector< double > & color,
    int type = 0)
```

添加/修改立方体（带尺寸范围）

参数

<i>name</i>	立方体名称
<i>pose</i>	位置
<i>length_min</i>	最小长度
<i>length_max</i>	最大长度
<i>width_min</i>	最小宽度
<i>width_max</i>	最大宽度
<i>height_min</i>	最小高度
<i>height_max</i>	最大高度
<i>color</i>	立方体颜色 {r, g, b, a}, 范围是 0.0~1.0
<i>type</i>	立方体绘制类型 0 代表画 6 个平面，1 代表只画边框线

addCube() [3/4]

```
void arcs::aubo_scope::RenderInterface::addCube (
    const std::string & name,
    int link_num,
    const std::vector< double > & pose,
    const std::vector< double > & size,
    const std::vector< double > & color,
    int type = 0)
```

addCube 添加立方体

参数

<i>name</i>	立方体名称

<i>link_num</i>	关节编号
<i>pose</i>	位姿
<i>size</i>	大小
<i>color</i>	颜色
<i>type</i>	类型

addCube() [4/4]

```
void arcs::aubo_scope::RenderInterface::addCube (
    const std::string & name,
    int link_num,
    const std::vector< double > & pose,
    float length_min,
    float length_max,
    float width_min,
    float width_max,
    float height_min,
    float height_max,
    const std::vector< double > & color,
    int type = 0)
```

addCube 添加立方体

参数

<i>name</i>	立方体名称
<i>link_num</i>	关节编号
<i>pose</i>	位姿
<i>length_min</i>	最小长度
<i>length_max</i>	最大长度
<i>width_min</i>	最小宽度
<i>width_max</i>	最大宽度
<i>height_min</i>	最小高度
<i>height_max</i>	最大高度
<i>color</i>	颜色
<i>type</i>	类型

addLine() [1/2]

```
void arcs::aubo_scope::RenderInterface::addLine (
    const std::string & name,
    const std::vector< std::vector< double > > & points,
    const std::vector< double > & color)
```

添加/修改线

参数

<i>name</i>	线名称

<i>link_num</i>	-1: TCP; 0: 底座; 1~6: 关节;
<i>pose</i>	点的参考坐标系, 默认 6 个 0
<i>points</i>	点集合
<i>color</i>	线颜色 {r, g, b, a}, 范围是 0.0~1.0

addLine() [2/2]

```
void arcs::aubo_scope::RenderInterface::addLine (
    const std::string & name,
    int link_num,
    const std::vector< double > & pose,
    const std::vector< std::vector< double > > & points,
    const std::vector< double > & color)
addLine 添加线
```

参数

<i>name</i>	线名称
<i>link_num</i>	关节编号
<i>pose</i>	位姿
<i>points</i>	点集合
<i>color</i>	颜色

addMesh()

```
int arcs::aubo_scope::RenderInterface::addMesh (
    const std::string & name,
    int link_num,
    const std::string & model_path,
    const std::vector< double > & pose)
addMesh 添加任意模型文件
```

参数

<i>name</i>	唯一标识
<i>link_num</i>	-1: TCP; 0: 底座; 1~6: 关节;
<i>model_path</i>	3ds 文件绝对路径
<i>pose</i>	模型文件安装位置 {x, y, z, rx, ry, rz}

返回

0: 成功; 其他失败

addPlane() [1/2]

```
void arcs::aubo_scope::RenderInterface::addPlane (
    const std::string & name,
    const std::vector< double > & pose,
```

```
const std::vector< double > & size,
const std::vector< double > & color)
```

添加/修改平面

参数

<i>name</i>	平面名称
<i>link_num</i>	-1: TCP; 0: 底座; 1~6: 关节;
<i>pose</i>	位置
<i>size</i>	平面大小 (长, 宽)
<i>color</i>	线颜色 {r, g, b, a}, 范围是 0.0~1.0

addPlane() [2/2]

```
void arcs::aubo_scope::RenderInterface::addPlane (
    const std::string & name,
    int link_num,
    const std::vector< double > & pose,
    const std::vector< double > & size,
    const std::vector< double > & color)
```

addPlane 添加平面

参数

<i>name</i>	平面名称
<i>link_num</i>	关节编号
<i>pose</i>	位姿
<i>size</i>	大小
<i>color</i>	颜色

addPointCloud()

```
void arcs::aubo_scope::RenderInterface::addPointCloud (
    const std::string & name,
    const std::vector< std::vector< double > > & points,
    const std::vector< double > & color)
```

添加/修改点云

参数

<i>name</i>	点云名称
<i>points</i>	点集合
<i>color</i>	点云颜色 rgba

addWaypoint()

```
bool arcs::aubo_scope::RenderInterface::addWaypoint (
    void * wp,
```

```
const std::vector< double > & pose,
const std::vector< double > & color = { 0.0, 1.0, 0.0, 1.0 },
bool axis_visible = false)
添加/修改路点
```

参数

<i>wp</i>	代表路点的唯一对象
<i>pose</i>	路点位姿 x,y,z,rx,ry,rz
<i>color</i>	路点颜色 {r, g, b, a}, 范围是 0.0~1.0
<i>axis_visible</i>	路点上是否显示位姿方向箭头

返回值

<i>true</i>	成功
<i>false</i>	失败

clearCubes()

```
void arcs::aubo_scope::RenderInterface::clearCubes ()
移除所有的立方体
```

clearLines()

```
void arcs::aubo_scope::RenderInterface::clearLines ()
移除所有的线
```

clearMeshes()

```
int arcs::aubo_scope::RenderInterface::clearMeshes (
    int link_num)
clearMeshes
```

参数

<i>link_num</i>	

返回

clearPlanes()

```
void arcs::aubo_scope::RenderInterface::clearPlanes ()
移除所有的平面
```

clearPointCloud()

```
void arcs::aubo_scope::RenderInterface::clearPointCloud ()
移除所有的点云
```

clearWaypoints()

```
void arcs::aubo_scope::RenderInterface::clearWaypoints ()
```

移除所有的路点

getMeshes()

```
std::vector< std::string > arcs::aubo_scope::RenderInterface::getMeshes (
    int link_num)
getMeshes 获取关节 link_num 的外设模型名字
```

参数

<i>link_num</i>	-1: TCP; 0: 底座; 1~6: 关节;

返回

名字列表

hasCoordinate()

```
bool arcs::aubo_scope::RenderInterface::hasCoordinate (
    const std::string & name)
查找指定坐标轴是否存在
```

参数

<i>name</i>	坐标轴名称

返回值

<i>true</i>	存在
<i>false</i>	不存在

hasPointCloud()

```
bool arcs::aubo_scope::RenderInterface::hasPointCloud (
    const std::string & name)
查找指定名称的点云是否存在
```

参数

<i>name</i>	点云名称

返回值

<i>true</i>	存在

<i>false</i>	不存在

hasWaypoint()

```
bool arcs::aubo_scope::RenderInterface::hasWaypoint (  
    void * wp)  
查找指定路点是否存在
```

参数

<i>wp</i>	代表路点的唯一对象

返回值

<i>true</i>	存在
<i>false</i>	不存在

removeCoordinate()

```
void arcs::aubo_scope::RenderInterface::removeCoordinate (  
    const std::string & name)  
移除指定坐标轴
```

参数

<i>name</i>	坐标轴名称

removeCube()

```
void arcs::aubo_scope::RenderInterface::removeCube (  
    const std::string & name)  
移除指定立方体
```

参数

<i>name</i>	立方体名称

removeLine()

```
void arcs::aubo_scope::RenderInterface::removeLine (  
    const std::string & name)  
移除指定线
```

参数

<i>name</i>	线名称

removeMesh()

```
int arcs::aubo_scope::RenderInterface::removeMesh (  
    const std::string & name)  
removeMesh
```

参数

<i>name</i>	

返回

removePlane()

```
void arcs::aubo_scope::RenderInterface::removePlane (  
    const std::string & name)  
移除指定平面
```

参数

<i>name</i>	名称

removePointCloud()

```
void arcs::aubo_scope::RenderInterface::removePointCloud (  
    const std::string & name)  
移除指定名称的点云
```

参数

<i>name</i>	点云名称

removeWaypoint()

```
void arcs::aubo_scope::RenderInterface::removeWaypoint (  
    void * wp)  
移除指定路点
```

参数

<i>wp</i>	代表路点的唯一对象

setBlendTrack() [1/2]

```
void arcs::aubo_scope::RenderInterface::setBlendTrack (
    void * wp,
    const std::vector< std::vector< double > > & track)
setBlendTrack 设置混合轨迹
```

参数

<i>wp</i>	路点对象
<i>track</i>	轨迹集合

setBlendTrack() [2/2]

```
void arcs::aubo_scope::RenderInterface::setBlendTrack (
    void * wp,
    const std::vector< std::vector< double > > & track,
    const std::vector< double > & color)
setBlendTrack 设置混合轨迹
```

参数

<i>wp</i>	路点对象
<i>track</i>	轨迹集合
<i>color</i>	颜色

setMoveTrack() [1/2]

```
void arcs::aubo_scope::RenderInterface::setMoveTrack (
    void * wp,
    const std::vector< std::vector< double > > & track)
添加/修改指定路点的交融轨迹
```

参数

<i>wp</i>	代表路点的唯一对象
<i>track</i>	路点位姿 x,y,z,rx,ry,rz 集合

setMoveTrack() [2/2]

```
void arcs::aubo_scope::RenderInterface::setMoveTrack (
    void * wp,
    const std::vector< std::vector< double > > & track,
    const std::vector< double > & color)
setMoveTrack 设置移动轨迹
```

参数

<i>wp</i>	路点对象

<i>track</i>	轨迹集合
<i>color</i>	颜色

setRenderEmbedded()

```
void arcs::aubo_scope::RenderInterface::setRenderEmbedded (
    bool is_embedded,
    QGridLayout * gly)
```

设置 Render 窗口嵌入/移除 (isEmbedded) 布局 gly 中

参数

<i>is_embedded</i>	是否嵌入
<i>gly</i>	布局

setShowOptions()

```
void arcs::aubo_scope::RenderInterface::setShowOptions (
    uint16_t options)
```

设置显示模式，控制各种元素的显隐

参数

<i>options</i>	选项值，详情查看 ShowOptions 枚举

11.111.5 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [render_interface.h](#) 第 682 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.111.6 类成员变量说明

d_

```
void* arcs::aubo_scope::RenderInterface::d_ { nullptr } [private]
```

在文件 [render_interface.h](#) 第 684 行定义.

该类的文档由以下文件生成:

- include/aubo_caps/domain/user_interface/[render_interface.h](#)

11.112 arcs::aubo_scope::ScrewdriverApiProvider 类参考

螺丝刀 API 提供者此接口提供对 AuboScope 中与螺丝刀贡献相关的功能和服务的访问。

```
#include <screw_driver_api_provider.h>
```


Public 成员函数

- virtual [~ScrewdriverApiProvider](#) ()=default
- virtual SystemApiPtr [getSystemAPI](#) ()=0
- virtual IoModelPtr [getIOModel](#) ()=0

11.112.1 详细描述

螺丝刀 API 提供者此接口提供对 AuboScope 中与螺丝刀贡献相关的功能和服务的访问。
在文件 [screw_driver_api_provider.h](#) 第 22 行定义。

11.112.2 构造及析构函数说明

[~ScrewdriverApiProvider\(\)](#)

```
virtual arcs::aubo_scope::ScrewdriverApiProvider::~~ScrewdriverApiProvider () [virtual], [default]
```

11.112.3 成员函数说明

[getIOModel\(\)](#)

```
virtual IoModelPtr arcs::aubo_scope::ScrewdriverApiProvider::getIOModel () [pure virtual]
```

返回

I/O 模型接口（例如用于获取在设置螺丝刀的自定义用户输入中显示的 I/O）

[getSystemAPI\(\)](#)

```
virtual SystemApiPtr arcs::aubo_scope::ScrewdriverApiProvider::getSystemAPI () [pure virtual]
```

返回

提供系统相关功能访问的接口

该类的文档由以下文件生成:

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_api_provider.h](#)

11.113 arcs::aubo_scope::ScrewdriverConfiguration 类参考

This interface provides access to register or setup various properties and capabilities of a screwdriver.
`#include <screw_driver_configuration.h>`

Public 成员函数

- virtual [~ScrewdriverConfiguration](#) ()=default
- virtual void [addScrewdriverProgram](#) (ScrewdriverProgramPtr screwdriverProgram)=0
Add a screwdriver program to the list of programs.
- virtual void [addScrewdriverPrograms](#) (std::vector< ScrewdriverProgramPtr > screwdriverProgram↵List)=0
Add a list of screwdriver programs to the list of programs.
- virtual void [registerProgramSelectionCapability](#) (std::function< void(ScriptWriterPtr, Program↵SelectionParametersPtr)> scriptCodeGenerator)=0
- virtual void [registerOperationTypeCapability](#) ()=0
Register an operation type capability for a screwdriver that supports/requires the parameter for the screwdriving operation type (i.e.
- virtual void [registerPrepareToStartScrewdriverCapability](#) (std::function< void(ScriptWriterPtr, PrepareToStartScrewdriverParametersPtr)> scriptCodeGenerator)=0
- virtual void [registerFeedScrewCapability](#) (std::function< void(ScriptWriterPtr, FeedScrewParameters↵Ptr)> scriptCodeGenerator)=0

- virtual void [registerDriveScrewOKCapability](#) (std::function< void(ScriptWriterPtr, DriveScrew←OKParametersPtr)> scriptCodeGenerator)=0
- virtual void [registerDriveScrewNotOKCapability](#) (std::function< void(ScriptWriterPtr, DriveScrew←NotOKParametersPtr)> scriptCodeGenerator)=0
- virtual void [registerScrewdriverReadyCapability](#) (std::function< void(ScriptWriterPtr, Screwdriver←ReadyParametersPtr)> scriptCodeGenerator)=0

11.113.1 详细描述

This interface provides access to register or setup various properties and capabilities of a screwdriver. It is optional to register these capabilities and properties.
在文件 [screw_driver_configuration.h](#) 第 163 行定义.

11.113.2 构造及析构函数说明

~ScrewdriverConfiguration()

```
virtual arcs::aubo_scope::ScrewdriverConfiguration::~~ScrewdriverConfiguration () [virtual], [default]
```

11.113.3 成员函数说明

addScrewdriverProgram()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::addScrewdriverProgram (
    ScrewdriverProgramPtr screwdriverProgram) [pure virtual]
```

Add a screwdriver program to the list of programs.

参数

<i>screwdriverProgram</i>	program to add, not null

异常

<i>DuplicateScrewdriverProgramID</i>	if the ids of the screwdriver programs (returned by the method ScrewdriverProgram#getId()) are not unique.
<i>NameOfScrewdriverProgramCannotBeNullOrEmpty</i>	if the display name of a program (returned by the method ScrewdriverProgram#getDisplayName()) is null or an empty string.

addScrewdriverPrograms()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::addScrewdriverPrograms (
    std::vector< ScrewdriverProgramPtr > screwdriverProgramList) [pure virtual]
```

Add a list of screwdriver programs to the list of programs.

参数

<i>screwdriverProgramList</i>	list of programs to add, not null

异常

<i>DuplicateScrewdriverProgramID</i>	if the ids of the screwdriver programs (returned by the method ScrewdriverProgram#getId()) are not unique.
<i>NameOfScrewdriverProgramCannotBeNullOrEmpty</i>	if the display name of a program (returned by the method ScrewdriverProgram#getDisplayName()) is null or an empty string.

registerDriveScrewNotOKCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerDriveScrewNotOKCapability (
    std::function< void(ScriptWriterPtr, DriveScrewNotOKParametersPtr)> scriptCodeGenerator) [pure
virtual]
```

Register a feedback capability for a screwdriver that is capable of providing information about whether or not an ongoing screwdriving operation (driving the screw) failed, i.e. ended in Not Ok (NOK) state.

Note: This capability must only be used to provide feedback about the failure of the screwing operation. It should not be used to provide feedback about successes. To support feedback about when the operation succeeds, register the Drive Screw Ok feedback capability the using the `registerDriveScrewOKCapability(ScriptC` method.

The provided implementation of the `ScriptCodeGenerator` interface must generate the script code for determining if the screwing operation ended in failure. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if the screwing operation failed, otherwise 'False' must be returned.

Note: A return value of 'False' must not be used to indicate that the operation succeeded, but simply that a failure was not (yet) detected. If the screwdriver can provide feedback on whether or not the screwing operation succeeded, the Drive Screw Ok feedback capability should be used.

Example:

The digital tool input 1 will go high if the operation ended in NOT OK-state. The script code is:

```
return get_tool_digital_in(1)
```

So the implementation of `ScriptCodeGenerator#generateScript(ScriptWriter, Object)` could be:

```
scriptWriter.appendLine("return get_tool_digital_in(1)");
```

参数

<i>scriptCodeGenerator</i>	script code generator responsible for generating the script code for determining if the screwing operation failed. When the script code needs to be generated, the <code>ScriptCodeGenerator#generateScript(ScriptWriter, Object)</code> method will be called (by AuboScope). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'.

异常

<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered

<i>CalledOutsideScrewdriverConfigurationPhase</i>	if this method is called at the wrong time, i.e. outside the scope of the <code>ScrewdriverContribution#configure←Screwdriver(ScrewdriverConfiguration, ScrewdriverApiProvider)</code> method.

registerDriveScrewOKCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerDriveScrewOKCapability (
    std::function< void(ScriptWriterPtr, DriveScrewOKParametersPtr)> scriptCodeGenerator) [pure
virtual]
```

Register a feedback capability for a screwdriver that is capable of providing information about whether or not an ongoing screwdriving operation (driving the screw) ended successfully (ended in OK-state).

Note: This capability must only be used to provide feedback about the success of the screwing operation. It should not be used to provide feedback about failures. To support feedback about when the operation fails, register the Drive Screw Not Ok feedback capability the using the `registerDriveScrewNotOKCapability(ScriptWriterPtr, DriveScrewNotOKParametersPtr)` method.

The provided implementation of the `ScriptCodeGenerator` interface must generate the script code for determining if the screwing operation succeeded. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if the screwing operation was successful, otherwise 'False' must be returned.

Note: A return value of 'False' must not be used to indicate that the operation failed, but simply that a success was not (yet) detected. If the screwdriver can provide feedback on whether or not the screwing operation failed, the Drive Screw Not Ok feedback capability must be used.

Example:

The digital tool input 0 will go high if the operation ended in OK-state. The script code is:

```
return get_tool_digital_in(0)
```

So the implementation of `ScriptCodeGenerator#generateScript(ScriptWriter, Object)` could be:

```
scriptWriter.appendLine("return get_tool_digital_in(0)");
```

参数

<i>scriptCodeGenerator</i>	script code generator responsible for generating the script code for determining if the screwing operation succeeded. When the script code needs to be generated, the <code>ScriptCodeGenerator#generateScript(ScriptWriter, Object)</code> method will be called (by AuboScope). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'.

异常

<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered
<i>CalledOutsideScrewdriverConfigurationPhase</i>	if this method is called at the wrong time, i.e. outside the scope of the <code>ScrewdriverContribution#configure←Screwdriver(ScrewdriverConfiguration, ScrewdriverApiProvider)</code> method.

registerFeedScrewCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerFeedScrewCapability (
    std::function< void(ScriptWriterPtr, FeedScrewParametersPtr)> scriptCodeGenerator) [pure virtual]
```

[Register](#) a capability for a screwdriver that supports automatic screw feeding.

The provided implementation of the `ScriptCodeGenerator` interface must generate the script code for feeding a screw to the screwdriver. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if the screw feed operation was successful and 'False' must be returned if the operation failed.

The generated script code is executed before the script code for preparing the screwdriver, i.e the script code generated by `registerPrepareToStartScrewdriverCapability(ScriptCodeGenerator)`.

参数

<i>scriptCodeGenerator</i>	script code generator responsible for generating the script code for feeding a screw to the screwdriver. When the script code needs to be generated, the <code>ScriptCodeGenerator#generateScript(ScriptWriter, Object)</code> method will be called (by AuboScope). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'

异常

<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered
<i>CalledOutsideScrewdriverConfigurationPhase</i>	if this method is called at the wrong time, i.e. outside the scope of the <code>ScrewdriverContribution#configure↵Screwdriver(ScrewdriverConfiguration, ScrewdriverApiProvider)</code> method.

registerOperationTypeCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerOperationTypeCapability () [pure virtual]
```

[Register](#) an operation type capability for a screwdriver that supports/requires the parameter for the screwdriving operation type (i.e.

if the operation is Tighten or Loosen).

异常

<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered
<i>CalledOutsideScrewdriverConfigurationPhase</i>	if this method is called at the wrong time, i.e. outside the scope of the <code>ScrewdriverContribution#configure↵Screwdriver(ScrewdriverConfiguration, ScrewdriverApiProvider)</code> method.

registerPrepareToStartScrewdriverCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerPrepareToStartScrewdriverCapability (
    std::function< void(ScriptWriterPtr, PrepareToStartScrewdriverParametersPtr)> scriptCodeGenerator)
[pure virtual]
```

Register a capability/requirement for a screwdriver that needs to generate script code for preparing the screwdriver for a screwdriving operation (before starting the screw driver).

The provided implementation of the `ScriptCodeGenerator` interface must generate the script code for preparing the screwdriver. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if the preparation of the screwdriver was successful and 'False' must be returned if the preparation failed.

The generated script code is executed before the script code for starting the screwdriver (could be from a different node), i.e the script code generated by `ScrewdriverContribution#generateStart↵ScrewdriverScript(ScriptWriter, ScrewdriverParameters)`.

参数

<i>scriptCodeGenerator</i>	script code generator responsible for generating the script code for preparing the screwdriver. When the script code needs to be generated, the <code>ScriptCodeGenerator#generateScript(ScriptWriter, Object)</code> method will be called (by <code>AuboScope</code>). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'.

异常

<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered
<i>CalledOutsideScrewdriverConfigurationPhase</i>	if this method is called at the wrong time, i.e. outside the scope of the <code>ScrewdriverContribution#configure↵Screwdriver(ScrewdriverConfiguration, ScrewdriverApiProvider)</code> method.

registerProgramSelectionCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerProgramSelectionCapability (
    std::function< void(ScriptWriterPtr, ProgramSelectionParametersPtr)> scriptCodeGenerator) [pure
virtual]
```

Register a program selection capability for a screwdriver that supports selecting between a set of screwdriver programs, typically defined on an external control box for the screwdriver.

The provided implementation of the `ScriptCodeGenerator` interface must generate the script code for selecting the specified screwdriver program. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if selecting the program succeeded and 'False' must be returned if the program selection failed.

The generated script code is executed before the script code for starting the screwdriver (could be from a different node), i.e the script code generated by `ScrewdriverContribution#generateStart↵ScrewdriverScript(ScriptWriter, ScrewdriverParameters)`.

参数

<i>programListProvider</i>	provider of the list of screwdriver programs, not null
<i>scriptCodeGenerator</i>	script code generator responsible for generating the script code for selecting the specified screwdriver program. When the script code needs to be generated, the <code>ScriptCodeGenerator#generateScript(ScriptWriter, Object)</code> method will be called (by <code>AuboScope</code>). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'.

异常

<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered
<i>CalledOutsideScrewdriverConfigurationPhase</i>	if this method is called at the wrong time, i.e. outside the scope of the <code>ScrewdriverContribution#configure←Screwdriver(ScrewdriverConfiguration, ScrewdriverApiProvider)</code> method.

registerScrewdriverReadyCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerScrewdriverReadyCapability (
    std::function< void(ScriptWriterPtr, ScrewdriverReadyParametersPtr)> scriptCodeGenerator) [pure
virtual]
```

[Register](#) a feedback capability for a screwdriver that is capable of providing information about whether or not the screwdriver device is ready to operate (receive commands). This capability is used (by Aubo←Scope) in a program to ensure that the screwdriver is ready before sending commands to it.

The provided implementation of the `ScriptCodeGenerator` interface must generate the script code for determining if the screwdriver is ready. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if the screwdriver is ready, otherwise 'False' must be returned.

The generated script code is executed before the script code for the program selected capability (if that capability has been registered).

参数

<i>scriptCodeGenerator</i>	script code generator responsible for generating the script code for determining if the screwdriver is ready. When the script code needs to be generated, the <code>ScriptCodeGenerator#generateScript(ScriptWriter, Object)</code> method will be called (by AuboScope). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'.

异常

<i>CapabilityAlreadyRegistered</i>	if this capability has already been registered
<i>CalledOutsideScrewdriverConfigurationPhase</i>	if this method is called at the wrong time, i.e. outside the scope of the <code>ScrewdriverContribution#configure←Screwdriver(ScrewdriverConfiguration, ScrewdriverApiProvider)</code> method.

该类的文档由以下文件生成:

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_configuration.h](#)

11.114 arcs::aubo_scope::ScrewdriverContribution 类参考

螺丝刀贡献此接口定义了向 AuboScope 添加螺丝刀功能的 API。

```
#include <screw_driver_contribution.h>
```


Public 成员函数

- virtual `~ScrewdriverContribution()` = default
- virtual `std::string getTitle()` = 0
此方法必须返回螺丝刀贡献的标题。标题显示在以下位置:
- virtual void `configureScrewdriver(ScrewdriverConfiguration screwdriverConfiguration, ScrewdriverAPIProviderPtr apiProvider)`
- virtual void `generateStartScrewdriverScript(ScriptWriterPtr scriptWriter, ScrewdriverParametersPtr parameters)` = 0
- virtual void `generateStopScrewdriverScript(ScriptWriterPtr scriptWriter, ScrewdriverParametersPtr parameters)` = 0

11.114.1 详细描述

螺丝刀贡献此接口定义了向 AuboScope 添加螺丝刀功能的 API。

实现此接口将把螺丝刀贡献”集成”（挂钩）到内置（原生）AuboScope 螺丝刀程序节点和螺丝刀安装画面中。这将向终端用户暴露螺丝刀支持的螺丝驱动功能。

注意：内置（原生）AuboScope 螺丝刀程序节点和螺丝刀贡献在 CB3 机器人上均不受支持。如果 AuboCap 在 CB3 机器人上注册了螺丝刀贡献，AuboCap 将无法激活。

螺丝刀程序节点允许终端用户编程螺丝驱动操作。终端用户可用的参数基于螺丝刀支持的属性和功能。至少，所有螺丝刀必须支持”默认”拧紧（启动螺丝刀）和停止（停止螺丝刀）操作。所有其他功能是可选的，可以在调用 `ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration, ScrewdriverAPIProvider)` 方法时注册。

螺丝刀安装画面中的集成允许用户选择螺丝刀来进行拧紧操作编程（使用螺丝刀程序节点）。它还提供了定义终端用户所需的任何自定义输入来设置螺丝刀的可能性，例如指定 IP 地址。此功能是可选的，因为此类配置属性是特定于每个螺丝刀设备的。相应的生成自定义 UI 将在用户选择螺丝刀时在画面中可访问，并替换默认的螺丝刀安装 UI。当调用 `ScrewdriverContribution#configureInstallation(CustomUserInputConfiguration, SystemConfiguration, TCPConfiguration, ScrewdriverAPIProvider)` 方法时，提供创建基于用户输入的自定义配置的访问权限。

在文件 `screw_driver_contribution.h` 第 78 行定义。

11.114.2 构造及析构函数说明

`~ScrewdriverContribution()`

```
virtual arcs::aubo_scope::ScrewdriverContribution::~~ScrewdriverContribution() [virtual], [default]
```

11.114.3 成员函数说明

`configureScrewdriver()`

```
virtual void arcs::aubo_scope::ScrewdriverContribution::configureScrewdriver (
    ScrewdriverConfiguration screwdriverConfiguration,
    ScrewdriverAPIProviderPtr apiProvider) [virtual]
```

当调用此方法时，使用 `configuration` 参数注册或设置螺丝刀的可选属性和功能。

此方法在螺丝刀配置阶段调用，在贡献注册后、新安装被加载或创建时调用。

注意：如果螺丝刀仅支持基本的”默认”拧紧和停止操作，请将此方法的实现留空。

参数

<i>screwdriverConfiguration</i>	可用于注册或设置螺丝刀属性和功能的配置实例
<i>apiProvider</i>	提供对 AuboScope 中可用于设置螺丝刀功能的相关功能和服务访问

generateStartScrewdriverScript()

```
virtual void arcs::aubo_scope::ScrewdriverContribution::generateStartScrewdriverScript (
    ScriptWriterPtr scriptWriter,
    ScrewdriverParametersPtr parameters) [pure virtual]
```

当调用此方法时，必须生成驱动螺丝的脚本代码。

终端用户定义/配置的已注册可选螺丝刀功能/要求的参数将作为脚本代码生成的输入。

参数

<i>scriptWriter</i>	使用此脚本编写器实例生成驱动螺丝的脚本代码
<i>parameters</i>	终端用户定义/配置的螺丝驱动操作参数

generateStopScrewdriverScript()

```
virtual void arcs::aubo_scope::ScrewdriverContribution::generateStopScrewdriverScript (
    ScriptWriterPtr scriptWriter,
    ScrewdriverParametersPtr parameters) [pure virtual]
```

当调用此方法时，必须生成停止螺丝刀的脚本代码（即螺丝刀必须停止当前的螺丝驱动操作）。

终端用户定义/配置的已注册可选螺丝刀功能/要求的参数将作为脚本代码生成的输入。

参数

<i>scriptWriter</i>	使用此脚本编写器实例生成停止螺丝刀的脚本代码
<i>parameters</i>	终端用户定义/配置的螺丝驱动操作参数

getTitle()

```
virtual std::string arcs::aubo_scope::ScrewdriverContribution::getTitle () [pure virtual]
```

此方法必须返回螺丝刀贡献的标题。标题显示在以下位置：

- 内置 AuboScope 螺丝驱动安装画面顶部选择螺丝刀的位置
- 内置 AuboScope 螺丝驱动程序节点的画面（如果已在螺丝驱动安装画面中选择了此螺丝刀）

此方法在启动时调用一次。

参数

<i>locale</i>	AuboScope 的当前语言环境，可用于支持多种语言的标题

返回

此螺丝刀贡献的标题，不能为 `null` 也不能为空字符串

该类的文档由以下文件生成：

- `include/aubo_caps/contribution/driver/screw_driver/screw_driver_contribution.h`

11.115 arcs::aubo_scope::ScrewdriverParameters 类参考

This interface represents parameters for a screwdriver operation defined/configured by the end user.

```
#include <screw_driver_parameters.h>
```

Public 成员函数

- virtual [~ScrewdriverParameters](#) ()=default
- virtual [ScrewdriverProgram](#) [getScrewdriverProgram](#) ()=0
- virtual [ScrewdriverOperationType](#) [getScrewdriverOperationType](#) ()=0

11.115.1 详细描述

This interface represents parameters for a screwdriver operation defined/configured by the end user. 在文件 [screw_driver_parameters.h](#) 第 77 行定义.

11.115.2 构造及析构造函数说明

~ScrewdriverParameters()

```
virtual arcs::aubo_scope::ScrewdriverParameters::~~ScrewdriverParameters () [virtual], [default]
```

11.115.3 成员函数说明

getScrewdriverOperationType()

```
virtual ScrewdriverOperationType arcs::aubo_scope::ScrewdriverParameters::getScrewdriverOperationType ()  
[pure virtual]
```

Get the selected screwdriver operation type.

This is only valid, if the operation type capability has been registered (using the method [ScrewdriverCapabilities#registerOperationTypeCapability](#)).

返回

The operation type selected by the user.

异常

<i>UnsupportedOperationException</i>	if the screwdriver has not registered the operation type capability

getScrewdriverProgram()

```
virtual ScrewdriverProgram arcs::aubo_scope::ScrewdriverParameters::getScrewdriverProgram () [pure virtual]
```

Get the selected screwdriver program to be used for the screwdriver operation.

This is only valid, if the program selection capability has been registered (using the method [ScrewdriverCapabilities#registerProgramSelectionCapability](#)([ScrewdriverProgramListProvider](#), [ScriptCodeGenerator](#))).

返回

The selected screwdriver program to use for the operation.

异常

<i>UnsupportedOperationException</i>	if the screwdriver has not registered the program selection capability

该类的文档由以下文件生成:

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_parameters.h](#)

11.116 arcs::aubo_scope::ScrewdriverProgram 类参考

This abstract class represents a screwdriver program, typically defined on an external control box for the screwdriver.

```
#include <screw_driver_parameters.h>
```

Public 成员函数

- virtual [~ScrewdriverProgram](#) ()=0
- virtual std::string [getId](#) ()=0
- virtual std::string [getDisplayname](#) ()=0

11.116.1 详细描述

This abstract class represents a screwdriver program, typically defined on an external control box for the screwdriver.

在文件 [screw_driver_parameters.h](#) 第 15 行定义.

11.116.2 构造及析构造函数说明

~ScrewdriverProgram()

```
virtual arcs::aubo_scope::ScrewdriverProgram::~ScrewdriverProgram () [pure virtual]
```

11.116.3 成员函数说明

getDisplayname()

```
virtual std::string arcs::aubo_scope::ScrewdriverProgram::getDisplayname () [pure virtual]
```

This method should return the name of the screwdriver program. The name will be displayed in the UI, e.g. in a combo box. The name returned is allowed to change over versions of the driver AuboCap as well as be different depending on the current language selected in AuboScope, in case translations are supported.

This method must be implemented.

返回

the name of the screwdriver program to display in the UI.

getId()

```
virtual std::string arcs::aubo_scope::ScrewdriverProgram::getId () [pure virtual]
```

This method must return an identifier to uniquely identify this screwdriver program. The identifier will be used for persistence and resolving the correct screwdriver program. Therefore the identifier must remain constant over time and versions of the screwdriver AuboCap.

This method must be implemented.

返回

a unique identifier string which will remain constant over time and versions of the screwdriver AuboCap.

该类的文档由以下文件生成:

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_parameters.h](#)

11.117 arcs::aubo_scope::ScrewdriverReadyParameters 类参考

This interface provides the parameters relevant for generating script for the Screwdriver Ready feedback operation capability (registered using `ScrewdriverFeedbackCapabilities#registerScrewdriverReadyCapability(ScriptCodeGenerator)`).

```
#include <screw_driver_configuration.h>
```

Public 成员函数

- virtual `~ScrewdriverReadyParameters()`=default

11.117.1 详细描述

This interface provides the parameters relevant for generating script for the Screwdriver Ready feedback operation capability (registered using `ScrewdriverFeedbackCapabilities#registerScrewdriverReadyCapability(ScriptCodeGenerator)`).

These parameters are passed when AuboScope calls the implementation of the `ScriptCodeGenerator#generateScript` method responsible for the generating the script code.

The interface is intentionally empty, but could have parameters in future releases.

在文件 `screw_driver_configuration.h` 第 151 行定义.

11.117.2 构造及析构造函数说明

`~ScrewdriverReadyParameters()`

```
virtual arcs::aubo_scope::ScrewdriverReadyParameters::~~ScrewdriverReadyParameters () [virtual], [default]
```

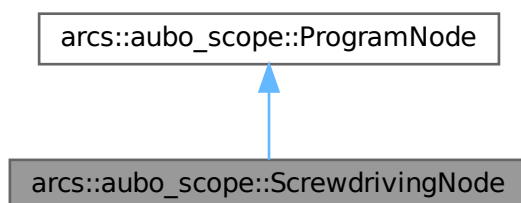
该类的文档由以下文件生成:

- `include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h`

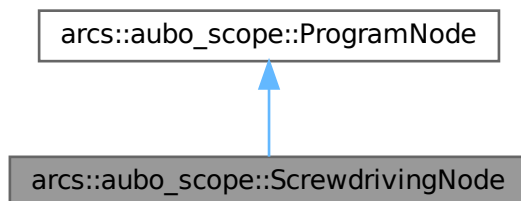
11.118 arcs::aubo_scope::ScrewdrivingNode 类参考

```
#include <screw_driving_node.h>
```

类 `arcs::aubo_scope::ScrewdrivingNode` 继承关系图:



`arcs::aubo_scope::ScrewdrivingNode` 的协作图:



Public 成员函数

- [ScrewdrivingNode](#) (ScrewdrivingNode &f)
- [ScrewdrivingNode](#) (ScrewdrivingNode &&f)
- virtual [~ScrewdrivingNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *Script←Writer::writeChildren*。

Private 成员函数

- [ScrewdrivingNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数**Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)**

- [ProgramNode](#) ()

11.118.1 详细描述

在文件 [screw_driving_node.h](#) 第 10 行定义.

11.118.2 构造及析构造函数说明**ScrewdrivingNode() [1/3]**

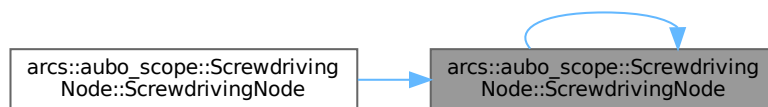
```
arcs::aubo_scope::ScrewdrivingNode::ScrewdrivingNode (
    ScrewdrivingNode & f)
```

引用了 [ScrewdrivingNode\(\)](#).

被这些函数引用 [ScrewdrivingNode\(\)](#) , 以及 [ScrewdrivingNode\(\)](#).
函数调用图:



这是这个函数的调用关系图:

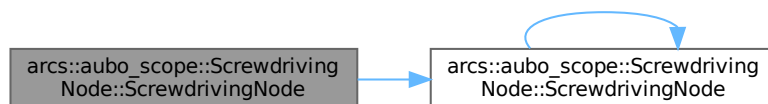


ScrewdrivingNode() [2/3]

```
arcs::aubo_scope::ScrewdrivingNode::ScrewdrivingNode (
    ScrewdrivingNode && f)
```

引用了 [ScrewdrivingNode\(\)](#).

函数调用图:



~ScrewdrivingNode()

```
virtual arcs::aubo_scope::ScrewdrivingNode::~~ScrewdrivingNode () [virtual]
```

ScrewdrivingNode() [3/3]

```
arcs::aubo_scope::ScrewdrivingNode::ScrewdrivingNode () [private]
```

11.118.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [screw_driving_node.h](#) 第 18 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.118.4 类成员变量说明

d_

```
void* arcs::aubo_scope::ScrewdrivingNode::d_ { nullptr } [private]
```

在文件 [screw_driving_node.h](#) 第 20 行定义.

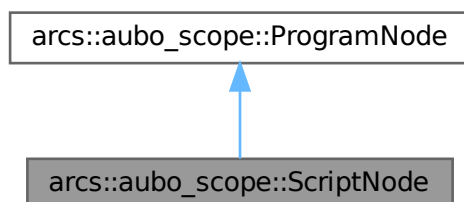
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/screw_driving_node.h](#)

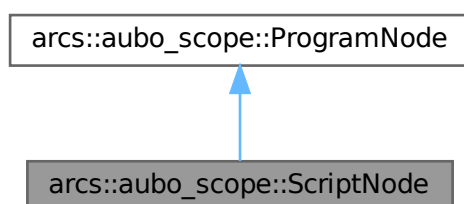
11.119 arcs::aubo_scope::ScriptNode 类参考

```
#include <script_node.h>
```

类 arcs::aubo_scope::ScriptNode 继承关系图:



arcs::aubo_scope::ScriptNode 的协作图:



Public 成员函数

- [ScriptNode](#) (ScriptNode &f)
- [ScriptNode](#) (ScriptNode &&f)
- virtual [~ScriptNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `Script←Writer::writeChildren`。

Private 成员函数

- [ScriptNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode\(\)](#)

11.119.1 详细描述

在文件 [script_node.h](#) 第 10 行定义.

11.119.2 构造及析构函数说明

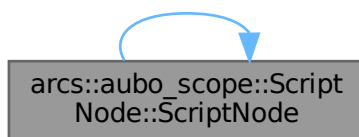
[ScriptNode\(\)](#) [1/3]

```
arcs::aubo_scope::ScriptNode::ScriptNode (
    ScriptNode & f)
```

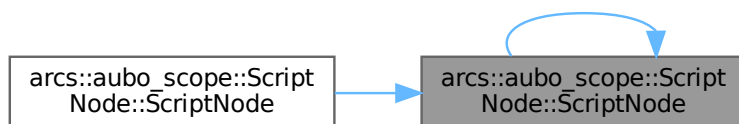
引用了 [ScriptNode\(\)](#).

被这些函数引用 [ScriptNode\(\)](#) , 以及 [ScriptNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

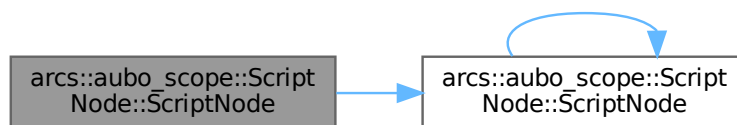


[ScriptNode\(\)](#) [2/3]

```
arcs::aubo_scope::ScriptNode::ScriptNode (
    ScriptNode && f)
```

引用了 [ScriptNode\(\)](#).

函数调用图:



~ScriptNode()

```
virtual arcs::aubo_scope::ScriptNode::~~ScriptNode () [virtual]
```

ScriptNode() [3/3]

```
arcs::aubo_scope::ScriptNode::ScriptNode () [private]
```

11.119.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [script_node.h](#) 第 18 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.119.4 类成员变量说明

d_

```
void* arcs::aubo_scope::ScriptNode::d_ { nullptr } [private]
```

在文件 [script_node.h](#) 第 20 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/script_node.h](#)

11.120 arcs::aubo_scope::ScriptWriter 类参考

脚本写入器此接口提供生成脚本代码的支持。

```
#include <script_writer.h>
```

Public 成员函数

- [ScriptWriter](#) (ScriptWriter &f)
- [ScriptWriter](#) (ScriptWriter &&f)
- virtual [~ScriptWriter](#) ()
- void [setLabel](#) (int lineno, const std::string &comment)
- void [appendLine](#) (const std::string &script_line)
使用自动缩进添加一行脚本代码。
- void [appendVectorDouble](#) (const std::vector< double > &value)
- void [appendRaw](#) (const std::string &script)
按原样追加脚本代码，不使用自动缩进。
- std::string [generateScript](#) ()

- 生成包含完整脚本代码的字符串。
- void `writeChildren` ()
 - 当为具有子节点的节点生成脚本代码时使用此方法。让子节点生成代码。
- void `assign` (VariablePtr variable, const std::string &expression, bool `sync`=false)
- void `assign` (const std::string &variableName, const ExpressionPtr &expression, bool `sync`=false)
- void `assign` (VariablePtr variable, const ExpressionPtr &expression, bool `sync`=false)
- void `incrementVariable` (const std::string &variable_name)
 - 变量值加 1。
- void `note` (const std::string &expression)
 - 添加注释。
- void `sleep` (double seconds)
 - 休眠指定秒数。
- void `sync` ()
 - 用完当前帧剩余时间。
- void `defineFunction` (const std::string &func_name)
 - 定义函数。
- void `anonyFunction` (const std::string &func_name)
- void `setRobotIndex` (int robot_index)
- void `returnMethod` ()
 - 返回。
- void `end` ()
 - 插入 *end*。
- void `lineFeed` ()
 - 插入空行。
- void `setPayload` (double mass, double x, double y, double z)
 - 设置负载质量和重心。
- void `setTcp` (const std::vector< double > &pose)
 - 设置 *TCP*。
- void `ifCondition` (const ExpressionPtr &expression)
 - 开始 *if* 条件。
- void `ifNotCondition` (const ExpressionPtr &expression)
 - 开始取反的 *if* 条件。
- void `elseifCondition` (const ExpressionPtr &expression)
 - 添加 *else-if* 分支。
- void `elseCondition` ()
 - 添加 *else* 分支。
- void `forCondition` (int count, int step)
 - 开始 *for* 循环。
- void `whileTrue` ()
 - 开始 *while true* 循环。
- void `whileCondition` (const ExpressionPtr &expression)
 - 开始 *while* 循环。
- void `whileNot` (const ExpressionPtr &expression)
 - 开始取反的 *while* 循环。
- void `defineThread` (const std::string &thread_name, bool loop_or_not)
 - 定义线程。
- void `runThread` (const std::string &thread_name)
 - 启动线程。
- void `killThread` (const std::string &thread_name)
 - 终止线程。
- std::string `getResolvedVariable` (const std::string &variable_name)
 - 获取可在脚本中使用的已注册变量名。
- void `increaseIndent` ()
- void `decreaseIndent` ()

Private 成员函数

- [ScriptWriter\(\)](#)

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.120.1 详细描述

脚本写入器此接口提供生成脚本代码的支持。
在文件 [script_writer.h](#) 第 23 行定义.

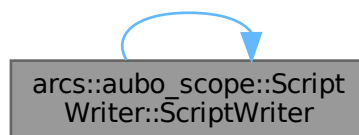
11.120.2 构造及析构函数说明**ScriptWriter()** [1/3]

```
arcs::aubo_scope::ScriptWriter::ScriptWriter (  
    ScriptWriter & f)
```

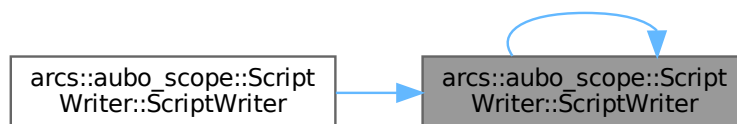
引用了 [ScriptWriter\(\)](#).

被这些函数引用 [ScriptWriter\(\)](#) , 以及 [ScriptWriter\(\)](#).

函数调用图:



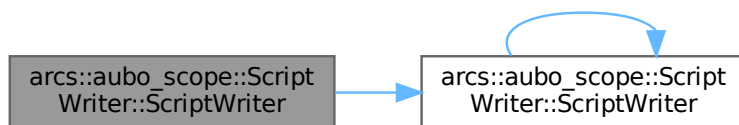
这是这个函数的调用关系图:

**ScriptWriter()** [2/3]

```
arcs::aubo_scope::ScriptWriter::ScriptWriter (  
    ScriptWriter && f)
```

引用了 [ScriptWriter\(\)](#).

函数调用图:



~ScriptWriter()

```
virtual arcs::aubo_scope::ScriptWriter::~~ScriptWriter () [virtual]
```

ScriptWriter() [3/3]

```
arcs::aubo_scope::ScriptWriter::ScriptWriter () [private]
```

11.120.3 成员函数说明

anonyFunction()

```
void arcs::aubo_scope::ScriptWriter::anonyFunction (
    const std::string & func_name)
```

appendLine()

```
void arcs::aubo_scope::ScriptWriter::appendLine (
    const std::string & script_line)
```

使用自动缩进添加一行脚本代码。

参数

<i>script_line</i>	要追加的脚本代码行。

appendRaw()

```
void arcs::aubo_scope::ScriptWriter::appendRaw (
    const std::string & script)
```

按原样追加脚本代码，不使用自动缩进。

参数

<i>script</i>	要追加的脚本代码。

appendVectorDouble()

```
void arcs::aubo_scope::ScriptWriter::appendVectorDouble (
    const std::vector< double > & value)
```

assign() [1/3]

```
void arcs::aubo_scope::ScriptWriter::assign (  
    const std::string & variableName,  
    const ExpressionPtr & expression,  
    bool sync = false)
```

变量赋值。在无 local 或 global 限定符的情况下赋值。

参数

<i>variableName</i>	变量名称。
<i>expression</i>	赋给变量的表达式。

引用了 [sync\(\)](#)。

函数调用图:

**assign()** [2/3]

```
void arcs::aubo_scope::ScriptWriter::assign (  
    VariablePtr variable,  
    const ExpressionPtr & expression,  
    bool sync = false)
```

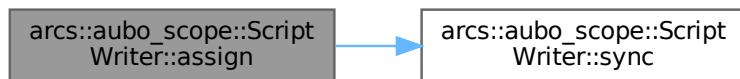
变量赋值。

参数

<i>variable</i>	变量。
<i>expression</i>	表达式。

引用了 [sync\(\)](#)。

函数调用图:



assign() [3/3]

```
void arcs::aubo_scope::ScriptWriter::assign (
    VariablePtr variable,
    const std::string & expression,
    bool sync = false)
```

引用了 [sync\(\)](#).

函数调用图:



decreaseIndent()

```
void arcs::aubo_scope::ScriptWriter::decreaseIndent ()
```

defineFunction()

```
void arcs::aubo_scope::ScriptWriter::defineFunction (
    const std::string & func_name)
```

定义函数。

defineThread()

```
void arcs::aubo_scope::ScriptWriter::defineThread (
    const std::string & thread_name,
    bool loop_or_not)
```

定义线程。

elseCondition()

```
void arcs::aubo_scope::ScriptWriter::elseCondition ()
```

添加 else 分支。

elseifCondition()

```
void arcs::aubo_scope::ScriptWriter::elseifCondition (
    const ExpressionPtr & expression)
```

添加 else-if 分支。

end()

```
void arcs::aubo_scope::ScriptWriter::end ()
```

插入 end。

forCondition()

```
void arcs::aubo_scope::ScriptWriter::forCondition (
    int count,
    int step)
```

开始 for 循环。

generateScript()

```
std::string arcs::aubo_scope::ScriptWriter::generateScript ()
```

生成包含完整脚本代码的字符串。

返回

生成的脚本字符串。

getResolvedVariable()

```
std::string arcs::aubo_scope::ScriptWriter::getResolvedVariable (
    const std::string & variable_name)
```

获取可在脚本中使用的已注册变量名。

参数

<i>variable_name</i>	变量名称。

返回

可在脚本中使用的变量名。

ifCondition()

```
void arcs::aubo_scope::ScriptWriter::ifCondition (
    const ExpressionPtr & expression)
```

开始 if 条件。

ifNotCondition()

```
void arcs::aubo_scope::ScriptWriter::ifNotCondition (
    const ExpressionPtr & expression)
```

开始取反的 if 条件。

increaseIndent()

```
void arcs::aubo_scope::ScriptWriter::increaseIndent ()
```

incrementVariable()

```
void arcs::aubo_scope::ScriptWriter::incrementVariable (
    const std::string & variable_name)
```

变量值加 1。

参数

<i>variable_name</i>	要递增的变量名称。

killThread()

```
void arcs::aubo_scope::ScriptWriter::killThread (  
    const std::string & thread_name)
```

终止线程。

lineFeed()

```
void arcs::aubo_scope::ScriptWriter::lineFeed ()
```

插入空行。

note()

```
void arcs::aubo_scope::ScriptWriter::note (  
    const std::string & expression)
```

添加注释。

returnMethod()

```
void arcs::aubo_scope::ScriptWriter::returnMethod ()
```

返回。

runThread()

```
void arcs::aubo_scope::ScriptWriter::runThread (  
    const std::string & thread_name)
```

启动线程。

setLabel()

```
void arcs::aubo_scope::ScriptWriter::setLabel (  
    int lineno,  
    const std::string & comment)
```

setPayload()

```
void arcs::aubo_scope::ScriptWriter::setPayload (  
    double mass,  
    double x,  
    double y,  
    double z)
```

设置负载质量和重心。

setRobotIndex()

```
void arcs::aubo_scope::ScriptWriter::setRobotIndex (  
    int robot_index)
```

setTcp()

```
void arcs::aubo_scope::ScriptWriter::setTcp (  
    const std::vector< double > & pose)
```

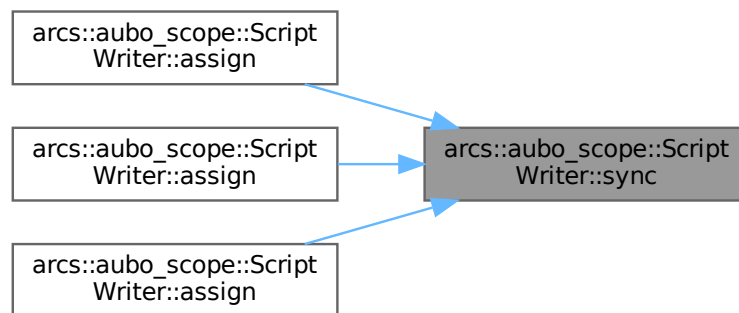
设置TCP。

sleep()

```
void arcs::aubo_scope::ScriptWriter::sleep (
    double seconds)
    休眠指定秒数。
```

sync()

```
void arcs::aubo_scope::ScriptWriter::sync ()
    用完当前帧剩余时间。
    被这些函数引用 assign\(\), assign\(\) , 以及 assign\(\).
    这是这个函数的调用关系图:
```

**whileCondition()**

```
void arcs::aubo_scope::ScriptWriter::whileCondition (
    const ExpressionPtr & expression)
    开始 while 循环。
```

whileNot()

```
void arcs::aubo_scope::ScriptWriter::whileNot (
    const ExpressionPtr & expression)
    开始取反的 while 循环。
```

whileTrue()

```
void arcs::aubo_scope::ScriptWriter::whileTrue ()
    开始 while true 循环。
```

writeChildren()

```
void arcs::aubo_scope::ScriptWriter::writeChildren ()
    当为具有子节点的节点生成脚本代码时使用此方法。让子节点生成代码。
```

11.120.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
    在文件 script\_writer.h 第 223 行定义.
    引用了 DataSwitch.
    被这些函数引用 DataSwitch.
```

11.120.5 类成员变量说明

d_

```
void* arcs::aubo_scope::ScriptWriter::d_ { nullptr } [private]
```

在文件 [script_writer.h](#) 第 225 行定义.

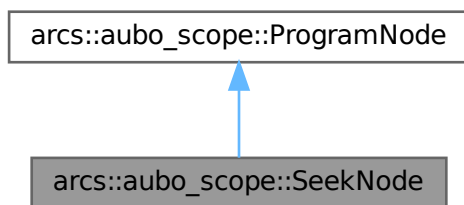
该类的文档由以下文件生成:

- [include/aubo_caps/domain/script/script_writer.h](#)

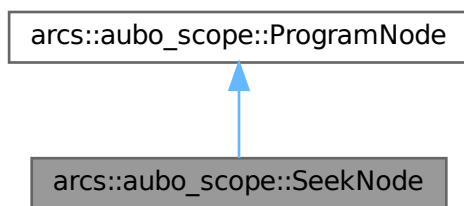
11.121 arcs::aubo_scope::SeekNode 类参考

```
#include <seek_node.h>
```

类 arcs::aubo_scope::SeekNode 继承关系图:



arcs::aubo_scope::SeekNode 的协作图:



Public 成员函数

- [SeekNode](#) (SeekNode &f)
- [SeekNode](#) (SeekNode &&f)
- virtual [~SeekNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `ScriptWriter::writeChildren`。

Private 成员函数

- [SeekNode\(\)](#)

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode\(\)](#)

11.121.1 详细描述

在文件 [seek_node.h](#) 第 10 行定义.

11.121.2 构造及析构函数说明

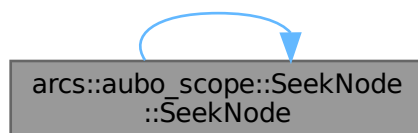
[SeekNode\(\)](#) [1/3]

```
arcs::aubo_scope::SeekNode::SeekNode (
    SeekNode & f)
```

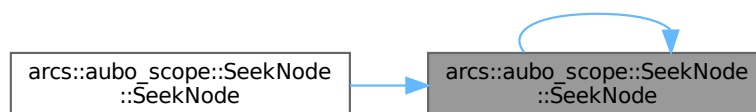
引用了 [SeekNode\(\)](#).

被这些函数引用 [SeekNode\(\)](#), 以及 [SeekNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

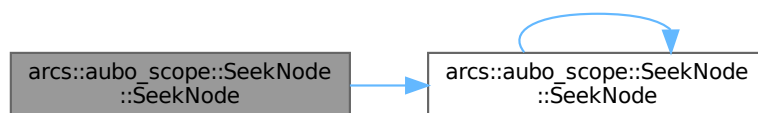


SeekNode() [2/3]

```
arcs::aubo_scope::SeekNode::SeekNode (
    SeekNode && f)
```

引用了 [SeekNode\(\)](#).

函数调用图:

**~SeekNode()**

```
virtual arcs::aubo_scope::SeekNode::~~SeekNode () [virtual]
```

SeekNode() [3/3]

```
arcs::aubo_scope::SeekNode::SeekNode () [private]
```

11.121.3 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [seek_node.h](#) 第 18 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.121.4 类成员变量说明**d_**

```
void* arcs::aubo_scope::SeekNode::d_ { nullptr } [private]
```

在文件 [seek_node.h](#) 第 20 行定义.

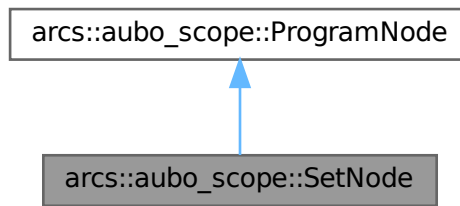
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/seek_node.h](#)

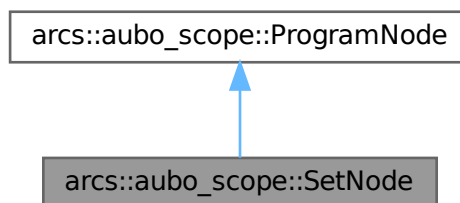
11.122 arcs::aubo_scope::SetNode 类参考

```
#include <set_node.h>
```

类 arcs::aubo_scope::SetNode 继承关系图:



arcs::aubo_scope::SetNode 的协作图:



Public 类型

- enum [ConfigType](#) {
[NO_TYPE](#) , [NO_ACTION](#) , [DIGITAL_OUTPUT](#) , [ANALOG_OUTPUT_CURRENT](#) ,
[ANALOG_OUTPUT_VOLTAGE](#) , [FLOAT_REGISTER_OUTPUT](#) , [EXPRESSION_OUTPUT](#) ,
[SINGLE_DIGITAL_PULSE](#) ,
[INCREMENT_VARIABLE](#) }

The configuration type used to determine which type of configuration this instance is.

- enum [OutputSelectionType](#) { [NO_OUTPUT](#) , [UNRESOLVED_OUTPUT](#) , [OUTPUT](#) }

Public 成员函数

- [SetNode](#) (SetNode &f)
- [SetNode](#) (SetNode &&f)
- virtual [~SetNode](#) ()
- [ConfigType](#) [getConfigType](#) ()
 返回配置类型。适当转换此实例以访问特定的 *getter*。
- void [setConfigType](#) ([ConfigType](#) type)
- double [getPulseTime](#) ()
- void [setPulseTime](#) (double time)
- float [getFloatValueToSet](#) ()
- void [setFloatValueToSet](#) (float value)
- double [getAnalogOutput](#) ()
- void [setAnalogOutput](#) (double value)

- `OutputSelectionType` `getOutputSelectionType` ()
- `ExpressionPtr` `getExpression` ()
- `void` `setExpression` (`ExpressionPtr` expression)
- `IoPtr` `getOutput` ()

The output selected can be an analog output, a digital output, a MODBUS output or a register output.

- `void` `setOutput` (`IoPtr` io)
- `bool` `getBoolValueToSet` ()

Depending on the type of output this is interpreted in different ways.

- `void` `setBoolValueToSet` (`bool` value)
- `double` `getVoltage` ()
- `void` `setVoltage` (`double` voltage)
- `double` `getCurrent` ()
- `void` `setCurrent` (`double` curr)

Public 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` (`ProgramNode` &f)
- `ProgramNode` (`ProgramNode` &&f)
- `virtual` `~ProgramNode` ()
- `ProgramNodeContributionPtr` `getProgramNodeContribution` ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `Script←Writer::writeChildren`。

Private 成员函数

- `SetNode` ()

Private 属性

- `void *` `d_` { `nullptr` }

友元

- `class` `DataSwitch`

额外继承的成员函数

Protected 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` ()

11.122.1 详细描述

在文件 `set_node.h` 第 12 行定义.

11.122.2 成员枚举类型说明

ConfigType

```
enum arcs::aubo_scope::SetNode::ConfigType
```

The configuration type used to determine which type of configuration this instance is.

枚举值

<code>NO_TYPE</code>	No selection has been made. This type has no further information.

NO_ACTION	No action has been selected. This type has no further information.
DIGITAL_OUTPUT	Digital output has been selected. The config instance can be cast to <code>DigitalOutputSetNodeConfig</code> .
ANALOG_OUTPUT_CURRENT	Analog output configured for electric current or no output has been selected. The config instance can be cast to <code>AnalogOutputCurrentSetNodeConfig</code> .
ANALOG_OUTPUT_VOLTAGE	Analog output configured for voltage has been selected. The config instance can be cast to <code>AnalogOutputVoltageSetNodeConfig</code> .
FLOAT_REGISTER_OUTPUT	Register using float type values has been selected. The config instance can be cast to <code>FloatRegisterOutputSetNodeConfig</code> .
EXPRESSION_OUTPUT	Expression output has been selected. The config instance can be cast to <code>ExpressionOutputSetNodeConfig</code> .
SINGLE_DIGITAL_PULSE	Single digital pulse output has been selected. The config instance can be cast to <code>SingleDigitalPulseSetNodeConfig</code> .
INCREMENT_VARIABLE	Increment variable has been selected. This type has no further information. Setting this type of config will be ignored and will have no effect on the node.

在文件 [set_node.h](#) 第 19 行定义.

OutputSelectionType

```
enum arcs::aubo_scope::SetNode::OutputSelectionType
```

枚举值

NO_OUTPUT	No output is selected.
UNRESOLVED_OUTPUT	The selected output is no longer available (e.g. because the end user has removed the output in the Installation or the needed Installation is not currently loaded).
OUTPUT	Output is selected. The Selection instance can be cast to <code>ExpressionOutput</code> .

在文件 [set_node.h](#) 第 104 行定义.

11.122.3 构造及析构函数说明

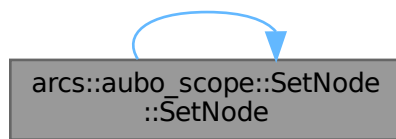
SetNode() [1/3]

```
arcs::aubo_scope::SetNode::SetNode (
    SetNode & f)
```

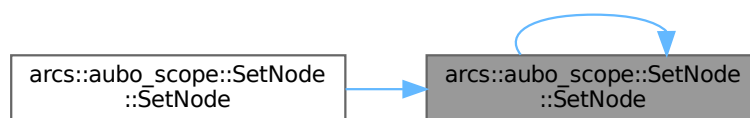
引用了 [SetNode\(\)](#).

被这些函数引用 [SetNode\(\)](#) , 以及 [SetNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

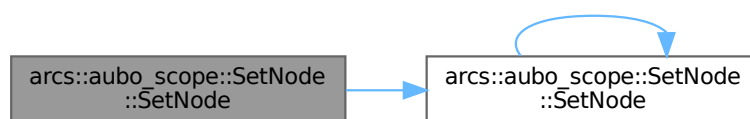


SetNode() [2/3]

```
arcs::aubo_scope::SetNode::SetNode (
    SetNode && f)
```

引用了 [SetNode\(\)](#).

函数调用图:



~SetNode()

```
virtual arcs::aubo_scope::SetNode::~~SetNode () [virtual]
```

SetNode() [3/3]

```
arcs::aubo_scope::SetNode::SetNode () [private]
```

11.122.4 成员函数说明

getAnalogOutput()

```
double arcs::aubo_scope::SetNode::getAnalogOutput ()
```


返回

the analog output value.

getBoolValueToSet()

```
bool arcs::aubo_scope::SetNode::getBoolValueToSet ()
```

Depending on the type of output this is interpreted in different ways.

For digital output and MODBUS output, **true** means the output will be set to HIGH, false means the output will be set to LOW. For boolean register the value is interpreted literally.

返回

the output value.

getConfigType()

```
ConfigType arcs::aubo_scope::SetNode::getConfigType ()
```

返回配置类型。适当转换此实例以访问特定的 getter。

返回

此配置的类型

getCurrent()

```
double arcs::aubo_scope::SetNode::getCurrent ()
```

返回

the electric current value/level to be applied to the output.

getExpression()

```
ExpressionPtr arcs::aubo_scope::SetNode::getExpression ()
```

返回

the expression whose evaluation will be applied to the output.

getFloatValueToSet()

```
float arcs::aubo_scope::SetNode::getFloatValueToSet ()
```

返回

the float value to be applied to the float register output.

getOutput()

```
IoPtr arcs::aubo_scope::SetNode::getOutput ()
```

The output selected can be an analog output, a digital output, a MODBUS output or a register output.

返回

the output to which the result of the evaluation of the expression will be applied.

getOutputSelectionType()

```
OutputSelectionType arcs::aubo_scope::SetNode::getOutputSelectionType ()
```

getPulseTime()

```
double arcs::aubo_scope::SetNode::getPulseTime ()
```

返回

the time for the length of the pulse.

getVoltage()

```
double arcs::aubo_scope::SetNode::getVoltage ()
```

返回

the voltage value/level to be applied to the output

setAnalogOutput()

```
void arcs::aubo_scope::SetNode::setAnalogOutput (  
    double value)
```

setBoolValueToSet()

```
void arcs::aubo_scope::SetNode::setBoolValueToSet (  
    bool value)
```

setConfigType()

```
void arcs::aubo_scope::SetNode::setConfigType (  
    ConfigType type)
```

setCurrent()

```
void arcs::aubo_scope::SetNode::setCurrent (  
    double curr)
```

setExpression()

```
void arcs::aubo_scope::SetNode::setExpression (  
    ExpressionPtr expression)
```

setFloatValueToSet()

```
void arcs::aubo_scope::SetNode::setFloatValueToSet (  
    float value)
```

setOutput()

```
void arcs::aubo_scope::SetNode::setOutput (  
    IoPtr io)
```

setPulseTime()

```
void arcs::aubo_scope::SetNode::setPulseTime (  
    double time)
```

setVoltage()

```
void arcs::aubo_scope::SetNode::setVoltage (  
    double voltage)
```

11.122.5 友元及相关符号说明

DataSwitch

friend class DataSwitch [friend]
 在文件 [set_node.h](#) 第 211 行定义.
 引用了 [DataSwitch](#).
 被这些函数引用 [DataSwitch](#).

11.122.6 类成员变量说明

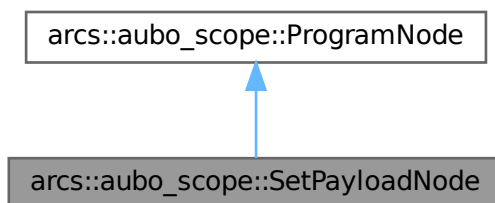
d_

void* arcs::aubo_scope::SetNode::d_ { nullptr } [private]
 在文件 [set_node.h](#) 第 213 行定义.
 该类的文档由以下文件生成:

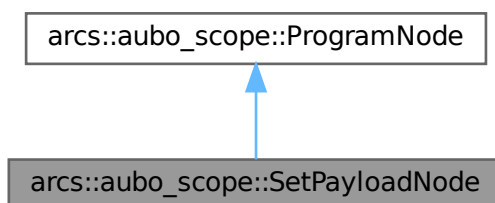
- [include/aubo_caps/domain/program/nodes/builtin/set_node.h](#)

11.123 arcs::aubo_scope::SetPayloadNode 类参考

#include <set_payload_node.h>
 类 arcs::aubo_scope::SetPayloadNode 继承关系图:



arcs::aubo_scope::SetPayloadNode 的协作图:



Public 类型

- enum [ConfigType](#) { [PAYLOAD_SELECTION](#) , [CUSTOM_PARAMETERS](#) }
- This base interface represents a configuration of the Set [Payload](#) program node (i.e.*

Public 成员函数

- `SetPayloadNode` (`SetPayloadNode &f`)
- `SetPayloadNode` (`SetPayloadNode &&f`)
- `virtual ~SetPayloadNode` ()
- `ConfigType getConfigType` ()
返回配置类型。适当转换此实例以访问特定的 *getter*。
- `void setConfigType` (`ConfigType type`)
- `double getMass` ()
- `void setMass` (`double mass`)
- `std::vector< double > getCenterOfGravity` ()
获取负载的重心 (*CoG*)。它定义为工具输出法兰中心与附着负载重心之间的偏移量。
- `void setCenterOfGravity` (`const std::vector< double > &cog`)

Public 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` (`ProgramNode &f`)
- `ProgramNode` (`ProgramNode &&f`)
- `virtual ~ProgramNode` ()
- `ProgramNodeContributionPtr getProgramNodeContribution` ()
获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `Script←Writer::writeChildren`。

Private 成员函数

- `SetPayloadNode` ()

Private 属性

- `void * d_` { `nullptr` }

友元

- `class DataSwitch`

额外继承的成员函数**Protected 成员函数继承自 `arcs::aubo_scope::ProgramNode`**

- `ProgramNode` ()

11.123.1 详细描述

在文件 `set_payload_node.h` 第 11 行定义。

11.123.2 成员枚举类型说明**ConfigType**

```
enum arcs::aubo_scope::SetPayloadNode::ConfigType
```

This base interface represents a configuration of the Set `Payload` program node (i.e. a `SetPayloadNode` instance).

枚举值

PAYLOAD_SELECTION	A payload (from the installation) is selected. The config instance can be cast to <code>SelectionPayloadNodeConfig</code> .

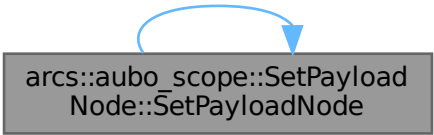
CUSTOM_PARAMETERS	Custom payload parameters are specified. The config instance can be cast to CustomParametersPayloadNodeConfig.

在文件 [set_payload_node.h](#) 第 19 行定义.

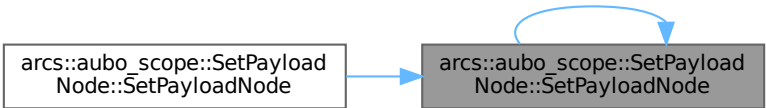
11.123.3 构造及析构造函数说明

SetPayloadNode() [1/3]

```
arcs::aubo_scope::SetPayloadNode::SetPayloadNode (  
    SetPayloadNode & f)  
引用了 SetPayloadNode\(\).  
被这些函数引用 SetPayloadNode\(\) , 以及 SetPayloadNode\(\).  
函数调用图:
```

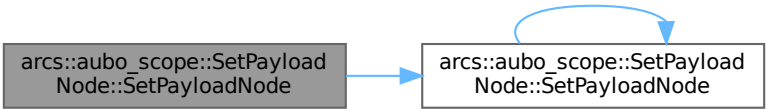


这是这个函数的调用关系图:



SetPayloadNode() [2/3]

```
arcs::aubo_scope::SetPayloadNode::SetPayloadNode (  
    SetPayloadNode && f)  
引用了 SetPayloadNode\(\).  
函数调用图:
```



~SetPayloadNode()

```
virtual arcs::aubo_scope::SetPayloadNode::~~SetPayloadNode () [virtual]
```

SetPayloadNode() [3/3]

```
arcs::aubo_scope::SetPayloadNode::SetPayloadNode () [private]
```

11.123.4 成员函数说明**getCenterOfGravity()**

```
std::vector< double > arcs::aubo_scope::SetPayloadNode::getCenterOfGravity ()
```

获取负载的重心 (CoG)。它定义为工具输出法兰中心与附着负载重心之间的偏移量。

返回

相对于工具输出法兰中心的负载重心 (CoG)

getConfigType()

```
ConfigType arcs::aubo_scope::SetPayloadNode::getConfigType ()
```

返回配置类型。适当转换此实例以访问特定的 getter。

返回

此配置的类型

getMass()

```
double arcs::aubo_scope::SetPayloadNode::getMass ()
```

返回

负载的总质量

setCenterOfGravity()

```
void arcs::aubo_scope::SetPayloadNode::setCenterOfGravity (
    const std::vector< double > & cog)
```

setConfigType()

```
void arcs::aubo_scope::SetPayloadNode::setConfigType (
    ConfigType type)
```

setMass()

```
void arcs::aubo_scope::SetPayloadNode::setMass (
    double mass)
```

11.123.5 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [set_payload_node.h](#) 第 88 行定义。
引用了 [DataSwitch](#)。
被这些函数引用 [DataSwitch](#)。

11.123.6 类成员变量说明

d_

```
void* arcs::aubo_scope::SetPayloadNode::d_ { nullptr } [private]
```

在文件 [set_payload_node.h](#) 第 90 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/set_payload_node.h](#)

11.124 arcs::aubo_scope::SpeedCapability 类参考

速度能力此接口表示夹爪已注册的速度能力，支持使用用户可配置的速度进行操作。

```
#include <speed_capability.h>
```

Public 成员函数

- virtual [~SpeedCapability](#) ()=default
- virtual void [updateCapability](#) (double minSpeed, double maxSpeed, double defaultGripSpeed, double defaultReleaseSpeed)=0

11.124.1 详细描述

速度能力此接口表示夹爪已注册的速度能力，支持使用用户可配置的速度进行操作。

通过此接口，可以在能力注册后调整速度的范围和默认值。这通常不适用于大多数夹爪，但当范围取决于另一个自定义夹爪设置（可由用户配置）时，这可能是相关的。

在文件 [speed_capability.h](#) 第 29 行定义.

11.124.2 构造及析构函数说明

[~SpeedCapability\(\)](#)

```
virtual arcs::aubo_scope::SpeedCapability::~~SpeedCapability () [virtual], [default]
```

11.124.3 成员函数说明

[updateCapability\(\)](#)

```
virtual void arcs::aubo_scope::SpeedCapability::updateCapability (
    double minSpeed,
    double maxSpeed,
    double defaultGripSpeed,
    double defaultReleaseSpeed) [pure virtual]
```

Update the range and default values for the speed supported by the gripper.

Updating the range could result in an already entered value in the configuration of a gripper program node being outside the range. This will make the given program node undefined, and force the end user to adjust the value to be valid again.

参数

<i>minSpeed</i>	minimum operating speed supported by the gripper
<i>maxSpeed</i>	maximum operating speed supported by the gripper
<i>defaultGripSpeed</i>	default value for the speed for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
<i>defaultReleaseSpeed</i>	default value for the speed for a release action. This value is used for, e.g. performing a "default" release action using the toolbar
<i>unit</i>	the unit for all specified values, not null

异常

<i>InvalidCapabilityRange</i>	if minSpeed > maxSpeed
<i>InvalidCapabilityRange</i>	if defaultGripSpeed or defaultReleaseSpeed is outside the range defined by minSpeed and maxSpeed.

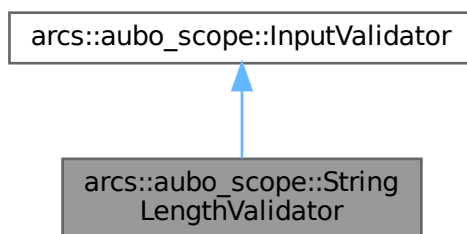
该类的文档由以下文件生成:

- include/aubo_caps/contribution/driver/gripper/capability/speed_capability.h

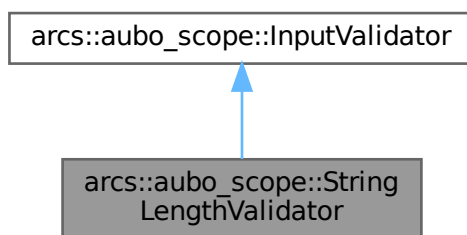
11.125 arcs::aubo_scope::StringLengthValidator 结构体参考

```
#include <input_validator.h>
```

类 arcs::aubo_scope::StringLengthValidator 继承关系图:



arcs::aubo_scope::StringLengthValidator 的协作图:



Public 成员函数

- [StringLengthValidator](#) (size_t min_length, size_t max_length)
- [StringLengthValidator](#) (std::function< std::pair< size_t, size_t >()> range_func)
- bool [isValid](#) (std::string value) override
验证输入值是否有效。
- std::string [getMessage](#) (std::string value) override
当值无效时返回有意义的提示信息。

Private 成员函数

- void [updateValue](#) ()

Private 属性

- size_t [min_length](#) { 0 }
- size_t [max_length](#) { ULONG_MAX }
- std::function< std::pair< size_t, size_t >()> [range_func](#) { nullptr }

11.125.1 详细描述

在文件 [input_validator.h](#) 第 146 行定义.

11.125.2 构造及析构造函数说明**StringLengthValidator()** [1/2]

```
arcs::aubo_scope::StringLengthValidator::StringLengthValidator (
    size_t min_length,
    size_t max_length)
```

StringLengthValidator() [2/2]

```
arcs::aubo_scope::StringLengthValidator::StringLengthValidator (
    std::function< std::pair< size_t, size_t >()> range_func)
```

11.125.3 成员函数说明**getMessage()**

```
std::string arcs::aubo_scope::StringLengthValidator::getMessage (
    std::string value) [override], [virtual]
```

当值无效时返回有意义的提示信息。

参数

<i>value</i>	无效的值，如果合理可包含在消息中

返回

提示信息

实现了 [arcs::aubo_scope::InputValidator](#).

isValid()

```
bool arcs::aubo_scope::StringLengthValidator::isValid (
    std::string value) [override], [virtual]
```

验证输入值是否有效。

参数

<i>value</i>	需要验证的值

返回

如果值有效则返回 `true`

实现了 [arcs::aubo_scope::InputValidator](#).

updateValue()

```
void arcs::aubo_scope::StringLengthValidator::updateValue () [private]
```

11.125.4 类成员变量说明**max_length__**

```
size_t arcs::aubo_scope::StringLengthValidator::max_length_ { ULONG_MAX } [private]
```

在文件 [input_validator.h](#) 第 162 行定义.

min_length__

```
size_t arcs::aubo_scope::StringLengthValidator::min_length_ { 0 } [private]
```

在文件 [input_validator.h](#) 第 161 行定义.

range_func__

```
std::function<std::pair<size_t, size_t>()> arcs::aubo_scope::StringLengthValidator::range_func_ { nullptr } [private]
```

在文件 [input_validator.h](#) 第 163 行定义.

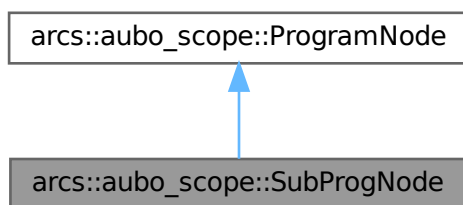
该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/user_interface/input_validator.h](#)

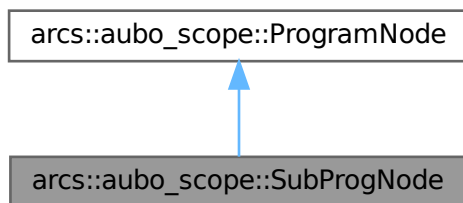
11.126 arcs::aubo_scope::SubProgNode 类参考

```
#include <sub_prog_node.h>
```

类 arcs::aubo_scope::SubProgNode 继承关系图:



arcs::aubo_scope::SubProgNode 的协作图:



Public 成员函数

- [SubProgNode](#) (SubProgNode &f)
- [SubProgNode](#) (SubProgNode &&f)
- virtual [~SubProgNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
 获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *Script←Writer::writeChildren*。

Private 成员函数

- [SubProgNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数**Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)**

- [ProgramNode](#) ()

11.126.1 详细描述

在文件 [sub_prog_node.h](#) 第 10 行定义.

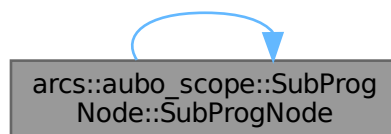
11.126.2 构造及析构函数说明**SubProgNode() [1/3]**

```
arcs::aubo_scope::SubProgNode::SubProgNode (
    SubProgNode & f)
```

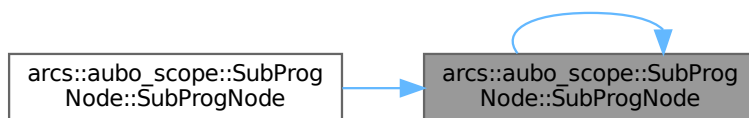
引用了 [SubProgNode\(\)](#).

被这些函数引用 [SubProgNode\(\)](#) , 以及 [SubProgNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

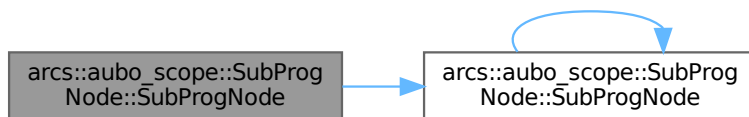


SubProgNode() [2/3]

```
arcs::aubo_scope::SubProgNode::SubProgNode (
    SubProgNode && f)
```

引用了 [SubProgNode\(\)](#).

函数调用图:



~SubProgNode()

```
virtual arcs::aubo_scope::SubProgNode::~~SubProgNode () [virtual]
```

SubProgNode() [3/3]

```
arcs::aubo_scope::SubProgNode::SubProgNode () [private]
```

11.126.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [sub_prog_node.h](#) 第 18 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.126.4 类成员变量说明

d_

```
void* arcs::aubo_scope::SubProgNode::d_ { nullptr } [private]
```

在文件 [sub_prog_node.h](#) 第 20 行定义.

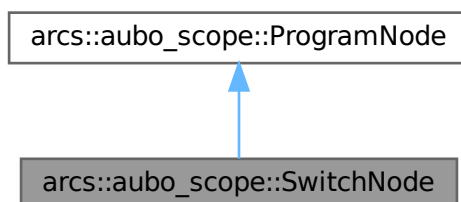
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/sub_prog_node.h](#)

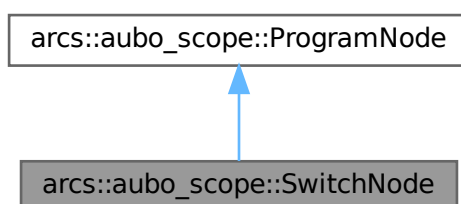
11.127 arcs::aubo_scope::SwitchNode 类参考

```
#include <switch_node.h>
```

类 arcs::aubo_scope::SwitchNode 继承关系图:



arcs::aubo_scope::SwitchNode 的协作图:



Public 成员函数

- [SwitchNode](#) (SwitchNode &f)
- [SwitchNode](#) (SwitchNode &&f)
- virtual [~SwitchNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [SwitchNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode\(\)](#)

11.127.1 详细描述

在文件 [switch_node.h](#) 第 10 行定义.

11.127.2 构造及析构函数说明

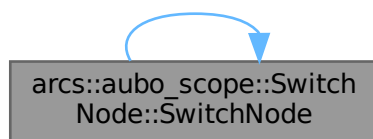
SwitchNode() [1/3]

```
arcs::aubo_scope::SwitchNode::SwitchNode (  
    SwitchNode & f)
```

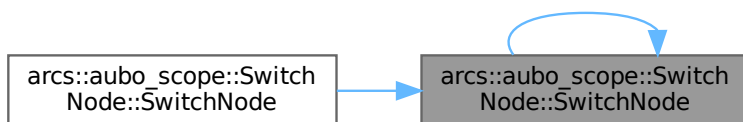
引用了 [SwitchNode\(\)](#).

被这些函数引用 [SwitchNode\(\)](#) , 以及 [SwitchNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

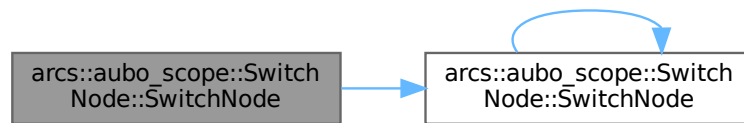


SwitchNode() [2/3]

```
arcs::aubo_scope::SwitchNode::SwitchNode (  
    SwitchNode && f)
```

引用了 [SwitchNode\(\)](#).

函数调用图:



~SwitchNode()

```
virtual arcs::aubo_scope::SwitchNode::~SwitchNode () [virtual]
```

SwitchNode() [3/3]

```
arcs::aubo_scope::SwitchNode::SwitchNode () [private]
```

11.127.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [switch_node.h](#) 第 18 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.127.4 类成员变量说明

d_

```
void* arcs::aubo_scope::SwitchNode::d_ { nullptr } [private]
```

在文件 [switch_node.h](#) 第 20 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/switch_node.h](#)

11.128 arcs::aubo_scope::SystemApi 类参考

```
#include <system_api.h>
```

Public 成员函数

- [SystemApi](#) (SystemApi &f)
- [SystemApi](#) (SystemApi &&f)
- virtual [~SystemApi](#) ()
- int [getMajorVersion](#) ()
获取软件版本号
- int [getMinorVersion](#) ()
获取软件次版本号
- int [getBugfixVersion](#) ()
获取软件 *Bug* 修复版本号
- int [getBuildNumber](#) ()
获取软件构建号

- `std::string getLocale ()`
返回系统的`Locale`语言区域设置。
- `std::string getLocaleForProgrammingLanguage ()`
返回系统中编程语言使用的`Locale`语言区域设置。
- `UnitType getUnitType ()`
以枚举类型返回系统的单位制。
- `std::string getSerialNumber ()`
获取机器人的序列号
- `bool isRealRobot ()`
检查是否控制真实机器人
- `std::vector< double > getJointMaxPositions ()`
获取机器人各关节的最大位置
- `std::vector< double > getJointMinPositions ()`
获取机器人各关节的最小位置
- `double getJointDefaultVelocity ()`
获取关节默认速度
- `double getJointDefaultAcceleration ()`
获取关节默认加速度
- `std::vector< double > getJointMaxVelocity ()`
获取各关节最大速度
- `std::vector< double > getJointMaxAcceleration ()`
获取各关节最大加速度
- `double getTcpDefaultVelocity ()`
获取`TCP` 默认速度
- `double getTcpDefaultAcceleration ()`
获取`TCP` 默认加速度
- `double getTcpMaxVelocity ()`
获取`TCP` 最大速度
- `double getTcpMaxAcceleration ()`
获取`TCP` 最大加速度
- `double getTcpMaxDistance ()`
获取`TCP` 最大移动距离
- `double getMaxPayloadMass ()`
获取最大负载质量
- `double getMaxPayloadDistance ()`
获取最大负载距离
- `std::vector< double > getHomeDefaultPosition ()`
获取默认`Home` 位置
- `std::vector< double > getHomePackagePosition ()`
获取包装`Home` 位置
- `std::unordered_map< std::string, std::vector< double > > getRobotDhParams (bool is_real)`
获取机器人`DH` 参数
- `IoConfigure getToolDigitalIOConfigure ()`
获取工具端数字`IO` 配置
- `IoConfigure getStandardDigitalInputConfigure ()`
获取标准数字输入配置
- `IoConfigure getConfigurableDigitalInputConfigure ()`
获取可配置数字输入配置
- `IoConfigure getStandardDigitalOutputConfigure ()`
获取标准数字输出配置
- `IoConfigure getConfigurableDigitalOutputConfigure ()`

获取可配置数字输出配置

- [IoConfigure getStandardAnalogInputConfigure \(\)](#)
获取标准模拟输入配置
- [IoConfigure getToolAnalogInputConfigure \(\)](#)
获取工具端模拟输入配置
- [IoConfigure getStandardAnalogOutputConfigure \(\)](#)
获取标准模拟输出配置
- [IoConfigure getToolAnalogOutputConfigure \(\)](#)
获取工具端模拟输出配置
- [IoConfigure getStaticLinkInputConfigure \(\)](#)
获取静态链路输入配置
- [IoConfigure getStaticLinkOutputConfigure \(\)](#)
获取静态链路输出配置
- [IoConfigure getENCConfigure \(\)](#)
获取编码器配置

Private 成员函数

- [SystemApi \(\)](#)

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.128.1 详细描述

在文件 [system_api.h](#) 第 54 行定义.

11.128.2 构造及析构函数说明

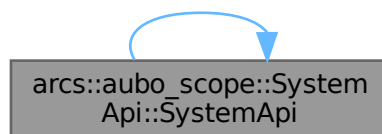
SystemApi() [1/3]

```
arcs::aubo_scope::SystemApi::SystemApi (
    SystemApi & f)
```

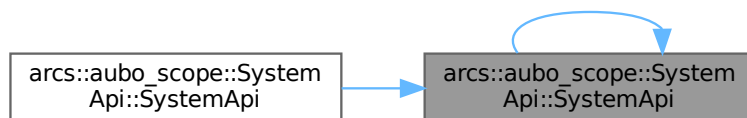
引用了 [SystemApi\(\)](#).

被这些函数引用 [SystemApi\(\)](#) , 以及 [SystemApi\(\)](#).

函数调用图:



这是这个函数的调用关系图:

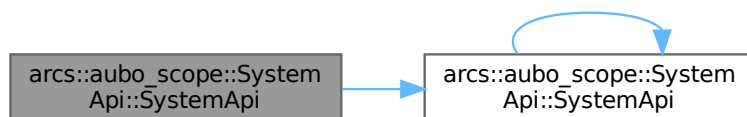


SystemApi() [2/3]

```
arcs::aubo_scope::SystemApi::SystemApi (
    SystemApi && f)
```

引用了 [SystemApi\(\)](#).

函数调用图:



~SystemApi()

```
virtual arcs::aubo_scope::SystemApi::~~SystemApi () [virtual]
```

SystemApi() [3/3]

```
arcs::aubo_scope::SystemApi::SystemApi () [private]
```

11.128.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [system_api.h](#) 第 556 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.128.4 类成员变量说明

d_

```
void* arcs::aubo_scope::SystemApi::d_ { nullptr } [private]
```

在文件 [system_api.h](#) 第 558 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/system_api.h](#)

11.129 arcs::aubo_scope::TCP 类参考

TCP 接口此接口表示 AuboScope 中的一个 TCP（工具中心点）。可以是用户定义的 TCP，也可以是由 AuboCap 添加的 TCP。

```
#include <tcp.h>
```

Public 成员函数

- **TCP** (TCP &f)
- **TCP** (TCP &&f)
- **~TCP** ()
- std::vector< double > **getOffset** ()
获取 TCP 相对于工具输出法兰的偏移量
- std::string **getDisplayname** ()
获取 TCP 显示名称。注意：TCP 可随时被重命名。
- bool **isResolvable** ()
TCP 无法保证始终存在。终端用户可加载不包含该 TCP 的其他安装，或该 TCP 可被终端用户或添加该 TCP 的 AuboCap 删除。此方法用于判断 TCP 是否存在。
- size_t **hashCode** ()
获取哈希码

Private 成员函数

- **TCP** ()

Private 属性

- void * **d_** { nullptr }

友元

- class **DataSwitch**

11.129.1 详细描述

TCP 接口此接口表示 AuboScope 中的一个 TCP（工具中心点）。可以是用户定义的 TCP，也可以是由 AuboCap 添加的 TCP。

在文件 [tcp.h](#) 第 25 行定义。

11.129.2 构造及析构函数说明

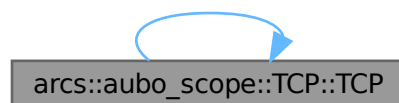
TCP() [1/3]

```
arcs::aubo_scope::TCP::TCP (  
    TCP & f)
```

引用了 [TCP\(\)](#)。

被这些函数引用 [TCP\(\)](#)，以及 [TCP\(\)](#)。

函数调用图:



这是这个函数的调用关系图:



TCP() [2/3]

```
arcs::aubo_scope::TCP::TCP (
    TCP && f)
```

引用了 [TCP\(\)](#).

函数调用图:



~TCP()

```
arcs::aubo_scope::TCP::~~TCP ()
```

TCP() [3/3]

```
arcs::aubo_scope::TCP::TCP () [private]
```

11.129.3 成员函数说明

getDisplayNames()

```
std::string arcs::aubo_scope::TCP::getDisplayNames ()
```

获取TCP 显示名称。注意：TCP 可随时被重命名。

返回

TCP 名称。

getOffset()

```
std::vector< double > arcs::aubo_scope::TCP::getOffset ()
```

获取TCP 相对于工具输出法兰的偏移量

返回

TCP 偏移量。

hashCode()

```
size_t arcs::aubo_scope::TCP::hashCode ()
```

获取哈希码

返回

哈希码。

isResolvable()

```
bool arcs::aubo_scope::TCP::isResolvable ()
```

TCP 无法保证始终存在。终端用户可加载不包含该TCP 的其他安装，或该TCP 可被终端用户或添加该TCP 的AuboCap 删除。此方法用于判断TCP 是否存在。

返回

如果TCP 存在返回 true，否则返回 false。

11.129.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [tcp.h](#) 第 91 行定义。

引用了 [DataSwitch](#)。

被这些函数引用 [DataSwitch](#)。

11.129.5 类成员变量说明**d_**

```
void* arcs::aubo_scope::TCP::d_ { nullptr } [private]
```

在文件 [tcp.h](#) 第 93 行定义。

该类的文档由以下文件生成:

- [include/aubo_caps/domain/tcp/tcp.h](#)

11.130 arcs::aubo_scope::TcpContributionModel 类参考

[TCP](#) 贡献模型此接口提供在 AuboScope 中添加、更新和删除 [TCP](#) 的功能。

```
#include <tcp_contribution_model.h>
```

Public 成员函数

- [TcpContributionModel](#) (TcpContributionModel &f)
- [TcpContributionModel](#) (TcpContributionModel &&f)
- virtual [~TcpContributionModel](#) ()
- TCPPtr [addTcp](#) (const std::string &idKey, const std::string &suggested_name, const std::vector< double > &offset)

[TCP](#) 向当前AuboScope 安装中添加一个TCP。添加后终端用户可选择该TCP，但不可修改。

- TCPPtr [getTcp](#) (const std::string &idKey)

[TCP](#) 返回此前由此AuboCap 使用相同 idKey 添加的TCP。

- void [updateTcp](#) (const std::string &idKey, const std::vector< double > &newOffset)

更新此AuboCap 已添加TCP 的偏移量。

- void [removeTcp](#) (const std::string &idKey)

从此AuboCap 中移除已添加的TCP。使用该TCP 的程序节点将变为未定义。

Private 成员函数

- [TcpContributionModel\(\)](#)

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

11.130.1 详细描述

[TCP](#) 贡献模型此接口提供在 AuboScope 中添加、更新和删除 [TCP](#) 的功能。

注意：此接口只能在安装贡献中使用。

在文件 [tcp_contribution_model.h](#) 第 27 行定义。

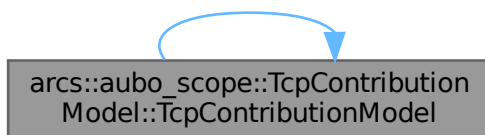
11.130.2 构造及析构函数说明**TcpContributionModel() [1/3]**

```
arcs::aubo_scope::TcpContributionModel::TcpContributionModel (
    TcpContributionModel & f)
```

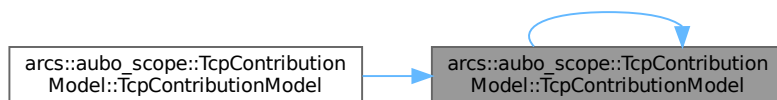
引用了 [TcpContributionModel\(\)](#)。

被这些函数引用 [TcpContributionModel\(\)](#) , 以及 [TcpContributionModel\(\)](#)。

函数调用图:



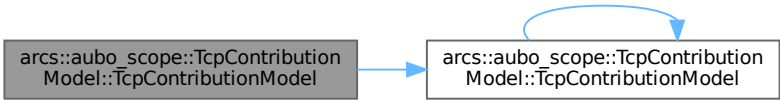
这是这个函数的调用关系图:

**TcpContributionModel() [2/3]**

```
arcs::aubo_scope::TcpContributionModel::TcpContributionModel (
    TcpContributionModel && f)
```

引用了 [TcpContributionModel\(\)](#)。

函数调用图:



```
~TcpContributionModel()

virtual arcs::aubo_scope::TcpContributionModel::~~TcpContributionModel () [virtual]

TcpContributionModel() [3/3]

arcs::aubo_scope::TcpContributionModel::TcpContributionModel () [private]
```

11.130.3 成员函数说明

```
addTcp()

TCPPtr arcs::aubo_scope::TcpContributionModel::addTcp (
    const std::string & idKey,
    const std::string & suggested_name,
    const std::vector< double > & offset)
TCP 向当前AuboScope 安装中添加一个TCP。添加后终端用户可选择该TCP，但不可修改。
```

参数

<i>idKey</i>	用于标识此TCP 的键。该键仅对此AuboCap 有效。
<i>suggested_name</i>	建议的TCP 名称。
<i>offset</i>	TCP 相对于工具输出法兰的偏移量。

返回

在AuboScope 中创建并注册的TCP。

异常

<i>TCPAlreadyAddedException</i>	如果已使用相同 idKey 添加过TCP。
<i>IllegalTCPNameException</i>	如果建议的名称不匹配所需正则表达式。
<i>IllegalArgumentException</i>	如果偏移量超出有效范围。

getTcp()

```
TCPPtr arcs::aubo_scope::TcpContributionModel::getTcp (  
    const std::string & idKey)
```

TCP 返回此前由此AuboCap 使用相同 idKey 添加的TCP。

参数

<i>idKey</i>	TCP 标识符键。

返回

此前添加的TCP，如果不存在则返回 `null`。

removeTcp()

```
void arcs::aubo_scope::TcpContributionModel::removeTcp (  
    const std::string & idKey)
```

从此AuboCap 中移除已添加的TCP。使用该TCP 的程序节点将变为未定义。

参数

<i>idKey</i>	添加TCP 时使用的键。

异常

<i>TCPNotFoundException</i>	如果不存在对应 idKey 的TCP。

updateTcp()

```
void arcs::aubo_scope::TcpContributionModel::updateTcp (  
    const std::string & idKey,  
    const std::vector< double > & newOffset)
```

更新此AuboCap 已添加TCP 的偏移量。

参数

<i>idKey</i>	TCP 标识符键。
<i>newOffset</i>	TCP 的新偏移量。

异常

<i>TCPNotFoundException</i>	如果不存在对应 idKey 的TCP。
<i>IllegalArgumentException</i>	如果偏移量超出有效范围。

11.130.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [tcp_contribution_model.h](#) 第 116 行定义。

引用了 [DataSwitch](#)。

被这些函数引用 [DataSwitch](#)。

11.130.5 类成员变量说明

d_

```
void* arcs::aubo_scope::TcpContributionModel::d_ { nullptr } [private]
```

在文件 [tcp_contribution_model.h](#) 第 118 行定义。

该类的文档由以下文件生成:

- [include/aubo_caps/domain/tcp/tcp_contribution_model.h](#)

11.131 arcs::aubo_scope::TcpModel 类参考

[TCP](#) 模型此接口提供对 [AuboScope](#) 中当前可用的 [TCP](#) 的访问。

```
#include <tcp_model.h>
```

Public 成员函数

- [TcpModel](#) (TcpModel &f)
- [TcpModel](#) (TcpModel &&f)
- virtual [~TcpModel](#) ()
- std::vector< [TCPPtr](#) > [getAll](#) ()
获取 *AuboScope* 中所有 *TCP* 的集合
- [TCPPtr](#) [getTcp](#) (const std::string &name)
TCP 根据名称获取 *TCP*
- [TCPPtr](#) [getDefaultTcp](#) ()
TCP 获取默认 *TCP*
- std::string [connectedTo](#) (const std::string &receiver, const std::function< void([TcpModel](#) *)> &slot)
注册变更回调
- void [disconnectFrom](#) (const std::string &receiver)
断开回调连接

Private 成员函数

- [TcpModel](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.131.1 详细描述

[TCP](#) 模型此接口提供对 AuboScope 中当前可用的 [TCP](#) 的访问。
在文件 [tcp_model.h](#) 第 24 行定义.

11.131.2 构造及析构造函数说明

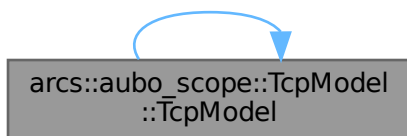
TcpModel() [1/3]

```
arcs::aubo_scope::TcpModel::TcpModel (  
    TcpModel & f)
```

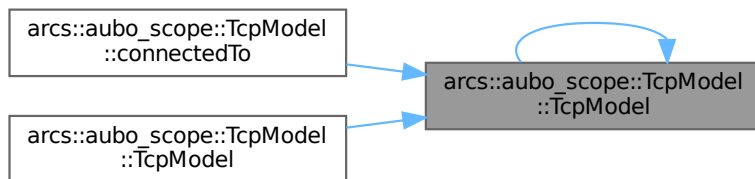
引用了 [TcpModel\(\)](#).

被这些函数引用 [connectedTo\(\)](#), [TcpModel\(\)](#) , 以及 [TcpModel\(\)](#).

函数调用图:



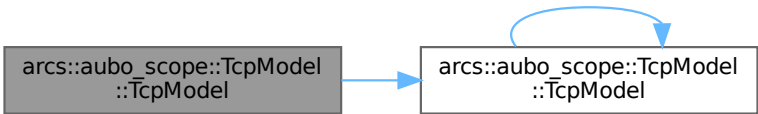
这是这个函数的调用关系图:

**TcpModel()** [2/3]

```
arcs::aubo_scope::TcpModel::TcpModel (  
    TcpModel && f)
```

引用了 [TcpModel\(\)](#).

函数调用图:



~TcpModel()

```
virtual arcs::aubo_scope::TcpModel::~TcpModel () [virtual]
```

TcpModel() [3/3]

```
arcs::aubo_scope::TcpModel::TcpModel () [private]
```

11.131.3 成员函数说明

connectedTo()

```
std::string arcs::aubo_scope::TcpModel::connectedTo (
    const std::string & receiver,
    const std::function< void(TcpModel *)> & slot)
注册变更回调
```

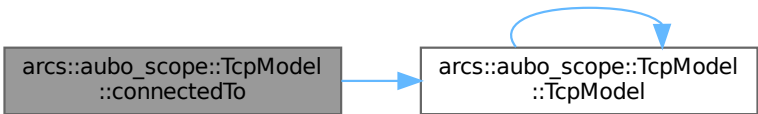
参数

<i>receiver</i>	接收者标识。
<i>slot</i>	回调函数。

返回

连接标识符。

引用了 [TcpModel\(\)](#).
函数调用图:



disconnectFrom()

```
void arcs::aubo_scope::TcpModel::disconnectFrom (  
    const std::string & receiver)
```

断开回调连接

参数

<i>receiver</i>	接收者标识。

getAll()

```
std::vector< TCPPtr > arcs::aubo_scope::TcpModel::getAll ()
```

获取AuboScope 中所有TCP 的集合

返回

所有TCP 的集合。

getDefaultTcp()

```
TCPPtr arcs::aubo_scope::TcpModel::getDefaultTcp ()
```

[TCP](#) 获取默认TCP

返回

默认TCP。

getTcp()

```
TCPPtr arcs::aubo_scope::TcpModel::getTcp (  
    const std::string & name)
```

[TCP](#) 根据名称获取TCP

参数

<i>name</i>	TCP 名称。

返回

对应名称的TCP。

11.131.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [tcp_model.h](#) 第 105 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.131.5 类成员变量说明

d_

```
void* arcs::aubo_scope::TcpModel::d_ { nullptr } [private]
```

在文件 [tcp_model.h](#) 第 107 行定义.

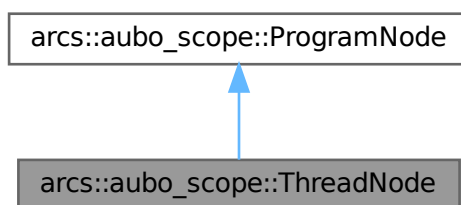
该类的文档由以下文件生成:

- [include/aubo_caps/domain/tcp/tcp_model.h](#)

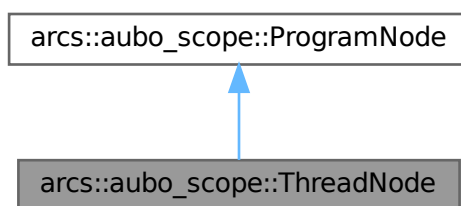
11.132 arcs::aubo_scope::ThreadNode 类参考

```
#include <thread_node.h>
```

类 arcs::aubo_scope::ThreadNode 继承关系图:



arcs::aubo_scope::ThreadNode 的协作图:



Public 成员函数

- [ThreadNode](#) (ThreadNode &f)
- [ThreadNode](#) (ThreadNode &&f)
- virtual [~ThreadNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`，应使用 `ScriptWriter::writeChildren`。

Private 成员函数

- `ThreadNode()`

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数

Protected 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode()`

11.132.1 详细描述

在文件 `thread_node.h` 第 10 行定义。

11.132.2 构造及析构函数说明

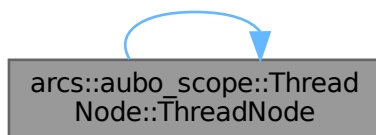
`ThreadNode()` [1/3]

```
arcs::aubo_scope::ThreadNode::ThreadNode (
    ThreadNode & f)
```

引用了 `ThreadNode()`。

被这些函数引用 `ThreadNode()`，以及 `ThreadNode()`。

函数调用图：



这是这个函数的调用关系图：

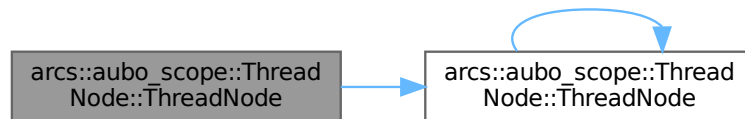


ThreadNode() [2/3]

```
arcs::aubo_scope::ThreadNode::ThreadNode (
    ThreadNode && f)
```

引用了 [ThreadNode\(\)](#).

函数调用图:

**~ThreadNode()**

```
virtual arcs::aubo_scope::ThreadNode::~~ThreadNode () [virtual]
```

ThreadNode() [3/3]

```
arcs::aubo_scope::ThreadNode::ThreadNode () [private]
```

11.132.3 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [thread_node.h](#) 第 18 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.132.4 类成员变量说明**d_**

```
void* arcs::aubo_scope::ThreadNode::d_ { nullptr } [private]
```

在文件 [thread_node.h](#) 第 20 行定义.

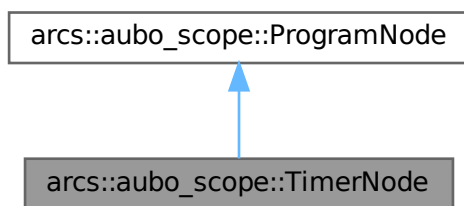
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/thread_node.h](#)

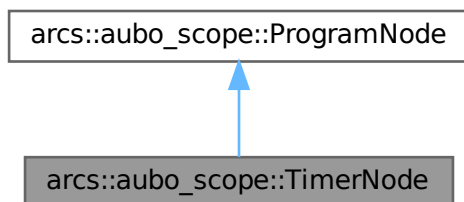
11.133 arcs::aubo_scope::TimerNode 类参考

```
#include <timer_node.h>
```

类 `arcs::aubo_scope::TimerNode` 继承关系图:



`arcs::aubo_scope::TimerNode` 的协作图:



Public 成员函数

- `TimerNode` (`TimerNode &f`)
- `TimerNode` (`TimerNode &&f`)
- `virtual ~TimerNode ()`

Public 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` (`ProgramNode &f`)
- `ProgramNode` (`ProgramNode &&f`)
- `virtual ~ProgramNode ()`
- `ProgramNodeContributionPtr` `getProgramNodeContribution ()`
 获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `ScriptWriter::writeChildren`。

Private 成员函数

- `TimerNode ()`

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode\(\)](#)

11.133.1 详细描述

在文件 [timer_node.h](#) 第 10 行定义.

11.133.2 构造及析构函数说明

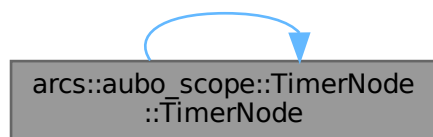
TimerNode() [1/3]

```
arcs::aubo_scope::TimerNode::TimerNode (  
    TimerNode & f)
```

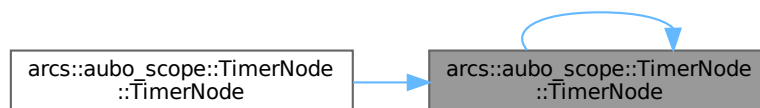
引用了 [TimerNode\(\)](#).

被这些函数引用 [TimerNode\(\)](#) , 以及 [TimerNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

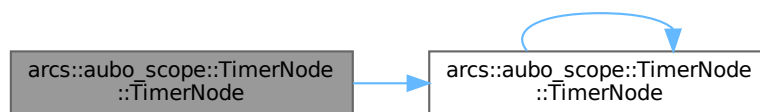


TimerNode() [2/3]

```
arcs::aubo_scope::TimerNode::TimerNode (  
    TimerNode && f)
```

引用了 [TimerNode\(\)](#).

函数调用图:



~TimerNode()

```
virtual arcs::aubo_scope::TimerNode::~~TimerNode () [virtual]
```

TimerNode() [3/3]

```
arcs::aubo_scope::TimerNode::TimerNode () [private]
```

11.133.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [timer_node.h](#) 第 18 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.133.4 类成员变量说明

d_

```
void* arcs::aubo_scope::TimerNode::d_ { nullptr } [private]
```

在文件 [timer_node.h](#) 第 20 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/timer_node.h](#)

11.134 arcs::aubo__scope::TreeNode 类参考

树节点此接口表示程序树中的一个节点，可用于构建以 AuboCap 程序节点为根的子树。

```
#include <tree_node.h>
```

Public 成员函数

- [TreeNode](#) (TreeNode &f)
- [TreeNode](#) (TreeNode &&f)
- virtual [~TreeNode](#) ()
- [TreeNodePtr addChild](#) (ProgramNodePtr program_node)
向子树添加一个子程序节点。
- [TreeNodePtr insertChildBefore](#) (TreeNodePtr existingChildNode, ProgramNodePtr program_node)
在子树中，在现有选中子节点之前插入一个子程序节点。将选中子节点及其后的节点移到新添加子节点之后的位置。
- [TreeNodePtr insertChildAfter](#) (TreeNodePtr existingChildNode, ProgramNodePtr program_node)
在子树中，在现有选中子节点之后插入一个子程序节点。将后续节点移到新添加子节点之后的位置。
- [TreeNodePtr cutChildNode](#) (TreeNodePtr child)

剪切子树中的一个子程序节点。

- bool [removeChild](#) (TreeNodePtr program_node)
从子树中移除子节点。注意：移除最后一个子节点将触发插入一个 *<empty>* 子节点。
- std::vector< TreeNodePtr > [getChildren](#) ()
- std::vector< TreeNodePtr > [getChildrenNotSuppressed](#) ()
- TreeNodePtr [getParent](#) ()
- ProgramNodePtr [getProgramNode](#) ()
- bool [isSuppressed](#) ()
- bool [isInProgramTree](#) ()
节点是否在程序树中；不在程序树中的节点可能已被剪切或删除。
- TreeNodePtr [locateDescendantTreeNode](#) (ProgramNodePtr program_node)
- void [setChildSequenceLocked](#) (bool isChildSequenceLocked)
配置子节点是否可被终端用户重新排列、删除或将其他节点插入到子节点序列中。
- int [traverse](#) (std::function< int(ProgramNodePtr, int, int)> nodeVisitor)

Private 成员函数

- [TreeNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.134.1 详细描述

树节点此接口表示程序树中的一个节点，可用于构建以 AuboCap 程序节点为根的子树。

使用 `ProgramModel#getRootTreeNode(ProgramNodeContribution)` 获取子树的根节点后，即可添加子节点。每次调用 `TreeNode#addChild(ProgramNode)` 都会返回一个新的 [TreeNode](#)，该节点又可以作为另一个子树的根节点。

在文件 `tree_node.h` 第 32 行定义。

11.134.2 构造及析构函数说明

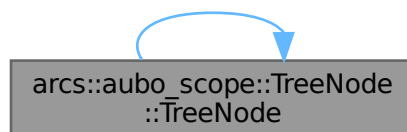
`TreeNode()` [1/3]

```
arcs::aubo_scope::TreeNode::TreeNode (
    TreeNode & f)
```

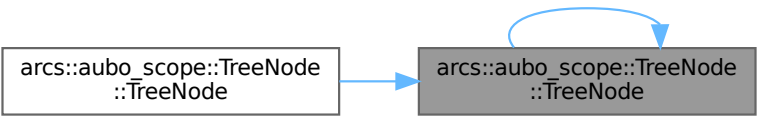
引用了 [TreeNode\(\)](#)。

被这些函数引用 [TreeNode\(\)](#)，以及 [TreeNode\(\)](#)。

函数调用图：

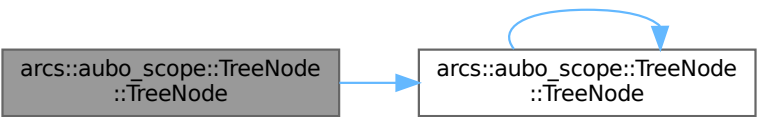


这是这个函数的调用关系图:



TreeNode() [2/3]

```
arcs::aubo_scope::TreeNode::TreeNode (  
    TreeNode && f)  
引用了 TreeNode\(\).  
函数调用图:
```



~TreeNode()

```
virtual arcs::aubo_scope::TreeNode::~~TreeNode () [virtual]
```

TreeNode() [3/3]

```
arcs::aubo_scope::TreeNode::TreeNode () [private]
```

11.134.3 成员函数说明

addChild()

```
TreeNodePtr arcs::aubo_scope::TreeNode::addChild (  
    ProgramNodePtr program_node)  
向子树添加一个子程序节点。
```

参数

<code>program_node</code>	使用 ProgramNodeFactory 构建的 ProgramNode

返回

返回一个 `TreeNode`，可用于向新添加的子节点添加子节点。

异常

<i>TreeStructureException</i>	如果不允许在此位置插入 ProgramNode
<i>IllegalStateException</i>	如果在 UndoableChanges 范围外从基于 Qt 的 AuboCap 调用

cutChildNode()

```
TreeNodePtr arcs::aubo_scope::TreeNode::cutChildNode (  
    TreeNodePtr child)  
剪切子树中的一个子程序节点。
```

参数

<i>child</i>	此 TreeNode 的现有 TreeNode 子节点

返回

返回被剪切掉的 [TreeNode](#)

异常

<i>TreeStructureException</i>	如果不允许剪切此 ProgramNode 或子节点不是此 TreeNode 的子节点
<i>IllegalStateException</i>	如果在 UndoableChanges 范围外调用

getChildren()

```
std::vector< TreeNodePtr > arcs::aubo_scope::TreeNode::getChildren ()
```

返回

[TreeNode](#) 对象列表，表示此 [TreeNode](#) 的所有子节点（包括以编程方式添加的和终端用户插入的）

getChildrenNotSuppressed()

```
std::vector< TreeNodePtr > arcs::aubo_scope::TreeNode::getChildrenNotSuppressed ()
```

getParent()

```
TreeNodePtr arcs::aubo_scope::TreeNode::getParent ()
```

返回

节点的父 [TreeNode](#)

getProgramNode()

```
ProgramNodePtr arcs::aubo_scope::TreeNode::getProgramNode ()
```

返回

返回子树中此位置的 ProgramNode。可以是内置 AuboScope 程序节点或 AuboCap 程序节点。

insertChildAfter()

```
TreeNodePtr arcs::aubo_scope::TreeNode::insertChildAfter (  
    TreeNodePtr existingChildNode,  
    ProgramNodePtr program_node)
```

在子树中，在现有选中子节点之后插入一个子程序节点。将后续节点移到新添加子节点之后的位置。

参数

<i>existingChildNode</i>	此 TreeNode 的现有 TreeNode 子节点
<i>program_node</i>	使用 ProgramNodeFactory 构建的 ProgramNode

返回

返回一个 [TreeNode](#)，可用于向新添加的子节点添加子节点。

异常

<i>TreeStructureException</i>	如果不允许在此位置插入或选中子节点不是此 TreeNode 的子节点
<i>IllegalStateException</i>	如果在 UndoableChanges 范围外调用

insertChildBefore()

```
TreeNodePtr arcs::aubo_scope::TreeNode::insertChildBefore (  
    TreeNodePtr existingChildNode,  
    ProgramNodePtr program_node)
```

在子树中，在现有选中子节点之前插入一个子程序节点。将选中子节点及其后的节点移到新添加子节点之后的位置。

参数

<i>existingChildNode</i>	此 TreeNode 的现有 TreeNode 子节点
<i>program_node</i>	使用 ProgramNodeFactory 构建的 ProgramNode

返回

返回一个 [TreeNode](#)，可用于向新添加的子节点添加子节点。

异常

<i>TreeStructureException</i>	如果不允许在此位置插入或选中子节点不是此 TreeNode 的子节点
<i>IllegalStateException</i>	如果在 UndoableChanges 范围外调用

isInProgramTree()

```
bool arcs::aubo_scope::TreeNode::isInProgramTree ()
```

节点是否在程序树中；不在程序树中的节点可能已被剪切或删除。

返回

`true` 节点在程序树中, `false` 节点不在程序树中 (可能已被剪切或删除)

isSuppressed()

```
bool arcs::aubo_scope::TreeNode::isSuppressed ()
```

locateDescendantTreeNode()

```
TreeNodePtr arcs::aubo_scope::TreeNode::locateDescendantTreeNode (  
    ProgramNodePtr program_node)
```

获取此 [TreeNode](#) 下子树中子程序节点 (ProgramNode 实例) 对应的 [TreeNode](#) 实例。
此方法可用于在遍历父节点子树时, 获取遇到子程序节点下的子树访问权限。

参数

<code>program_node</code>	需要获取对应 TreeNode 表示的程序节点, 不能为 <code>null</code>

返回

指定程序节点的 [TreeNode](#) 实例

异常

<code>ProgramNodeNotInSubTreeException</code>	当程序节点不在子树中时

removeChild()

```
bool arcs::aubo_scope::TreeNode::removeChild (  
    TreeNodePtr program_node)
```

从子树中移除子节点。注意: 移除最后一个子节点将触发插入一个 `<empty>` 子节点。

参数

<code>program_node</code>	要移除的 TreeNode 子节点

返回

成功移除返回 `true`, 否则返回 `false`

异常

<code>TreeStructureException</code>	如果移除子节点会使树处于非法状态
<code>IllegalStateException</code>	如果在 <code>UndoableChanges</code> 范围外调用

setChildSequenceLocked()

```
void arcs::aubo_scope::TreeNode::setChildSequenceLocked (  
    bool isChildSequenceLocked)
```

配置子节点是否可被终端用户重新排列、删除或将其他节点插入到子节点序列中。

参数

<i>isChildSequenceLocked</i>	如果为 <code>true</code> ，则此 TreeNode 下的直接子节点将被锁定

traverse()

```
int arcs::aubo_scope::TreeNode::traverse (
    std::function< int(ProgramNodePtr, int, int)> nodeVisitor)
```

此方法以深度优先方式遍历此树节点下的整个子树（对应于程序树中的自顶向下方式）。

使用节点访问器进行回调，回调的重载取决于遇到的节点类型。

所有 `visit` 方法都具有程序节点、兄弟索引和深度作为参数，以便在需要时过滤节点。

参数

<i>nodeVisitor</i>	回调实例

返回

0 表示正常退出，非零表示异常或错误导致提前退出

11.134.4 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [tree_node.h](#) 第 342 行定义。

引用了 [DataSwitch](#)。

被这些函数引用 [DataSwitch](#)。

11.134.5 类成员变量说明**d_**

```
void* arcs::aubo_scope::TreeNode::d_ { nullptr } [private]
```

在文件 [tree_node.h](#) 第 344 行定义。

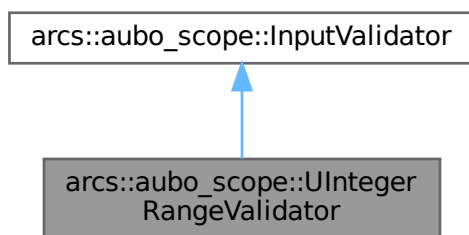
该类的文档由以下文件生成:

- `include/aubo_caps/domain/program/structure/tree_node.h`

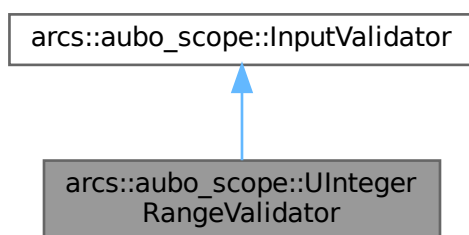
11.135 arcs::aubo_scope::UIntegerRangeValidator 结构体参考

```
#include <input_validator.h>
```


类 arcs::aubo_scope::UIntegerRangeValidator 继承关系图:



arcs::aubo_scope::UIntegerRangeValidator 的协作图:



Public 成员函数

- [UIntegerRangeValidator](#) (uint32_t min_value, uint32_t max_value)
- [UIntegerRangeValidator](#) (std::function< std::pair< uint32_t, uint32_t >()> range_func)
- bool [isValid](#) (std::string value) override
验证输入值是否有效。
- std::string [getMessage](#) (std::string value) override
当值无效时返回有意义的提示信息。

Private 成员函数

- void [updateValue](#) ()

Private 属性

- uint32_t [min_value_](#) { 0 }
- uint32_t [max_value_](#) { UINT_MAX }
- std::function< std::pair< uint32_t, uint32_t >()> [range_func_](#) { nullptr }

11.135.1 详细描述

在文件 [input_validator.h](#) 第 97 行定义.

11.135.2 构造及析构造函数说明

UIntegerRangeValidator() [1/2]

```
arcs::aubo_scope::UIntegerRangeValidator::UIntegerRangeValidator (
    uint32_t min_value,
    uint32_t max_value)
```

UIntegerRangeValidator() [2/2]

```
arcs::aubo_scope::UIntegerRangeValidator::UIntegerRangeValidator (
    std::function< std::pair< uint32_t, uint32_t >()> range_func)
```

11.135.3 成员函数说明

getMessage()

```
std::string arcs::aubo_scope::UIntegerRangeValidator::getMessage (
    std::string value) [override], [virtual]
```

当值无效时返回有意义的提示信息。

参数

<i>value</i>	无效的值，如果合理可包含在消息中

返回

提示信息

实现了 [arcs::aubo_scope::InputValidator](#).

isValid()

```
bool arcs::aubo_scope::UIntegerRangeValidator::isValid (
    std::string value) [override], [virtual]
```

验证输入值是否有效。

参数

<i>value</i>	需要验证的值

返回

如果值有效则返回 `true`

实现了 [arcs::aubo_scope::InputValidator](#).

updateValue()

```
void arcs::aubo_scope::UIntegerRangeValidator::updateValue () [private]
```

11.135.4 类成员变量说明

max__value__

```
uint32_t arcs::aubo_scope::UIntegerRangeValidator::max_value_ { UINT_MAX } [private]
```

在文件 [input_validator.h](#) 第 113 行定义。

min_value__

```
uint32_t arcs::aubo_scope::UIntegerRangeValidator::min_value_ { 0 } [private]
```

在文件 [input_validator.h](#) 第 112 行定义.

range_func__

```
std::function<std::pair<uint32_t, uint32_t>()> arcs::aubo_scope::UIntegerRangeValidator::range_func_←  
{ nullptr } [private]
```

在文件 [input_validator.h](#) 第 115 行定义.

该结构体的文档由以下文件生成:

- [include/aubo_caps/domain/user_interface/input_validator.h](#)

11.136 arcs::aubo_scope::UndoRedoManager 类参考

撤销/重做管理器此接口用于记录终端用户在 AuboCap 程序节点贡献画面中触发的更改（例如，点击按钮时）。应在 AuboCap 程序节点中使用，以支持 AuboScope 中的撤销/重做功能。

```
#include <undo_redo_manager.h>
```

Public 成员函数

- [UndoRedoManager](#) (UndoRedoManager &f)
- [UndoRedoManager](#) (UndoRedoManager &&f)
- virtual [~UndoRedoManager](#) ()
- void [recordChanges](#) (const std::string &text, std::function< void()> undoableChanges)

Private 成员函数

- [UndoRedoManager](#) ()

Private 属性

- void * [d__](#) { nullptr }

友元

- class [DataSwitch](#)

11.136.1 详细描述

撤销/重做管理器此接口用于记录终端用户在 AuboCap 程序节点贡献画面中触发的更改（例如，点击按钮时）。应在 AuboCap 程序节点中使用，以支持 AuboScope 中的撤销/重做功能。

注意：请勿对 AuboCap 安装节点使用此功能，因为它不受支持（且无效）。

注意：此功能不适用于具有基于 HTML 的用户界面的 AuboCap 程序节点，因为这些节点已支持自动撤销。

在文件 [undo_redo_manager.h](#) 第 35 行定义.

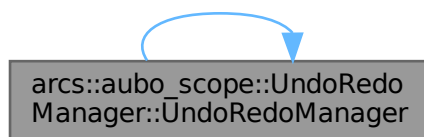
11.136.2 构造及析构函数说明**UndoRedoManager() [1/3]**

```
arcs::aubo_scope::UndoRedoManager::UndoRedoManager (  
    UndoRedoManager & f)
```

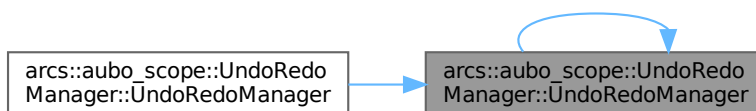
引用了 [UndoRedoManager\(\)](#).

被这些函数引用 [UndoRedoManager\(\)](#) , 以及 [UndoRedoManager\(\)](#).

函数调用图:



这是这个函数的调用关系图:

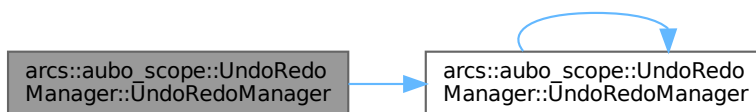


UndoRedoManager() [2/3]

```
arcs::aubo_scope::UndoRedoManager::UndoRedoManager (
    UndoRedoManager && f)
```

引用了 [UndoRedoManager\(\)](#).

函数调用图:



~UndoRedoManager()

```
virtual arcs::aubo_scope::UndoRedoManager::~~UndoRedoManager () [virtual]
```

UndoRedoManager() [3/3]

```
arcs::aubo_scope::UndoRedoManager::UndoRedoManager () [private]
```

11.136.3 成员函数说明

recordChanges()

```
void arcs::aubo_scope::UndoRedoManager::recordChanges (
```

```
const std::string & text,
std::function< void()> undoableChanges)
```

将一组变更归并为一个撤销/重做步骤。

目前撤销/重做支持程序树和数据模型的更改。

参数

<i>undoableChanges</i>	被记录为一个撤销/重做步骤的代码块。在AuboScope 中按下” 撤销” 将撤销此代码块内对程序树和数据模型所做的所有更改。

11.136.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [undo_redo_manager.h](#) 第 69 行定义。

引用了 [DataSwitch](#)。

被这些函数引用 [DataSwitch](#)。

11.136.5 类成员变量说明

d_

```
void* arcs::aubo_scope::UndoRedoManager::d_ { nullptr } [private]
```

在文件 [undo_redo_manager.h](#) 第 71 行定义。

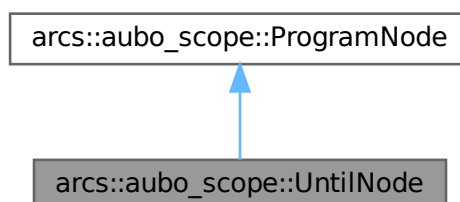
该类的文档由以下文件生成:

- [include/aubo_caps/domain/undoredo/undo_redo_manager.h](#)

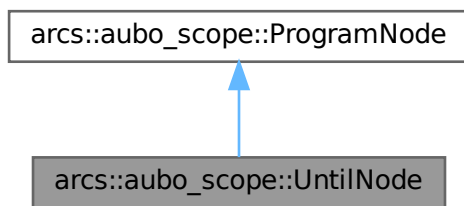
11.137 arcs::aubo_scope::UntilNode 类参考

```
#include <until_node.h>
```

类 arcs::aubo_scope::UntilNode 继承关系图:



arcs::aubo_scope::UntilNode 的协作图:



Public 类型

- enum `ConfigType` : short { `EXPRESSION` , `TOOL_CONTACT` , `IO_INPUT` , `DISTANCE` , `REACHED_WAYPOINT` , `NONE_CONFIG_TYPE` }
The configuration type used to determine which type of configuration this instance is.
- enum `CompareOperator` : int { `LESS_THAN` , `GREATER_THAN` }
Compare operator types LESS_THAN and GREATER_THAN used when waiting for an analog input to go lower or higher, respectively, than a threshold
- enum `BlendParameterType` { `NO_BLEND` , `CUSTOM_BLEND` }
Blend Parameters can be of different types The parameter type is used to determine the explicit blend parameters (can be used in place of instanceof operations).
- enum `CustomDecelerationParameterType` { `Unknown` , `Angular` , `Cartesian` }
- enum `DecelerationParameterType` { `SHARED_DECELERATION` , `CUSTOM_DECELERATION` }
- enum `IOInputSelectionType` { `NONE_INPUT_SELECT` , `DIGITAL_INPUT` , `ANALOG_INPUT` , `FLOAT_REGISTER_INPUT` }
The type of the input selection.

Public 成员函数

- `UntilNode` (`UntilNode &f`)
- `UntilNode` (`UntilNode &&f`)
- virtual `~UntilNode` ()
- `ConfigType` `getConfigType` ()
This method returns the type of configuration.
- void `setConfigType` (`ConfigType` type)
- `BlendParameterType` `getBlendParameterType` ()
- void `setBlendParameterType` (`BlendParameterType` type)
- void `setWaypointBlendParameter` (double suggested_blend)
Creates blend parameters for an Until node.
- double `getWaypointBlendParameter` ()
- `DecelerationParameterType` `getDecelerationParameterType` ()
- void `setDecelerationParameterType` (`DecelerationParameterType` type)
- void `setDecelerationParameters` (double deceleration)
- double `getDecelerationParameters` ()
- `CustomDecelerationParameterType` `getCustomDecelerationParameterType` ()
- void `setCustomDecelerationParameterType` (`CustomDecelerationParameterType` type)

- double [getAngularDeceleration](#) ()
The rate at which to decelerate when the until condition is met.
- void [setAngularDeceleration](#) (double dec)
- double [getCartesianDeceleration](#) ()
The rate at which to decelerate when the until condition is met.
- void [setCartesianDeceleration](#) (double dec)
- void [setExpressionConfig](#) (ExpressionPtr expression)
- ExpressionPtr [getExpressionConfig](#) ()
- double [getDistance](#) ()
The distance the robot will travel.
- void [setDistance](#) (double dis)
- [IOInputSelectionType](#) [getIOInputSelectionType](#) ()
- void [setIOInputSelectionType](#) ([IOInputSelectionType](#) type)
- void [setDigitalInputValueToProceed](#) (bool value_to_proceed_until)
Creates an I/O Input configuration for an Until node.
- bool [getDigitalInputValueToProceed](#) ()
- void [setAnalogInputValueToProceed](#) (double value)
Creates an I/O Input configuration for an Until node.
- double [getAnalogInputValueToProceed](#) ()
- void [setCompareOperator](#) ([CompareOperator](#) compare_operator)
- [CompareOperator](#) [getCompareOperator](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ([ProgramNode](#) &f)
- [ProgramNode](#) ([ProgramNode](#) &&f)
- virtual [~ProgramNode](#) ()
- [ProgramNodeContributionPtr](#) [getProgramNodeContribution](#) ()
获取程序节点贡献。不推荐用户直接调用 [ProgramNodeContribution::generateScript](#), 应使用 [Script←Writer::writeChildren](#)。

Private 成员函数

- [UntilNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.137.1 详细描述

Until 节点是一种必须放在 [Waypoint](#) 节点或 [Direction](#) 节点下的程序节点类型。

当调用 [TreeNode::addChild](#) 和 [#removeChild](#) 时, Until 节点具有特殊行为。默认情况下 Until 节点是叶子节点, 但可以向其添加动作。当添加动作时, 它们会转换为允许子节点的分支节点。当最后一个子节点被移除时, 它们又变回叶子节点。

在文件 [until_node.h](#) 第 31 行定义。

11.137.2 成员枚举类型说明

BlendParameterType

```
enum arcs::aubo_scope::UntilNode::BlendParameterType
```

Blend Parameters can be of different types The parameter type is used to determine the explicit blend parameters (can be used in place of instanceof operations).

枚举值

NO_BLEND	The robot arm will stop without performing a blend.
CUSTOM_BLEND	Custom blend has been selected. The blend instance can be cast to CustomBlendParameters.

在文件 [until_node.h](#) 第 111 行定义.

CompareOperator

```
enum arcs::aubo_scope::UntilNode::CompareOperator : int
```

Compare operator types LESS_THAN and GREATER_THAN used when waiting for an analog input to go lower or higher, respectively, than a threshold

枚举值

LESS_THAN	
GREATER_THAN	

在文件 [until_node.h](#) 第 100 行定义.

ConfigType

```
enum arcs::aubo_scope::UntilNode::ConfigType : short
```

The configuration type used to determine which type of configuration this instance is.

枚举值

EXPRESSION	An Until node where the until condition is true once the evaluation of an expression is true. The config instance can be cast to ExpressionUntilNodeConfig.
TOOL_CONTACT	An Until node where the until condition is true once an impact on the Tool is detected. This functionality is not supported by the API.
IO_INPUT	An Until node where the until condition is true once a given I/O reaches a certain value. The config can be cast to IOInputUntilNodeConfig.
DISTANCE	An Until node where the until condition is true once a given distance is reached. The config can be cast to DistanceUntilNodeConfig.
REACHED_WAYPOINT	An Until node where the until condition is true once its parent Waypoint is reached. The config can be cast to ReachedWaypointUntilNodeConfig.
NONE_CONFIG_TYPE	An undefined Until node with no type set yet.

在文件 [until_node.h](#) 第 38 行定义.

CustomDecelerationParameterType

```
enum arcs::aubo_scope::UntilNode::CustomDecelerationParameterType
```

枚举值

Unknown	This deceleration was either created specifying values for both AngularAcceleration and Acceleration and/or is in use in an Until node that is not yet in the program tree.
Angular	This deceleration is currently used under a MoveJ node.
Cartesian	This deceleration is currently used under a MoveL or MoveP node.

在文件 [until_node.h](#) 第 129 行定义.

DecelerationParameterType

```
enum arcs::aubo_scope::UntilNode::DecelerationParameterType
```

The deceleration parameter type used to determine which type of deceleration parameter this instance is.

枚举值

SHARED_DECELERATION	Shared deceleration has been selected. The acceleration values from the parent Move node will be used as deceleration (absolute value).
CUSTOM_DECELERATION	Custom deceleration has been selected. This instance can be cast to CustomDecelerationParameters .

在文件 [until_node.h](#) 第 154 行定义.

IOInputSelectionType

```
enum arcs::aubo_scope::UntilNode::IOInputSelectionType
```

The type of the input selection.

枚举值

NONE_INPUT_SELECT	The node is not fully configured.
DIGITAL_INPUT	Digital input is selected. The config instance can be cast to DigitalInputUntilNodeConfig .
ANALOG_INPUT	Analog input is selected. The config instance can be cast to AnalogInputUntilNodeConfig .
FLOAT_REGISTER_INPUT	A float input register is selected. The config instance can be cast to FloatRegisterInputUntilNodeConfig .

在文件 [until_node.h](#) 第 175 行定义.

11.137.3 构造及析构函数说明

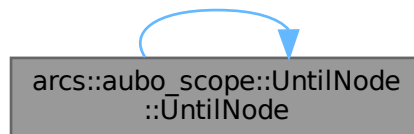
UntilNode() [1/3]

```
arcs::aubo_scope::UntilNode::UntilNode (  
    UntilNode & f)
```

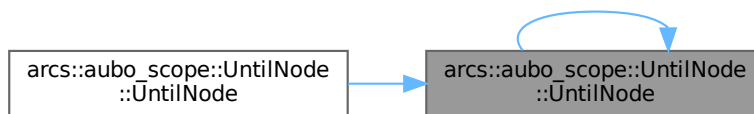
引用了 [UntilNode\(\)](#).

被这些函数引用 [UntilNode\(\)](#) , 以及 [UntilNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

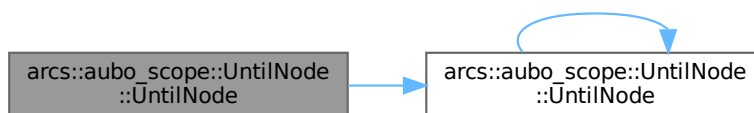


UntilNode() [2/3]

```
arcs::aubo_scope::UntilNode::UntilNode (  
    UntilNode && f)
```

引用了 [UntilNode\(\)](#).

函数调用图:



~UntilNode()

```
virtual arcs::aubo_scope::UntilNode::~~UntilNode () [virtual]
```

UntilNode() [3/3]

```
arcs::aubo_scope::UntilNode::UntilNode () [private]
```

11.137.4 成员函数说明

getAnalogInputValueToProceed()

```
double arcs::aubo_scope::UntilNode::getAnalogInputValueToProceed ()
```

getAngularDeceleration()

```
double arcs::aubo_scope::UntilNode::getAngularDeceleration ()
```

The rate at which to decelerate when the until condition is met.

返回

the specified angular deceleration of the until node (absolute value).

getBlendParameterType()

```
BlendParameterType arcs::aubo_scope::UntilNode::getBlendParameterType ()
```

getCartesianDeceleration()

```
double arcs::aubo_scope::UntilNode::getCartesianDeceleration ()
```

The rate at which to decelerate when the until condition is met.

返回

the specified Cartesian deceleration of the until node (absolute value).

getCompareOperator()

```
CompareOperator arcs::aubo_scope::UntilNode::getCompareOperator ()
```

getConfigType()

```
ConfigType arcs::aubo_scope::UntilNode::getConfigType ()
```

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

返回

the type of this config.

getCustomDecelerationParameterType()

```
CustomDecelerationParameterType arcs::aubo_scope::UntilNode::getCustomDecelerationParameterType ()
```

返回

the deceleration type. This can change depending on the Move node type above the Until node.

getDecelerationParameters()

```
double arcs::aubo_scope::UntilNode::getDecelerationParameters ()
```

getDecelerationParameterType()

```
DecelerationParameterType arcs::aubo_scope::UntilNode::getDecelerationParameterType ()
```

getDigitalInputValueToProceed()

```
bool arcs::aubo_scope::UntilNode::getDigitalInputValueToProceed ()
```

getDistance()

```
double arcs::aubo_scope::UntilNode::getDistance ()
```

The distance the robot will travel.

返回

The distance.

getExpressionConfig()

```
ExpressionPtr arcs::aubo_scope::UntilNode::getExpressionConfig ()
```

getIOInputSelectionType()

```
IOInputSelectionType arcs::aubo_scope::UntilNode::getIOInputSelectionType ()
```

getWaypointBlendParameter()

```
double arcs::aubo_scope::UntilNode::getWaypointBlendParameter ()
```

setAnalogInputValueToProceed()

```
void arcs::aubo_scope::UntilNode::setAnalogInputValueToProceed (
    double value)
```

Creates an I/O Input configuration for an Until node.

The until condition for this configuration is true once the the signal value of a given analog input goes past a specified electric Current threshold.

参数

<i>input</i>	the analog input whose signal value will determine whether the until condition is met, not <code>null</code> .
<i>compareOperator</i>	the operator to use when comparing the signal value of the selected analog input to the threshold value, not <code>null</code> . Available options are "less than" (<) or "greater than" (>).
<i>currentThreshold</i>	the electric Current threshold for the signal value of the analog input. Execution will proceed until the signal value of the selected analog input goes past this threshold.
<i>errorHandler</i>	the error handler for validation errors. If using <code>ErrorHandler#AUTO_CORRECT</code> this will clamp the value to the nearest valid current value.

返回

the Until node configuration.

异常

<i>IllegalArgumentException</i>	if <i>input</i> is an output (does not support reading of values). See <code>IO#isInput()</code> .

setAngularDeceleration()

```
void arcs::aubo_scope::UntilNode::setAngularDeceleration (
    double dec)
```

setBlendParameterType()

```
void arcs::aubo_scope::UntilNode::setBlendParameterType (
    BlendParameterType type)
```

setCartesianDeceleration()

```
void arcs::aubo_scope::UntilNode::setCartesianDeceleration (
    double dec)
```

setCompareOperator()

```
void arcs::aubo_scope::UntilNode::setCompareOperator (
    CompareOperator compare_operator)
```

setConfigType()

```
void arcs::aubo_scope::UntilNode::setConfigType (
    ConfigType type)
```

setCustomDecelerationParameterType()

```
void arcs::aubo_scope::UntilNode::setCustomDecelerationParameterType (
    CustomDecelerationParameterType type)
```

setDecelerationParameters()

```
void arcs::aubo_scope::UntilNode::setDecelerationParameters (
    double deceleration)
```

Creates deceleration parameter for an Until node. Can be used in an [Expression](#) configuration type.

An angular deceleration is used when the type of the parent Move node is a MoveJ, whereas a Cartesian deceleration is used when the move type is a MoveL or MoveP.

参数

<i>deceleration</i>	the angular deceleration (the value for the Cartesian deceleration will be the default value that AuboScope uses), not null .
<i>decelerationErrorHandler</i>	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid angular deceleration value.

返回

the deceleration parameter for the Until node.

setDecelerationParameterType()

```
void arcs::aubo_scope::UntilNode::setDecelerationParameterType (
    DecelerationParameterType type)
```

setDigitalInputValueToProceed()

```
void arcs::aubo_scope::UntilNode::setDigitalInputValueToProceed (
    bool value_to_proceed_until)
```

Creates an I/O Input configuration for an Until node.

The until condition for this configuration is true once the signal value of a given digital input goes HIGH or LOW.

参数

<i>input</i>	the digital input whose signal value will determine whether the until condition is met, not null .
<i>valueToProceedUntil</i>	true , if the until condition is met when the input signal value goes HIGH, false otherwise (condition met when signal value is LOW).

返回

the Until node configuration.

异常

<i>IllegalArgumentException</i>	if input is an output (does not support reading of values). See I0#isInput() .

setDistance()

```
void arcs::aubo_scope::UntilNode::setDistance (
    double dis)
```

setExpressionConfig()

```
void arcs::aubo_scope::UntilNode::setExpressionConfig (
    ExpressionPtr expression)
```

Creates an [Expression](#) configuration for an Until node with an expression and default value for the deceleration parameter. The until condition for this configuration is true once the evaluation of the expression is true.

Note that this default value may change between AuboScope versions.

参数

<i>expression</i>	the expression whose evaluation will determine whether the until condition is met.

返回

the Until node configuration.

setIOInputSelectionType()

```
void arcs::aubo_scope::UntilNode::setIOInputSelectionType (
    IOInputSelectionType type)
```

setWaypointBlendParameter()

```
void arcs::aubo_scope::UntilNode::setWaypointBlendParameter (
    double suggested_blend)
```

Creates blend parameters for an Until node.

Can be used in a Distance configuration type.

参数

<i>suggestedBlend</i>	This blend value may be (automatically) corrected depending on the value of the distance parameter the Distance configuration.

返回

the blend parameters for the Until node.

11.137.5 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [until_node.h](#) 第 367 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.137.6 类成员变量说明**d_**

```
void* arcs::aubo_scope::UntilNode::d_ { nullptr } [private]
```

在文件 [until_node.h](#) 第 369 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/until_node.h](#)

11.138 arcs::aubo_scope::UserInteraction 类参考

用户交互接口此接口提供向终端用户请求输入或操作的功能。

```
#include <user_interaction.h>
```

Public 成员函数

- [UserInteraction](#) ([UserInteraction](#) &f)
- [UserInteraction](#) ([UserInteraction](#) &&f)
- virtual [~UserInteraction](#) ()
- void [getUserDefinedRobotPosition](#) ([RobotPositionCallback](#) callback)
- void [getUserDefinedRobotPosition](#) (const std::vector< double > &tcpOffset, [RobotPositionCallback](#) callback)
- 指定 *tcpOffset* 获取用户定义的机器人位置
- void [getUserDefinedRobotPosition](#) (const std::string &tcpOffset, [RobotPositionCallback](#) callback)
- 指定 *tcpOffset* 字符串获取用户定义的机器人位置
- void [getUserDefinedRobotPosition](#) (const TCPPtr tcpOffset, [RobotPositionCallback](#) callback)
- 指定 *TCP* 对象获取用户定义的机器人位置
- KeyboardManagerPtr [getKeyboardManager](#) ()
- int [requestUserToMoveJoint](#) (const std::vector< double > &q, [RobotMovementCallback](#) callback)
- int [requestUserToMoveJoint](#) (const std::string &q, [RobotMovementCallback](#) callback)

requestUserToMoveJoint

- int [requestUserToMoveJoint](#) (const std::vector< double > &pose, const std::vector< double > &q, [RobotMovementCallback](#) callback)
- QT_DEPRECATED int [requestUserToMoveLine](#) (const std::vector< double > &pose, [RobotMovementCallback](#) callback)
- int [requestUserToMoveLine](#) (const std::vector< double > &pose, const std::vector< double > &tcpOffset, [RobotMovementCallback](#) callback)
- QPixmap * [getRobotThumb](#) (const std::vector< double > &q, const std::vector< double > &pose)
获取机器人缩略图
- void [setOperationLogDisplayName](#) (QWidget *pbn, const std::string &name)
- RenderInterfacePtr [getRenderInterface](#) ()
- PopupManagerPtr [getPopupManager](#) ()

Private 成员函数

- [UserInteraction](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.138.1 详细描述

用户交互接口此接口提供向终端用户请求输入或操作的功能。
在文件 [user_interaction.h](#) 第 35 行定义。

11.138.2 构造及析构函数说明

[UserInteraction\(\)](#) [1/3]

```
arcs::aubo_scope::UserInteraction::UserInteraction (
    UserInteraction & f)
```

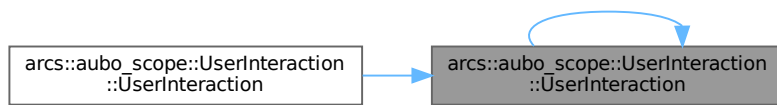
引用了 [UserInteraction\(\)](#).

被这些函数引用 [UserInteraction\(\)](#) , 以及 [UserInteraction\(\)](#).

函数调用图:



这是这个函数的调用关系图:

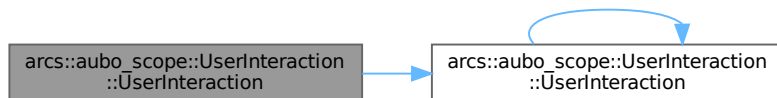


UserInteraction() [2/3]

```
arcs::aubo_scope::UserInteraction::UserInteraction (
    UserInteraction && f)
```

引用了 [UserInteraction\(\)](#).

函数调用图:



~UserInteraction()

```
virtual arcs::aubo_scope::UserInteraction::~~UserInteraction () [virtual]
```

UserInteraction() [3/3]

```
arcs::aubo_scope::UserInteraction::UserInteraction () [private]
```

11.138.3 成员函数说明

getKeyboardManager()

```
KeyboardManagerPtr arcs::aubo_scope::UserInteraction::getKeyboardManager ()
```

This method provides a factory for creating keyboard inputs which are used to configure a virtual keyboard/keypad and to request it to be displayed for a Qt GUI component.

NOTE: This functionality is only relevant for AuboCap nodes with a Qt-based user interface (see [ProgramNodeService](#), [InstallationNodeService](#) and [ToolbarService](#)).

返回

factory providing access to keyboard inputs.

getPopupManager()

```
PopupManagerPtr arcs::aubo_scope::UserInteraction::getPopupManager ()
```

返回

This method provides popup manager.

getRenderInterface()

```
RenderInterfacePtr arcs::aubo_scope::UserInteraction::getRenderInterface ()
```

返回

This method provides robot model drawing objects.

getRobotThumb()

```
QPixmap * arcs::aubo_scope::UserInteraction::getRobotThumb (
    const std::vector< double > & q,
    const std::vector< double > & pose)
```

获取机器人缩略图

参数

<i>q</i>	关节角度
<i>pose</i>	位姿

返回

机器人缩略图

getUserDefinedRobotPosition() [1/4]

```
void arcs::aubo_scope::UserInteraction::getUserDefinedRobotPosition (
    const std::string & tcpOffset,
    RobotPositionCallback callback)
```

指定 tcpOffset 字符串获取用户定义的机器人位置

参数

<i>tcpOffset</i>	TCP 偏移量名称
<i>callback</i>	回调函数

getUserDefinedRobotPosition() [2/4]

```
void arcs::aubo_scope::UserInteraction::getUserDefinedRobotPosition (
    const std::vector< double > & tcpOffset,
    RobotPositionCallback callback)
```

指定 tcpOffset 获取用户定义的机器人位置

参数

<i>tcpOffset</i>	TCP 偏移量
<i>callback</i>	回调函数

getUserDefinedRobotPosition() [3/4]

```
void arcs::aubo_scope::UserInteraction::getUserDefinedRobotPosition (
    const TCPPtr tcpOffset,
    RobotPositionCallback callback)
```

指定 TCP 对象获取用户定义的机器人位置

参数

<i>tcpOffset</i>	TCP 对象
<i>callback</i>	回调函数

getUserDefinedRobotPosition() [4/4]

```
void arcs::aubo_scope::UserInteraction::getUserDefinedRobotPosition (
    RobotPositionCallback callback)
```

Request the end user to use the robot to define a robot position. Override the RobotPositionCallback2#onOk(PositionParameters) method to execute code once the end user is done. Optionally override the RobotPositionCallback2#onCancel() method to execute code if the end user cancels the operation.

This method is asynchronous, i.e. the method will return immediately. Only when the end user is done, either RobotPositionCallback2#onOk(PositionParameters) or RobotPositionCallback2#onCancel() will be called.

NOTE: This functionality should not be used in a toolbar contribution (see ToolbarService).

参数

<i>callback</i>	the instance callbacks will be made to.

requestUserToMoveJoint() [1/3]

```
int arcs::aubo_scope::UserInteraction::requestUserToMoveJoint (
    const std::string & q,
    RobotMovementCallback callback)
```

requestUserToMoveJoint

参数

<i>q</i>	
<i>callback</i>	

返回**requestUserToMoveJoint()** [2/3]

```
int arcs::aubo_scope::UserInteraction::requestUserToMoveJoint (
    const std::vector< double > & pose,
    const std::vector< double > & q,
    RobotMovementCallback callback)
```

参数

<i>pose</i>	The target pose the robot TCP will move to.
<i>q</i>	The reference joint Angle.
<i>callback</i>	The instance which callbacks will be made to.

异常

<i>IllegalStateException</i>	If the plugin screen is not active (visible).

requestUserToMoveJoint() [3/3]

```
int arcs::aubo_scope::UserInteraction::requestUserToMoveJoint (
    const std::vector< double > & q,
    RobotMovementCallback callback)
```

Transition to the move tab widget to allow/request the end user to move the robot (either automatically or manually) to a desired target position specified by joint positions.

The Automove function will move the robot to the target joint positions linearly in joint space.

NOTE: This method does not take the currently active [TCP](#) offset into account, i.e. the Automove function will have the specified joint positions as target.

参数

<i>q</i>	The target joint positions to move to.
<i>callback</i>	The instance which callbacks will be made to.

异常

<i>IllegalStateException</i>	If the plugin screen is not active (visible).

requestUserToMoveLine() [1/2]

```
int arcs::aubo_scope::UserInteraction::requestUserToMoveLine (
    const std::vector< double > & pose,
    const std::vector< double > & tcpOffset,
    RobotMovementCallback callback)
```

Transition to the move tab widget to allow/request the end user to move the robot (either automatically or manually) to a desired target position specified by a pose.

The Automove function will move the robot [TCP](#) to the target position linearly in Cartesian space. If this is not possible, the robot will move to the target linearly in joint space.

The current joint positions of the robot will be used as starting point for inverse kinematics to calculate a target joint vector corresponding to desired pose, taking the currently active [TCP](#) offset into account.

参数

<i>pose</i>	The target pose the robot TCP will move to.

<i>tcpOffset</i>	
<i>callback</i>	The instance which callbacks will be made to.

异常

<i>IllegalStateException</i>	If the plugin screen is not active (visible).

requestUserToMoveLine() [2/2]

```
QT_DEPRECATED int arcs::aubo_scope::UserInteraction::requestUserToMoveLine (
    const std::vector< double > & pose,
    RobotMovementCallback callback)
```

Transition to the move tab widget to allow/request the end user to move the robot (either automatically or manually) to a desired target position specified by a pose.

The Automove function will move the robot [TCP](#) to the target position linearly in Cartesian space. If this it not possible, the robot will move to the target linearly in joint space.

The current joint positions of the robot will be used as starting point for inverse kinematics to calculate a target joint vector corresponding to desired pose, taking the currently active [TCP](#) offset into account.

参数

<i>pose</i>	The target pose the robot TCP will move to.
<i>callback</i>	The instance which callbacks will be made to.

异常

<i>IllegalStateException</i>	If the plugin screen is not active (visible).

setOperationLogDisplayName()

```
void arcs::aubo_scope::UserInteraction::setOperationLogDisplayName (
    QWidget * pbn,
    const std::string & name)
setOperationLogDisplayName 设置操作日志显示名称
```

参数

<i>pbn</i>	界面按钮
<i>name</i>	显示名称

11.138.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 `user_interaction.h` 第 283 行定义.
 引用了 `DataSwitch`.
 被这些函数引用 `DataSwitch`.

11.138.5 类成员变量说明

`d_`

```
void* arcs::aubo_scope::UserInteraction::d_ { nullptr } [private]
```

在文件 `user_interaction.h` 第 285 行定义.

该类的文档由以下文件生成:

- `include/aubo_caps/domain/user_interface/user_interaction.h`

11.139 arcs::aubo_scope::UserInterfaceApi 类参考

```
#include <user_interface_api.h>
```

Public 成员函数

- `UserInterfaceApi` (`UserInterfaceApi &f`)
- `UserInterfaceApi` (`UserInterfaceApi &&f`)
- `virtual ~UserInterfaceApi` ()
- `UserInteractionPtr` `getUserInteraction` ()
UserInteraction

Private 成员函数

- `UserInterfaceApi` ()

Private 属性

- `void *` `d_` { `nullptr` }

友元

- `class` `DataSwitch`

11.139.1 详细描述

在文件 `user_interface_api.h` 第 25 行定义.

11.139.2 构造及析构函数说明

`UserInterfaceApi`() [1/3]

```
arcs::aubo_scope::UserInterfaceApi::UserInterfaceApi (  
    UserInterfaceApi & f)
```

引用了 `UserInterfaceApi`() .

被这些函数引用 `UserInterfaceApi`() , 以及 `UserInterfaceApi`() .

函数调用图:



这是这个函数的调用关系图:



UserInterfaceApi() [2/3]

```
arcs::aubo_scope::UserInterfaceApi::UserInterfaceApi (
    UserInterfaceApi && f)
```

引用了 [UserInterfaceApi\(\)](#).

函数调用图:



~UserInterfaceApi()

```
virtual arcs::aubo_scope::UserInterfaceApi::~~UserInterfaceApi () [virtual]
```

UserInterfaceApi() [3/3]

```
arcs::aubo_scope::UserInterfaceApi::UserInterfaceApi () [private]
```

11.139.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [user_interface_api.h](#) 第 46 行定义.

引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.139.4 类成员变量说明

d_

```
void* arcs::aubo_scope::UserInterfaceApi::d_ { nullptr } [private]
```

在文件 [user_interface_api.h](#) 第 48 行定义.

该类的文档由以下文件生成:

- include/aubo_caps/domain/user_interface/[user_interface_api.h](#)

11.140 arcs::aubo__scope::ValueFactory 类参考

值工厂接口

```
#include <value_factory_provider.h>
```

Public 成员函数

- [ValueFactory](#) (ValueFactory &f)
- [ValueFactory](#) (ValueFactory &&f)
- virtual [~ValueFactory](#) ()
- ExpressionPtr [createExpression](#) ()
实例化一个新的 *ExpressionBuilder* 来创建单个有效表达式。
- ExpressionPtr [createExpression](#) (const std::string &str)

Private 成员函数

- [ValueFactory](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.140.1 详细描述

值工厂接口

概述

此接口提供使用工厂创建不可变值对象的方法。[SimpleValueFactory](#) 帮助创建简单值对象。[ExpressionBuilder](#) 帮助构建有效表达式。[BlendFactory](#) 帮助创建过渡对象。[JointPositionFactory](#) 帮助创建关节位置对象。[PoseFactory](#) 帮助创建姿态对象。

在文件 [value_factory_provider.h](#) 第 32 行定义.

11.140.2 构造及析构函数说明

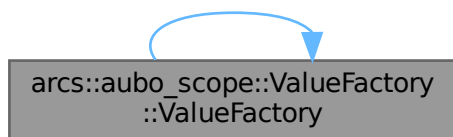
[ValueFactory](#)() [1/3]

```
arcs::aubo_scope::ValueFactory::ValueFactory (  
    ValueFactory & f)
```

引用了 [ValueFactory](#)() .

被这些函数引用 [ValueFactory](#)() , 以及 [ValueFactory](#)() .

函数调用图:



这是这个函数的调用关系图:

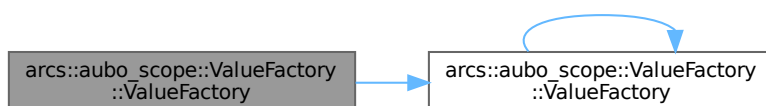


ValueFactory() [2/3]

```
arcs::aubo_scope::ValueFactory::ValueFactory (
    ValueFactory && f)
```

引用了 [ValueFactory\(\)](#).

函数调用图:



~ValueFactory()

```
virtual arcs::aubo_scope::ValueFactory::~ValueFactory () [virtual]
```

ValueFactory() [3/3]

```
arcs::aubo_scope::ValueFactory::ValueFactory () [private]
```

11.140.3 成员函数说明

createExpression() [1/2]

```
ExpressionPtr arcs::aubo_scope::ValueFactory::createExpression ()
```

实例化一个新的ExpressionBuilder来创建单个有效表达式。

返回

表达式构建器。

createExpression() [2/2]

```
ExpressionPtr arcs::aubo_scope::ValueFactory::createExpression (
    const std::string & str)
```

11.140.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [value_factory_provider.h](#) 第 55 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.140.5 类成员变量说明

d_

```
void* arcs::aubo_scope::ValueFactory::d_ { nullptr } [private]
```

在文件 [value_factory_provider.h](#) 第 57 行定义.

该类的文档由以下文件生成:

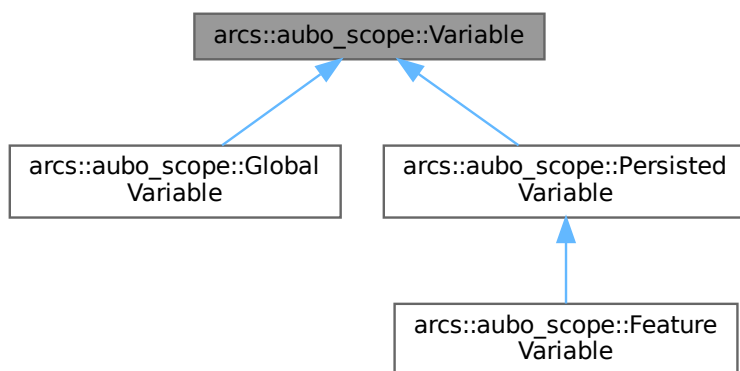
- [include/aubo_caps/domain/value/value_factory_provider.h](#)

11.141 arcs::aubo_scope::Variable 类参考

变量接口表示 AuboScope 中的变量。变量可用于在程序中存储和传递数据。

```
#include <variable.h>
```

类 arcs::aubo_scope::Variable 继承关系图:



Public 类型

- enum [Type](#) : int { [GLOBAL](#) , [VALUE_PERSISTED](#) , [FEATURE](#) }

The variable type used to determine which type of variable this instance is.

Public 成员函数

- [Variable](#) (Variable &f)
- [Variable](#) (Variable &&f)
- virtual [~Variable](#) ()
- [Type](#) [getType](#) ()
获取变量类型
- std::string [getDisplayname](#) ()
获取变量显示名称。注意：变量可随时被重命名。
- bool [equals](#) (VariablePtr object)
判断是否引用相同的内部变量
- size_t [hashCode](#) ()
获取哈希码
- std::string [toString](#) ()
获取变量的字符串表示

Protected 成员函数

- [Variable](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.141.1 详细描述

变量接口表示 AuboScope 中的变量。变量可用于在程序中存储和传递数据。
在文件 [variable.h](#) 第 24 行定义。

11.141.2 成员枚举类型说明**Type**

```
enum arcs::aubo_scope::Variable::Type : int
```

The variable type used to determine which type of variable this instance is.

枚举值

GLOBAL	GLOBAL means that the variable is available everywhere in a program. This variable instance can be cast to GlobalVariable .
VALUE_PERSISTED	VALUE_PERSISTED means that the variable is stored with the installation. These variables are defined in the installation and will retain their values even when a robot is turned off and on. This variable instance can be cast to ValuePersistedVariable .
FEATURE	FEATURE means that a Feature is marked as a variable. The variable is defined in the installation and will exist even when a robot is turned off and on. This variable instance can be cast to FeatureVariable .

在文件 [variable.h](#) 第 35 行定义。

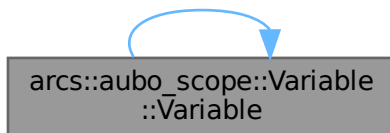
11.141.3 构造及析构函数说明

Variable() [1/3]

```
arcs::aubo_scope::Variable::Variable (  
    Variable & f)
```

引用了 [Variable\(\)](#).

被这些函数引用 [Variable\(\)](#) , 以及 [Variable\(\)](#).
函数调用图:



这是这个函数的调用关系图:

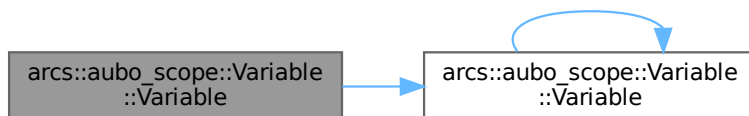


Variable() [2/3]

```
arcs::aubo_scope::Variable::Variable (  
    Variable && f)
```

引用了 [Variable\(\)](#).

函数调用图:



~Variable()

```
virtual arcs::aubo_scope::Variable::~~Variable () [virtual]
```

Variable() [3/3]

```
arcs::aubo_scope::Variable::Variable () [protected]
```

11.141.4 成员函数说明**equals()**

```
bool arcs::aubo_scope::Variable::equals (
    VariablePtr object)
```

判断是否引用相同的内部变量

返回

如果对象引用相同的内部变量返回 true。

getDisplayName()

```
std::string arcs::aubo_scope::Variable::getDisplayName ()
```

获取变量显示名称。注意：变量可随时被重命名。

返回

变量显示名称。

getType()

```
Type arcs::aubo_scope::Variable::getType ()
```

获取变量类型

返回

变量类型枚举。

hashCode()

```
size_t arcs::aubo_scope::Variable::hashCode ()
```

获取哈希码

返回

此对象的哈希码。

toString()

```
std::string arcs::aubo_scope::Variable::toString ()
```

获取变量的字符串表示

返回

变量的字符串表示。

11.141.5 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [variable.h](#) 第 139 行定义。
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.141.6 类成员变量说明

d_

```
void* arcs::aubo_scope::Variable::d_ { nullptr } [private]
```

在文件 [variable.h](#) 第 140 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/variable/variable.h](#)

11.142 arcs::aubo_scope::VariableFactory 类参考

```
#include <variable_factory.h>
```

Public 成员函数

- [VariableFactory](#) (VariableFactory &f)
- [VariableFactory](#) (VariableFactory &&f)
- [~VariableFactory](#) ()
- GlobalVariablePtr [createGlobalVariable](#) (const std::string &suggested_name)
- GlobalVariablePtr [createGlobalVariable](#) (const std::string &suggested_name, ExpressionPtr initialValue)
- FeatureVariablePtr [createFeatureVariable](#) (const std::string &suggested_name, const std::vector<double> &pose)

Private 成员函数

- [VariableFactory](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.142.1 详细描述

This interface supplies methods to create variable objects.

NOTE: Only use this functionality in a AuboCap program node (not relevant for an installation node).

在文件 [variable_factory.h](#) 第 21 行定义.

11.142.2 构造及析构造函数说明

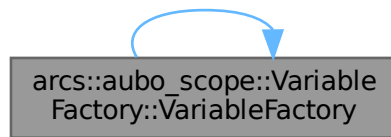
VariableFactory() [1/3]

```
arcs::aubo_scope::VariableFactory::VariableFactory (
    VariableFactory & f)
```

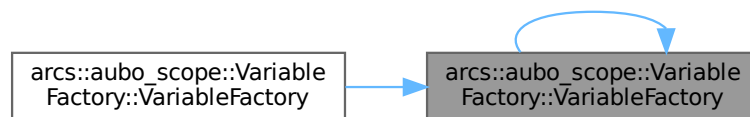
引用了 [VariableFactory\(\)](#).

被这些函数引用 [VariableFactory\(\)](#) , 以及 [VariableFactory\(\)](#).

函数调用图:



这是这个函数的调用关系图:

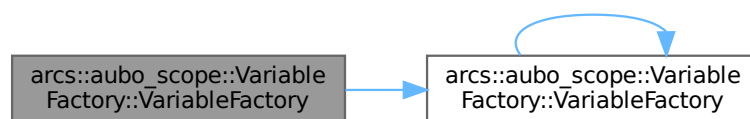


VariableFactory() [2/3]

```
arcs::aubo_scope::VariableFactory::VariableFactory (
    VariableFactory && f)
```

引用了 [VariableFactory\(\)](#).

函数调用图:



~VariableFactory()

```
arcs::aubo_scope::VariableFactory::~~VariableFactory ()
```

VariableFactory() [3/3]

```
arcs::aubo_scope::VariableFactory::VariableFactory () [private]
```

11.142.3 成员函数说明

createFeatureVariable()

```
FeatureVariablePtr arcs::aubo_scope::VariableFactory::createFeatureVariable (  
    const std::string & suggested_name,  
    const std::vector< double > & pose)
```

创建特征变量。

参数

<i>suggested_name</i>	建议的变量名称。
<i>pose</i>	特征的位姿。

返回

FeatureVariable。

异常

<i>VariableException</i>	名称非法时抛出。

createGlobalVariable() [1/2]

```
GlobalVariablePtr arcs::aubo_scope::VariableFactory::createGlobalVariable (  
    const std::string & suggested_name)
```

创建全局变量。在AuboScope 中注册后将对所有程序节点可见。

参数

<i>suggested_name</i>	建议的变量名称。

返回

GlobalVariable。

异常

<i>VariableException</i>	名称非法时抛出。

createGlobalVariable() [2/2]

```
GlobalVariablePtr arcs::aubo_scope::VariableFactory::createGlobalVariable (  
    const std::string & suggested_name,  
    ExpressionPtr initialValue)
```

创建带初始值的全局变量。

参数

<i>suggested_name</i>	建议的变量名称。
<i>initialValue</i>	初始化表达式。

返回

GlobalVariable。

11.142.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [variable_factory.h](#) 第 98 行定义。
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.142.5 类成员变量说明

d_

```
void* arcs::aubo_scope::VariableFactory::d_ { nullptr } [private]
```

在文件 [variable_factory.h](#) 第 100 行定义。
该类的文档由以下文件生成:

- [include/aubo_caps/domain/variable/variable_factory.h](#)

11.143 arcs::aubo_scope::VariableModel 类参考

变量模型提供返回系统中当前已注册变量的方法。变量在存储到[DataModel](#) 实例中或用于配置内置AuboScope 程序节点时即被注册。

```
#include <variable_model.h>
```

Public 成员函数

- [VariableModel](#) (VariableModel &f)
- [VariableModel](#) (VariableModel &&f)
- [~VariableModel](#) ()
- [std::vector< VariablePtr > getAll](#) ()
获取系统中当前已注册的所有变量
- [VariablePtr get](#) (const std::string &name)
- [std::vector< VariablePtr > get](#) (std::function< bool(VariablePtr)> filter=[](VariablePtr) { return true;})
使用过滤器获取已注册变量的子集
- [VariableFactoryPtr getVariableFactory](#) ()
获取变量工厂，用于创建变量对象
- [FeatureVariablePtr getFeatureVariable](#) (FeaturePtr feature)
获取特征对应的特征变量
- [std::string connectedTo](#) (const std::string &receiver, const std::function< void(VariableModel *)> &slot)
注册变更回调
- [void disconnectFrom](#) (const std::string &receiver)
断开回调连接

Private 成员函数

- [VariableModel\(\)](#)

Private 属性

- `void * d\_ { nullptr }`

友元

- `class DataSwitch`

11.143.1 详细描述

变量模型提供返回系统中当前已注册变量的方法。变量在存储到[DataModel](#) 实例中或用于配置内置AuboScope 程序节点时即被注册。

在文件 [variable_model.h](#) 第 29 行定义。

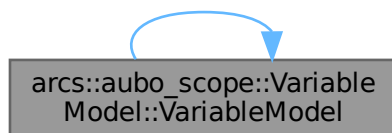
11.143.2 构造及析构函数说明**VariableModel() [1/3]**

```
arcs::aubo_scope::VariableModel::VariableModel (
    VariableModel & f)
```

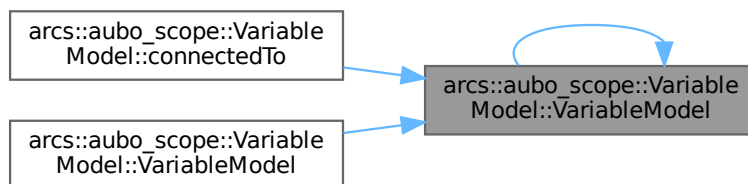
引用了 [VariableModel\(\)](#)。

被这些函数引用 [connectedTo\(\)](#), [VariableModel\(\)](#) , 以及 [VariableModel\(\)](#)。

函数调用图:



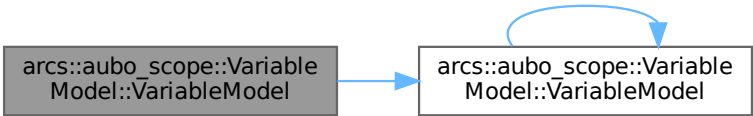
这是这个函数的调用关系图:



VariableModel() [2/3]

```
arcs::aubo_scope::VariableModel::VariableModel (  
    VariableModel && f)
```

引用了 [VariableModel\(\)](#).
函数调用图:



~VariableModel()

```
arcs::aubo_scope::VariableModel::~~VariableModel ()
```

VariableModel() [3/3]

```
arcs::aubo_scope::VariableModel::VariableModel () [private]
```

11.143.3 成员函数说明

connectedTo()

```
std::string arcs::aubo_scope::VariableModel::connectedTo (  
    const std::string & receiver,  
    const std::function< void(VariableModel *)> & slot)
```

注册变更回调

参数

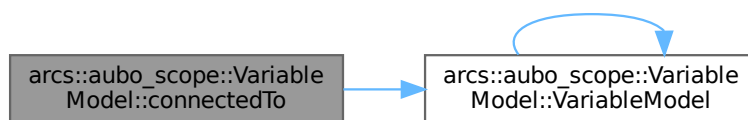
<i>receiver</i>	接收者标识。
<i>slot</i>	回调函数。

返回

连接标识符。

引用了 `VariableModel()`。

函数调用图:



disconnectFrom()

```
void arcs::aubo_scope::VariableModel::disconnectFrom (
    const std::string & receiver)
```

断开回调连接

参数

<i>receiver</i>	接收者标识。

get() [1/2]

```
VariablePtr arcs::aubo_scope::VariableModel::get (
    const std::string & name)
```

get() [2/2]

```
std::vector< VariablePtr > arcs::aubo_scope::VariableModel::get (
    std::function< bool(VariablePtr)> filter = [](VariablePtr) { return true;})
```

使用过滤器获取已注册变量的子集

参数

<i>filter</i>	过滤器。

返回

被过滤器接受的变量集合。

getAll()

```
std::vector< VariablePtr > arcs::aubo_scope::VariableModel::getAll ()
```

获取系统中当前已注册的所有变量

返回
已注册变量的集合。

getFeatureVariable()
`FeatureVariablePtr arcs::aubo_scope::VariableModel::getFeatureVariable (`
 `FeaturePtr feature)`
获取特征对应的特征变量

参数

<i>feature</i>	带有变量的特征。

返回
特征的变量。

异常

<i>IllegalArgumentException</i>	如果特征不在安装中或特征没有变量。

getVariableFactory()
`VariableFactoryPtr arcs::aubo_scope::VariableModel::getVariableFactory ()`
获取变量工厂，用于创建变量对象

返回
变量工厂。

11.143.4 友元及相关符号说明

DataSwitch
`friend class DataSwitch [friend]`
在文件 [variable_model.h](#) 第 133 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.143.5 类成员变量说明

d_
`void* arcs::aubo_scope::VariableModel::d_ { nullptr } [private]`
在文件 [variable_model.h](#) 第 135 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/variable/variable_model.h](#)

11.144 arcs::aubo_scope::ViewApiProvider 类参考

视图 API 提供者提供与安装或程序节点贡献的视图（UI）相关的功能访问。
`#include <view_api_provider.h>`

Public 成员函数

- [ViewApiProvider \(\)](#)
- [ViewApiProvider \(ViewApiProvider &f\)](#)
- [ViewApiProvider \(ViewApiProvider &&f\)](#)
- [virtual ~ViewApiProvider \(\)](#)
- [SystemApiPtr getSystemApi \(\)](#)
提供对系统相关功能的访问。
- [UiThreadApiPtr getUserInterfaceApi \(\)](#)
提供对用户界面和用户交互相关功能的访问。

Private 属性

- `void * d\_ { nullptr }`

友元

- `class DataSwitch`

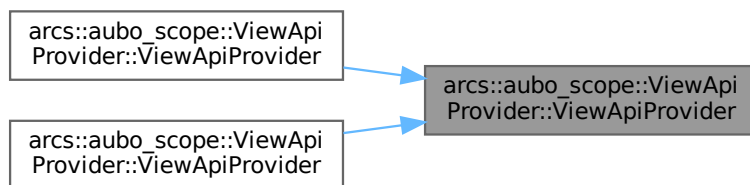
11.144.1 详细描述

视图 API 提供者提供与安装或程序节点贡献的视图 (UI) 相关的功能访问。

注意：此接口不适用于具有基于 HTML 用户界面的 AuboCap 程序和安装节点。
在文件 [view_api_provider.h](#) 第 27 行定义。

11.144.2 构造及析构函数说明**ViewApiProvider() [1/3]**

`arcs::aubo_scope::ViewApiProvider::ViewApiProvider ()`
被这些函数引用 [ViewApiProvider\(\)](#) , 以及 [ViewApiProvider\(\)](#)。
这是这个函数的调用关系图:

**ViewApiProvider() [2/3]**

`arcs::aubo_scope::ViewApiProvider::ViewApiProvider (`
 `ViewApiProvider & f)`
引用了 [ViewApiProvider\(\)](#)。

函数调用图:



ViewApiProvider() [3/3]

```
arcs::aubo_scope::ViewApiProvider::ViewApiProvider (
    ViewApiProvider && f)
```

引用了 [ViewApiProvider\(\)](#).

函数调用图:



~ViewApiProvider()

```
virtual arcs::aubo_scope::ViewApiProvider::~~ViewApiProvider () [virtual]
```

11.144.3 成员函数说明

getSystemApi()

```
SystemApiPtr arcs::aubo_scope::ViewApiProvider::getSystemApi ()
```

提供对系统相关功能的访问。

返回

SystemAPI 实例

getUserInterfaceApi()

```
UserInterfaceApiPtr arcs::aubo_scope::ViewApiProvider::getUserInterfaceApi ()
```

提供对用户界面和用户交互相关功能的访问。

返回

UserInterfaceAPI 实例

11.144.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [view_api_provider.h](#) 第 65 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.144.5 类成员变量说明

d_

```
void* arcs::aubo_scope::ViewApiProvider::d_ { nullptr } [private]
```

在文件 [view_api_provider.h](#) 第 66 行定义.

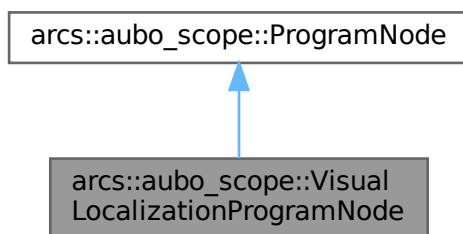
该类的文档由以下文件生成:

- [include/aubo_caps/contribution/view_api_provider.h](#)

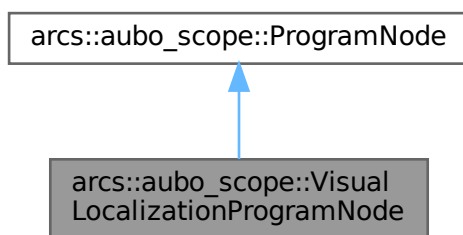
11.145 arcs::aubo_scope::VisualLocalizationProgramNode 类参考

```
#include <visual_localization_program_node.h>
```

类 arcs::aubo_scope::VisualLocalizationProgramNode 继承关系图:



arcs::aubo_scope::VisualLocalizationProgramNode 的协作图:



Public 成员函数

- [VisualLocalizationProgramNode](#) (VisualLocalizationProgramNode &f)
- [VisualLocalizationProgramNode](#) (VisualLocalizationProgramNode &&f)
- [virtual ~VisualLocalizationProgramNode](#) ()

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)

- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [VisualLocalizationProgramNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.145.1 详细描述

在文件 [visual_localization_program_node.h](#) 第 10 行定义.

11.145.2 构造及析构函数说明

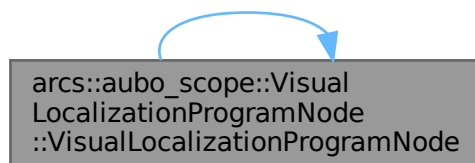
[VisualLocalizationProgramNode\(\)](#) [1/3]

```
arcs::aubo_scope::VisualLocalizationProgramNode::VisualLocalizationProgramNode (
    VisualLocalizationProgramNode & f)
```

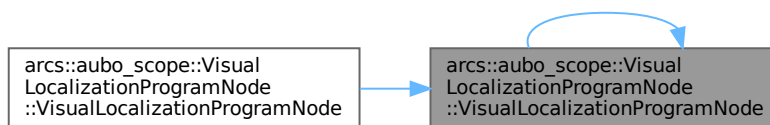
引用了 [VisualLocalizationProgramNode\(\)](#).

被这些函数引用 [VisualLocalizationProgramNode\(\)](#) , 以及 [VisualLocalizationProgramNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

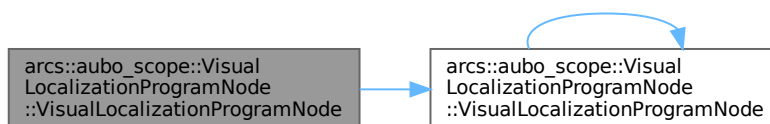


VisualLocalizationProgramNode() [2/3]

```
arcs::aubo_scope::VisualLocalizationProgramNode::VisualLocalizationProgramNode (
    VisualLocalizationProgramNode && f)
```

引用了 [VisualLocalizationProgramNode\(\)](#).

函数调用图:



~VisualLocalizationProgramNode()

```
virtual arcs::aubo_scope::VisualLocalizationProgramNode::~~VisualLocalizationProgramNode () [virtual]
```

VisualLocalizationProgramNode() [3/3]

```
arcs::aubo_scope::VisualLocalizationProgramNode::VisualLocalizationProgramNode () [private]
```

11.145.3 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [visual_localization_program_node.h](#) 第 18 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.145.4 类成员变量说明

d_

```
void* arcs::aubo_scope::VisualLocalizationProgramNode::d_ { nullptr } [private]
```

在文件 [visual_localization_program_node.h](#) 第 20 行定义.

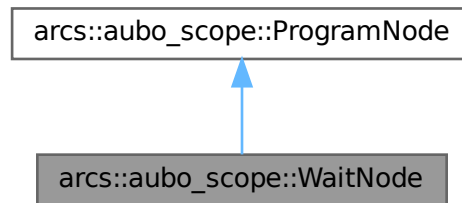
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/visual_localization_program_node.h](#)

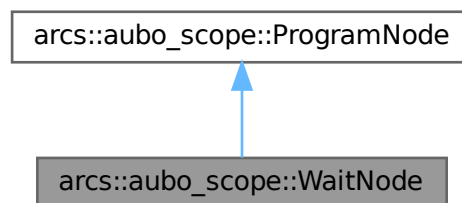
11.146 arcs::aubo_scope::WaitNode 类参考

```
#include <wait_node.h>
```

类 arcs::aubo_scope::WaitNode 继承关系图:



arcs::aubo_scope::WaitNode 的协作图:



Public 类型

- enum `ConfigType` : int {
`NO_TYPE`, `NO_WAIT`, `TIME`, `DIGITAL_INPUT`,
`ANALOG_INPUT_CURRENT`, `ANALOG_INPUT_VOLTAGE`, `FLOAT_REGISTER_INPUT`,
`EXPRESSION_INPUT` }

The configuration type used to determine which type of configuration this instance is.

- enum `CompareOperator` : int { `LESS_THAN`, `GREATER_THAN` }

Compare operator types LESS_THAN and GREATER_THAN used when waiting for an analog input to go lower or higher, respectively, than a threshold

Public 成员函数

- `WaitNode` (`WaitNode &f`)
- `WaitNode` (`WaitNode &&f`)
- virtual `~WaitNode` ()
- `ConfigType` `getConfigType` ()
This method returns the type of configuration.
- void `setConfigType` (`ConfigType` type)
- double `getWaitTime` ()
- void `setWaitTime` (double time)

- bool `getBoolValueToWaitFor` ()
Depending on the type of input this is interpreted in different ways.
- void `setBoolValueToWaitFor` (bool value)
- float `getFloatValueToWaitFor` ()
- void `setFloatValueToWaitFor` (float value)
- void `setDigitalInput` (DigitalIoPtr input)
Creates a configuration for waiting for a digital input to go high or low.
- DigitalIoPtr `getDigitalInput` ()
- void `setModbusInput` (ModbusIoPtr input)
Creates a configuration for waiting for a digital MODBUS input to go high or low.
- ModbusIoPtr `getModbusInput` ()
- void `setBoolRegister` (BoolRegisterPtr input)
Creates a configuration for waiting for a boolean register to go true or false.
- BoolRegisterPtr `getBoolRegister` ()
- void `setCompareOperator` (CompareOperator compareOperator)
- CompareOperator `getCompareOperator` ()
- void `setAnalogInput` (AnalogIoPtr input)
Creates a configuration for waiting for an analog input to go past an electric current threshold.
- AnalogIoPtr `getAnalogInput` ()
- void `setDoubleRegisterInput` (DoubleRegisterPtr input)
Creates a configuration for waiting for a register to go past a float threshold.
- DoubleRegisterPtr `getDoubleRegisterInput` ()
- void `setExpressionInputConfig` (ExpressionPtr expression)
Create a configuration for waiting for an expression to be evaluated to true.
- ExpressionPtr `getExpressionInputConfig` ()

Public 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` (ProgramNode &f)
- `ProgramNode` (ProgramNode &&f)
- virtual `~ProgramNode` ()
- ProgramNodeContributionPtr `getProgramNodeContribution` ()
获取程序节点贡献。不推荐用户直接调用 `ProgramNodeContribution::generateScript`, 应使用 `Script←Writer::writeChildren`.

Private 成员函数

- `WaitNode` ()

Private 属性

- void * `d_` { nullptr }

友元

- class `DataSwitch`

额外继承的成员函数

Protected 成员函数继承自 `arcs::aubo_scope::ProgramNode`

- `ProgramNode` ()

11.146.1 详细描述

在文件 `wait_node.h` 第 16 行定义.

11.146.2 成员枚举类型说明

CompareOperator

```
enum arcs::aubo_scope::WaitNode::CompareOperator : int
```

Compare operator types LESS_THAN and GREATER_THAN used when waiting for an analog input to go lower or higher, respectively, than a threshold

枚举值

LESS_THAN	
GREATER_THAN	

在文件 [wait_node.h](#) 第 101 行定义.

ConfigType

```
enum arcs::aubo_scope::WaitNode::ConfigType : int
```

The configuration type used to determine which type of configuration this instance is.

枚举值

NO_TYPE	
NO_WAIT	Nothing to wait.
TIME	Wait type is time. The config instance can be cast to <code>TimeWaitNodeConfig</code> .
DIGITAL_INPUT	Wait type is digital_input. The config instance can be cast to <code>DigitalInputWaitNodeConfig</code> .
ANALOG_INPUT_CURRENT	Wait type is analog_input_current. The config instance can be cast to <code>AnalogInputCurrentWaitNodeConfig</code> .
ANALOG_INPUT_VOLTAGE	Wait type is analog_input_voltage. The config instance can be cast to <code>AnalogInputVoltageWaitNodeConfig</code> .
FLOAT_REGISTER_INPUT	Wait type is float_register_input. The config instance can be cast to <code>FloatRegisterInputWaitNodeConfig</code> .
EXPRESSION_INPUT	Wait type is expression_input. The config instance can be cast to <code>ExpressionInputWaitNodeConfig</code> .

在文件 [wait_node.h](#) 第 23 行定义.

11.146.3 构造及析构造函数说明

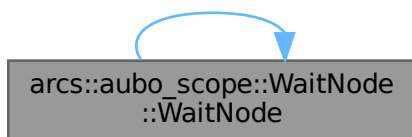
WaitNode() [1/3]

```
arcs::aubo_scope::WaitNode::WaitNode (
    WaitNode & f)
```

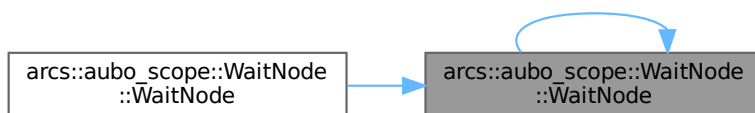
引用了 [WaitNode\(\)](#).

被这些函数引用 [WaitNode\(\)](#) , 以及 [WaitNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

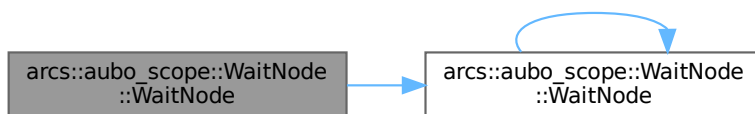


WaitNode() [2/3]

```
arcs::aubo_scope::WaitNode::WaitNode (  
    WaitNode && f)
```

引用了 [WaitNode\(\)](#).

函数调用图:



~WaitNode()

```
virtual arcs::aubo_scope::WaitNode::~~WaitNode () [virtual]
```

WaitNode() [3/3]

```
arcs::aubo_scope::WaitNode::WaitNode () [private]
```

11.146.4 成员函数说明

getAnalogInput()

```
AnalogIoPtr arcs::aubo_scope::WaitNode::getAnalogInput ()
```

getBoolRegister()

```
BoolRegisterPtr arcs::aubo_scope::WaitNode::getBoolRegister ()
```

getBoolValueToWaitFor()

```
bool arcs::aubo_scope::WaitNode::getBoolValueToWaitFor ()
```

Depending on the type of input this is interpreted in different ways.

For digital inputs and MODBUS inputs, **true** means the input to wait for must be HIGH, **false** means the input to wait for must be LOW. For boolean registers the value is interpreted literally.

返回

the input value to wait for.

getCompareOperator()

```
CompareOperator arcs::aubo_scope::WaitNode::getCompareOperator ()
```

getConfigType()

```
ConfigType arcs::aubo_scope::WaitNode::getConfigType ()
```

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

返回

the type of this config.

getDigitalInput()

```
DigitalIoPtr arcs::aubo_scope::WaitNode::getDigitalInput ()
```

getDoubleRegisterInput()

```
DoubleRegisterPtr arcs::aubo_scope::WaitNode::getDoubleRegisterInput ()
```

getExpressionInputConfig()

```
ExpressionPtr arcs::aubo_scope::WaitNode::getExpressionInputConfig ()
```

getFloatValueToWaitFor()

```
float arcs::aubo_scope::WaitNode::getFloatValueToWaitFor ()
```

getModbusInput()

```
ModbusIoPtr arcs::aubo_scope::WaitNode::getModbusInput ()
```

getWaitTime()

```
double arcs::aubo_scope::WaitNode::getWaitTime ()
```

返回

the time to wait for. second

setAnalogInput()

```
void arcs::aubo_scope::WaitNode::setAnalogInput (
    AnalogIoPtr input)
```

Creates a configuration for waiting for an analog input to go past an electric current threshold.

参数

<i>input</i>	on which to wait, not <code>null</code> .
<i>compareOperator</i>	the operator to use when comparing input to value, not <code>null</code> . Available options are "less than" (<) or "greater than" (>).
<i>current</i>	the electric current threshold to wait for, not <code>null</code> .
<i>errorHandler</i>	the error handler for validation errors. If using <code>ErrorHandler#AUTO_CORRECT</code> this will clamp the value to the nearest valid current value.

返回

the configuration.

异常

<i>IllegalArgumentException</i>	if <code>input</code> is an output (does not support reading of values). See <code>IO#isInput()</code> .

setBoolRegister()

```
void arcs::aubo_scope::WaitNode::setBoolRegister (
    BoolRegisterPtr input)
```

Creates a configuration for waiting for a boolean register to go `true` or `false`.

参数

<i>input</i>	the boolean register input on which to wait, not <code>null</code> .
<i>value_to_wait_for</i>	<code>true</code> , if waiting for the register to be <code>true</code> , <code>false</code> otherwise.

返回

the configuration.

异常

<i>IllegalArgumentException</i>	if <code>input</code> is an output (does not support reading of values). See <code>IO#isInput()</code> .

setBoolValueToWaitFor()

```
void arcs::aubo_scope::WaitNode::setBoolValueToWaitFor (
    bool value)
```

setCompareOperator()

```
void arcs::aubo_scope::WaitNode::setCompareOperator (
    CompareOperator compareOperator)
```


setConfigType()

```
void arcs::aubo_scope::WaitNode::setConfigType (  
    ConfigType type)
```

setDigitalInput()

```
void arcs::aubo_scope::WaitNode::setDigitalInput (  
    DigitalIoPtr input)
```

Creates a configuration for waiting for a digital input to go high or low.

参数

<i>input</i>	the digital input on which to wait, not <code>null</code> .
<i>value_to_wait_for</i>	<code>true</code> , if waiting for the input signal value/level to go HIGH, <code>false</code> otherwise (waiting for LOW signal value/level).

返回

the configuration.

异常

<i>IllegalArgumentException</i>	if <code>input</code> is an output (does not support reading of values). See <code>IO#isInput()</code> .

setDoubleRegisterInput()

```
void arcs::aubo_scope::WaitNode::setDoubleRegisterInput (  
    DoubleRegisterPtr input)
```

Creates a configuration for waiting for a register to go past a float threshold.

参数

<i>input</i>	the float register input on which to wait, not <code>null</code> .
<i>compareOperator</i>	the operator to use when comparing register to value, not <code>null</code> . Available options are "less than" (<) or "greater than" (>).
<i>value_to_wait_for</i>	the float threshold to wait for.

返回

the configuration.

异常

<i>IllegalArgumentException</i>	if <code>input</code> is an output (does not support reading of values). See <code>IO#isInput()</code> .

setExpressionInputConfig()

```
void arcs::aubo_scope::WaitNode::setExpressionInputConfig (
    ExpressionPtr expression)
```

Create a configuration for waiting for an expression to be evaluated to true.

参数

<i>expression</i>	the expression to evaluate, not <code>null</code> .

返回

the configuration.

setFloatValueToWaitFor()

```
void arcs::aubo_scope::WaitNode::setFloatValueToWaitFor (
    float value)
```

setModbusInput()

```
void arcs::aubo_scope::WaitNode::setModbusInput (
    ModbusIoPtr input)
```

Creates a configuration for waiting for a digital MODBUS input to go high or low.

参数

<i>input</i>	the digital MODBUS input on which to wait, not <code>null</code> .
<i>value_to_wait_for</i>	<code>true</code> , if waiting for the input signal value/level to go HIGH, <code>false</code> otherwise (waiting for LOW signal value/level).

返回

the configuration.

异常

<i>IllegalArgumentException</i>	if <code>input</code> is an output (does not support reading of values) or if <code>output</code> is not a digital output. See <code>I0#isInput()</code> and <code>I0#getType()</code> .

setWaitTime()

```
void arcs::aubo_scope::WaitNode::setWaitTime (
    double time)
```

11.146.5 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 `wait_node.h` 第 239 行定义。
引用了 `DataSwitch`。
被这些函数引用 `DataSwitch`。

11.146.6 类成员变量说明

d_

```
void* arcs::aubo_scope::WaitNode::d_ { nullptr } [private]
```

在文件 [wait_node.h](#) 第 241 行定义.

该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/wait_node.h](#)

11.147 arcs::aubo_scope::Waypoint 类参考

路点接口此接口组合了定义机器人位置的参数。

```
#include <waypoint.h>
```

Public 成员函数

- [Waypoint](#) (Waypoint &f)
- [Waypoint](#) (Waypoint &&f)
- [~Waypoint](#) ()
- [std::string getDisplayName](#) ()
获取路点显示名称
- [std::vector< double > getPose](#) ()
获取机器人位置的位姿 (包含 *TCP* 偏移)
- [std::vector< double > getJointPositions](#) ()
获取与机器人位姿对应的关节位置 (考虑 *TCP* 偏移)
- [std::vector< double > getTcpOffset](#) ()
获取 *TCP* 偏移位姿 (用于定义机器人位置)
- [bool isDefined](#) ()
检查路点是否已定义
- [size_t hashCode](#) ()
获取哈希码

Private 成员函数

- [Waypoint](#) ()

Private 属性

- `void * d_ { nullptr }`

友元

- `class DataSwitch`

11.147.1 详细描述

路点接口此接口组合了定义机器人位置的参数。

在文件 [waypoint.h](#) 第 23 行定义.

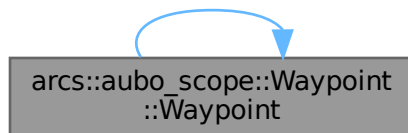
11.147.2 构造及析构造函数说明

Waypoint() [1/3]

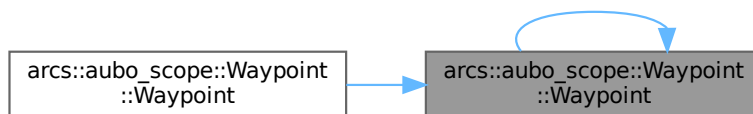
```
arcs::aubo_scope::Waypoint::Waypoint (  
    Waypoint & f)
```

引用了 [Waypoint\(\)](#).

被这些函数引用 [Waypoint\(\)](#) , 以及 [Waypoint\(\)](#).
函数调用图:



这是这个函数的调用关系图:

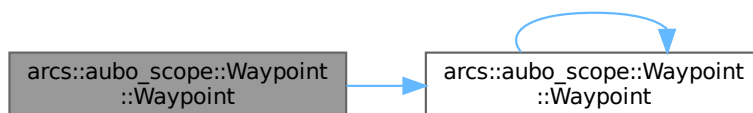


Waypoint() [2/3]

```
arcs::aubo_scope::Waypoint::Waypoint (  
    Waypoint && f)
```

引用了 [Waypoint\(\)](#).

函数调用图:



~Waypoint()

```
arcs::aubo_scope::Waypoint::~~Waypoint ()
```

Waypoint() [3/3]

```
arcs::aubo_scope::Waypoint::Waypoint () [private]
```

11.147.3 成员函数说明**getDisplayName()**

```
std::string arcs::aubo_scope::Waypoint::getDisplayName ()  
获取路点显示名称
```

返回

显示名称。

getJointPositions()

```
std::vector< double > arcs::aubo_scope::Waypoint::getJointPositions ()  
获取与机器人位姿对应的关节位置（考虑TCP 偏移）
```

返回

关节位置。

getPose()

```
std::vector< double > arcs::aubo_scope::Waypoint::getPose ()  
获取机器人位置的位姿（包含TCP 偏移）
```

返回

位姿。

getTcpOffset()

```
std::vector< double > arcs::aubo_scope::Waypoint::getTcpOffset ()  
获取TCP 偏移位姿（用于定义机器人位置）
```

返回

TCP 偏移。

hashCode()

```
size_t arcs::aubo_scope::Waypoint::hashCode ()  
获取哈希码
```

返回

哈希码。

isDefined()

```
bool arcs::aubo_scope::Waypoint::isDefined ()  
检查路点是否已定义
```

返回

是否已定义。

11.147.4 友元及相关符号说明

DataSwitch

friend class DataSwitch [friend]
 在文件 [waypoint.h](#) 第 111 行定义。
 引用了 [DataSwitch](#).
 被这些函数引用 [DataSwitch](#).

11.147.5 类成员变量说明

d_

void* arcs::aubo_scope::Waypoint::d_ { nullptr } [private]
 在文件 [waypoint.h](#) 第 113 行定义。
 该类的文档由以下文件生成:

- include/aubo_caps/domain/waypoint/[waypoint.h](#)

11.148 arcs::aubo_scope::WaypointModel 类参考

路点模型提供从当前机器人获取路点的方法。

#include <waypoint_model.h>

Public 成员函数

- [WaypointModel](#) (WaypointModel &f)
- [WaypointModel](#) (WaypointModel &&f)
- [~WaypointModel](#) ()
- std::vector< WaypointPtr > [getWaypoints](#) ()
 获取当前安装中的几何路点
- WaypointPtr [getByName](#) (const std::string &name)
 根据名称获取路点
- WaypointPtr [addWaypoint](#) (const std::string &suggestedName, const std::vector< double > &pose)
 向当前 *AuboScope* 安装中添加一个路点。添加后终端用户可选择该路点，但不可修改。
- WaypointPtr [addWaypoint](#) (const std::string &suggestedName, const std::vector< double > &pose, const std::vector< double > &joint, const std::vector< double > &offset)
 向当前 *AuboScope* 安装中添加一个包含关节角度和偏移的路点。
- void [removeWaypoint](#) (const std::string &name)
 从此 *AuboCap* 中移除已添加的路点。使用该路点的程序节点将变为未定义状态，因为路点不再可解析。
- void [renameWaypoint](#) (const std::string &name, const std::string &newName)
 根据名称重命名模型中的路点

Private 成员函数

- [WaypointModel](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

11.148.1 详细描述

路点模型提供从当前机器人获取路点的方法。

在文件 [waypoint_model.h](#) 第 22 行定义。

11.148.2 构造及析构造函数说明

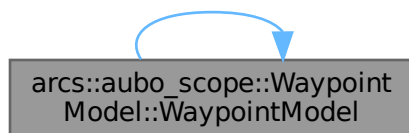
WaypointModel() [1/3]

```
arcs::aubo_scope::WaypointModel::WaypointModel (  
    WaypointModel & f)
```

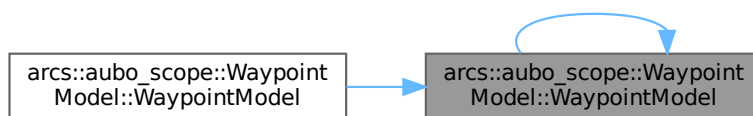
引用了 [WaypointModel\(\)](#).

被这些函数引用 [WaypointModel\(\)](#) , 以及 [WaypointModel\(\)](#).

函数调用图:



这是这个函数的调用关系图:

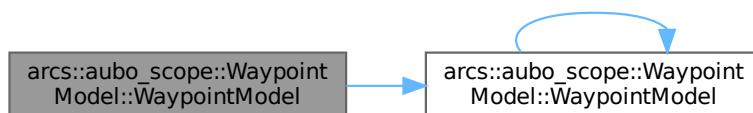


WaypointModel() [2/3]

```
arcs::aubo_scope::WaypointModel::WaypointModel (  
    WaypointModel && f)
```

引用了 [WaypointModel\(\)](#).

函数调用图:



~WaypointModel()

```
arcs::aubo_scope::WaypointModel::~~WaypointModel ()
```

WaypointModel() [3/3]

```
arcs::aubo_scope::WaypointModel::WaypointModel () [private]
```

11.148.3 成员函数说明

addWaypoint() [1/2]

```
WaypointPtr arcs::aubo_scope::WaypointModel::addWaypoint (
    const std::string & suggestedName,
    const std::vector< double > & pose)
```

向当前AuboScope 安装中添加一个路点。添加后终端用户可选择该路点，但不可修改。

参数

<i>suggestedName</i>	建议的路点名称。有效名称必须匹配正则表达式 <code>[a-zA-Z][a-zA-Z0-9_]{0,14}</code> ，总共 15 个字符。最终名称可从返回的Waypoint 实例获取。
<i>pose</i>	路点相对于机器人基座的位姿。

返回

在AuboScope 中创建并注册的路点。

异常

<i>WaypointAlreadyAddedException</i>	如果之前已用相同的 <code>idKey</code> 标识符添加过路点。使用 <code>getWaypoint(String)</code> 检查路点是否已添加到当前安装中。使用 <code>updateWaypoint(String, Pose)</code> 来更新路点。
<i>IllegalWaypointNameException</i>	如果建议的名称不匹配所需的正则表达式。

addWaypoint() [2/2]

```
WaypointPtr arcs::aubo_scope::WaypointModel::addWaypoint (
    const std::string & suggestedName,
    const std::vector< double > & pose,
    const std::vector< double > & joint,
    const std::vector< double > & offset)
```

向当前AuboScope 安装中添加一个包含关节角度和偏移的路点。

参数

<i>suggestedName</i>	建议的路点名称。有效名称必须匹配正则表达式 <code>[a-zA-Z][a-zA-Z0-9_]{0,14}</code> ，总共 15 个字符。最终名称可从返回的Waypoint 实例获取。
<i>pose</i>	路点相对于机器人基座的位姿。
<i>joint</i>	路点相对于机器人基座的关节角度。
<i>offset</i>	路点相对于机器人基座的TCP 偏移。

返回

在AuboScope 中创建并注册的路点。

异常

<i>WaypointAlreadyAddedException</i>	如果之前已用相同的 <code>idKey</code> 标识符添加过路点。
<i>IllegalWaypointNameException</i>	如果建议的名称不匹配所需的正则表达式。

getName()

```
WaypointPtr arcs::aubo_scope::WaypointModel::getName (  
    const std::string & name)  
根据名称获取路点
```

参数

<i>name</i>	路点名称。

返回

对应名称的路点。

getWaypoints()

```
std::vector< WaypointPtr > arcs::aubo_scope::WaypointModel::getWaypoints ()  
获取当前安装中的几何路点
```

返回

当前安装中的几何路点集合。

removeWaypoint()

```
void arcs::aubo_scope::WaypointModel::removeWaypoint (  
    const std::string & name)  
从此AuboCap 中移除已添加的路点。使用该路点的程序节点将变为未定义状态，因为路点不再可解析。
```

参数

<i>name</i>	添加路点时使用的名称。

异常

<i>WaypointNotFoundException</i>	如果没有与提供的 <code>idKey</code> 对应的路点存在。

renameWaypoint()

```
void arcs::aubo_scope::WaypointModel::renameWaypoint (  
    const std::string & name,  
    const std::string & newName)
```

根据名称重命名模型中的路点

参数

<i>name</i>	原始名称。
<i>newName</i>	新名称。

11.148.4 友元及相关符号说明

DataSwitch

```
friend class DataSwitch [friend]
```

在文件 [waypoint_model.h](#) 第 169 行定义.

引用了 [DataSwitch](#).

被这些函数引用 [DataSwitch](#).

11.148.5 类成员变量说明

d_

```
void* arcs::aubo_scope::WaypointModel::d_ { nullptr } [private]
```

在文件 [waypoint_model.h](#) 第 171 行定义.

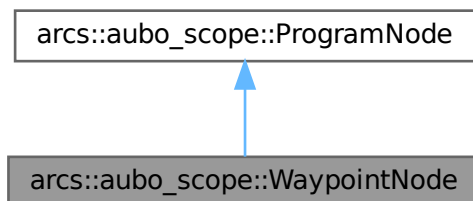
该类的文档由以下文件生成:

- [include/aubo_caps/domain/waypoint/waypoint_model.h](#)

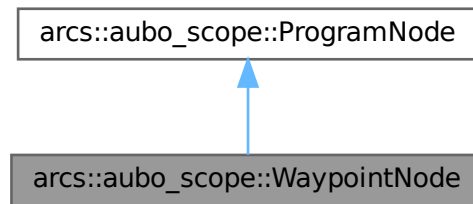
11.149 arcs::aubo_scope::WaypointNode 类参考

```
#include <waypoint_node.h>
```

类 arcs::aubo_scope::WaypointNode 继承关系图:



arcs::aubo_scope::WaypointNode 的协作图:



Public 类型

- enum `ConfigType` : int { `FIXED_POSITION` = 0 , `VARIABLE_POSITION` , `RELATIVE_POSITION` }
- The configuration type used to determine which type of configuration this instance is.*
- enum `BlendType` : int { `NO_BLEND` = 0 , `SHARED_BLEND` , `WAYPOINT_BLEND` }
- The blend parameters type used to determine which type of blend parameters this instance is.*
- enum `MotionType` : int { `TIME_MOTION` , `JOINT_MOTION` , `TOOL_MOTION` , `SHARED_MOTION` }
- The blend parameters type used to determine which type of blend parameters this instance is.*
- enum `VariableType` : int { `VARIABLE_POSE` , `VARIABLE_JOINT` }
- enum `RelativePositionType` : int { `TEACHING` , `INPUT` }
- enum `RelativeOffsetType` : int { `CUSTOM` , `VARIABLE` }
- enum class `PositionDefinition` : int { `UNDEFINED` , `DEFINED` }
- enum `VariableDefinition` : int { `UNDEFINED` , `DEFINED` }

Public 成员函数

- `WaypointNode` (`WaypointNode &f`)
- `WaypointNode` (`WaypointNode &&f`)
- virtual `~WaypointNode` ()
- `ConfigType` `getConfigType` ()
- This method returns the type of configuration.*
- `BlendType` `getBlendType` ()
- Get blend type of the waypoint*
- `MotionType` `getMotionType` ()
- `PositionDefinition` `getPositionDefinition` ()
- Cast this instance appropriately to have access to specific getters.*
- `VariableDefinition` `getVariableDefinition` ()
- Cast this instance appropriately to have access to specific getters.*
- double `getBlend` ()
- double `getJointSpeed` ()
- double `getJointAcceleration` ()
- double `getTime` ()
- double `getToolSpeed` ()
- double `getToolAcceleration` ()
- std::vector< double > `getPose` ()
- std::vector< double > `getJointPositions` ()
- std::vector< double > `getTcpOffset` ()

- VariablePtr [getVariable](#) ()
- void [setSharedMotionParameters](#) ()
- void [setTimeMotionParameters](#) (unsigned int duration)
- void [setJointMotionParameters](#) (double joint_speed, double joint_acceleration)
- void [setToolMotionParameters](#) (double tool_speed, double tool_acceleration)
- void [setFixedPositionConfig](#) ()
- void [setFixedPositionConfig](#) (const std::vector< double > &pose_including_tcp, const std::vector< double > &qNear)
- void [setVariablePositionConfig](#) ()
- void [setVariablePositionConfig](#) (VariablePtr variable)

Creates a variable position configuration for a waypoint with a defined variable

Public 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
获取程序节点贡献。不推荐用户直接调用 *ProgramNodeContribution::generateScript*, 应使用 *ScriptWriter::writeChildren*。

Private 成员函数

- [WaypointNode](#) ()

Private 属性

- void * [d_](#) { nullptr }

友元

- class [DataSwitch](#)

额外继承的成员函数

Protected 成员函数继承自 [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.149.1 详细描述

在文件 [waypoint_node.h](#) 第 13 行定义.

11.149.2 成员枚举类型说明

BlendType

```
enum arcs::aubo\_scope::WaypointNode::BlendType : int
```

The blend parameters type used to determine which type of blend parameters this instance is.

枚举值

NO_BLEND	No blend has been selected (the robot arm will stop at this point).
SHARED_BLEND	Shared blend has been selected.
WAYPOINT_BLEND	> Waypoint blend has been selected. The blend instance can be cast to WaypointBlendParameters .

在文件 [waypoint_node.h](#) 第 52 行定义.

ConfigType

```
enum arcs::aubo_scope::WaypointNode::ConfigType : int
```

The configuration type used to determine which type of configuration this instance is.

枚举值

FIXED_POSITION	Fixed position has been selected (fixed waypoint). The config instance can be cast to <code>FixedPositionWaypointNodeConfig</code> .
VARIABLE_POSITION	Variable position has been selected (variable waypoint). The config instance can be cast to <code>VariablePositionWaypointNodeConfig</code> .
RELATIVE_POSITION	Relative position has been selected (relative waypoint).

在文件 [waypoint_node.h](#) 第 20 行定义.

MotionType

```
enum arcs::aubo_scope::WaypointNode::MotionType : int
```

The blend parameters type used to determine which type of blend parameters this instance is.

枚举值

TIME_MOTION	Time motion parameters has been selected. This instance can be cast to <code>TimeMotionParameters</code> .
JOINT_MOTION	Joint motion parameters has been selected. This instance can be cast to <code>JointMotionParameters</code> .
TOOL_MOTION	Tool motion parameters has been selected. This instance can be cast to <code>ToolMotionParameters</code> .
SHARED_MOTION	Shared motion parameters has been selected.

在文件 [waypoint_node.h](#) 第 78 行定义.

PositionDefinition

```
enum class arcs::aubo_scope::WaypointNode::PositionDefinition : int [strong]
```

枚举值

UNDEFINED	The position has not been defined.
DEFINED	The position of the waypoint has been defined. Cast this instance to <code>FixedPositionDefinedWaypointNodeConfig</code> .

在文件 [waypoint_node.h](#) 第 152 行定义.

RelativeOffsetType

```
enum arcs::aubo_scope::WaypointNode::RelativeOffsetType : int
```

枚举值

CUSTOM	Custom offset has been selected.
VARIABLE	Variable type has been selected.

在文件 [waypoint_node.h](#) 第 139 行定义.

RelativePositionType

```
enum arcs::aubo_scope::WaypointNode::RelativePositionType : int
```

枚举值

TEACHING	The teaching location has been selected.
INPUT	The input offset has been selected.

在文件 [waypoint_node.h](#) 第 126 行定义.

VariableDefinition

```
enum arcs::aubo_scope::WaypointNode::VariableDefinition : int
```

枚举值

UNDEFINED	A variable has not been selected. This instance has no further information.
DEFINED	A variable has been selected making this config defined. Cast this instance to VariablePositionDefinedWaypointNodeConfig .

在文件 [waypoint_node.h](#) 第 169 行定义.

VariableType

```
enum arcs::aubo_scope::WaypointNode::VariableType : int
```

枚举值

VARIABLE_POSE	This variable type pose parameters has been selected.
VARIABLE_JOINT	This variable type joint parameters has been selected.

在文件 [waypoint_node.h](#) 第 113 行定义.

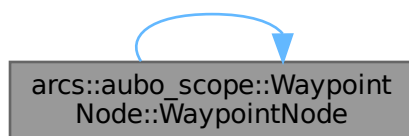
11.149.3 构造及析构函数说明**WaypointNode()** [1/3]

```
arcs::aubo_scope::WaypointNode::WaypointNode (  
    WaypointNode & f)
```

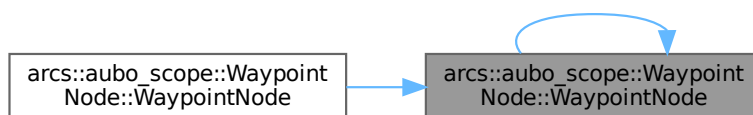
引用了 [WaypointNode\(\)](#).

被这些函数引用 [WaypointNode\(\)](#) , 以及 [WaypointNode\(\)](#).

函数调用图:



这是这个函数的调用关系图:

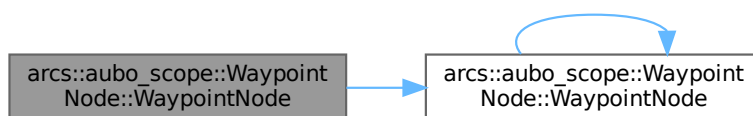


WaypointNode() [2/3]

```
arcs::aubo_scope::WaypointNode::WaypointNode (  
    WaypointNode && f)
```

引用了 [WaypointNode\(\)](#).

函数调用图:



~WaypointNode()

```
virtual arcs::aubo_scope::WaypointNode::~~WaypointNode () [virtual]
```

WaypointNode() [3/3]

```
arcs::aubo_scope::WaypointNode::WaypointNode () [private]
```

11.149.4 成员函数说明

getBlend()

```
double arcs::aubo_scope::WaypointNode::getBlend ()
```

Get the blend to be used for the movement.

Note that, if the configuration has not been applied to a waypoint node in the program tree, the blend value returned is a "suggested" value, since the blend may be (automatically) corrected depending on surrounding waypoints in the program tree (when it is applied to a waypoint node).

返回

the blend for the waypoint.

getBlendType()

```
BlendType arcs::aubo_scope::WaypointNode::getBlendType ()
```

Get blend type of the waypoint

返回

the blend type.

getConfigType()

```
ConfigType arcs::aubo_scope::WaypointNode::getConfigType ()
```

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

返回

the type of this config.

getJointAcceleration()

```
double arcs::aubo_scope::WaypointNode::getJointAcceleration ()
```

返回

the joint acceleration to use for the movement.

getJointPositions()

```
std::vector< double > arcs::aubo_scope::WaypointNode::getJointPositions ()
```

返回

the joint positions corresponding to the pose of the robot (when taking the TCP offset into account)

getJointSpeed()

```
double arcs::aubo_scope::WaypointNode::getJointSpeed ()
```

返回

the joint speed to be achieved.

getMotionType()

```
MotionType arcs::aubo_scope::WaypointNode::getMotionType ()
```

getPose()

```
std::vector< double > arcs::aubo_scope::WaypointNode::getPose ()
```

返回

the waypoint position and orientation.

getPositionDefinition()

`PositionDefinition` arcs::aubo_scope::WaypointNode::getPositionDefinition ()

Cast this instance appropriately to have access to specific getters.

返回

the position definition.

getTcpOffset()

`std::vector< double >` arcs::aubo_scope::WaypointNode::getTcpOffset ()

返回

the pose for the `TCP` offset (used when defining the robot position)

getTime()

`double` arcs::aubo_scope::WaypointNode::getTime ()

返回

the time for the waypoint movement.

getToolAcceleration()

`double` arcs::aubo_scope::WaypointNode::getToolAcceleration ()

返回

the tool acceleration to use for the movement

getToolSpeed()

`double` arcs::aubo_scope::WaypointNode::getToolSpeed ()

返回

the tool speed to be achieved.

getVariable()

`VariablePtr` arcs::aubo_scope::WaypointNode::getVariable ()

返回

the variable for the waypoint.

getVariableDefinition()

`VariableDefinition` arcs::aubo_scope::WaypointNode::getVariableDefinition ()

Cast this instance appropriately to have access to specific getters.

返回

the variable definition.

setFixedPositionConfig() [1/2]

```
void arcs::aubo_scope::WaypointNode::setFixedPositionConfig ()
```

Creates a fixed position configuration with default values for a waypoint.
Note that these default values may change between AuboScope versions.

返回

the waypoint configuration.

setFixedPositionConfig() [2/2]

```
void arcs::aubo_scope::WaypointNode::setFixedPositionConfig (
    const std::vector< double > & pose_including_tcp,
    const std::vector< double > & qNear)
```

Creates a fixed position configuration for a waypoint with a defined pose.

参数

<i>pose_including_tcp</i>	the position and orientation for the waypoint, not null .
<i>qNear</i>	the joint positions that will be used for the inverse kinematics calculations as a robot configuration near the waypoint pose, not null .
<i>blendParameters</i>	the blend parameters for the waypoint, not null .
<i>waypoint_motion_parameters</i>	the motion parameters for the waypoint, not null .

返回

the waypoint configuration.

setJointMotionParameters()

```
void arcs::aubo_scope::WaypointNode::setJointMotionParameters (
    double joint_speed,
    double joint_acceleration)
```

Creates joint motion parameters for a waypoint.
Can be used for waypoints under a MoveJ.

参数

<i>jointSpeed</i>	the joint speed to be achieved, not null .
<i>jointSpeedErrorHandler</i>	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid joint speed value.
<i>joint_acceleration</i>	the joint acceleration to be used for the movement, not null .
<i>joint_accelerationErrorHandler</i>	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid joint acceleration value.

返回

the joint motion parameters for the waypoint.

setSharedMotionParameters()

```
void arcs::aubo_scope::WaypointNode::setSharedMotionParameters ()
```

Creates shared motion parameters for a waypoint.

Can be used for waypoints under a MoveJ, MoveL or a MoveP.

返回

the motion parameters for the waypoint.

setTimeMotionParameters()

```
void arcs::aubo_scope::WaypointNode::setTimeMotionParameters (
```

```
    unsigned int duration)
```

Creates time motion parameters for a waypoint.

Can be used for waypoints under a MoveJ or a MoveL.

参数

<i>duration</i>	the time the movement should take, not null .
<i>errorHandler</i>	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid time value.

返回

the time motion parameters for the waypoint.

setToolMotionParameters()

```
void arcs::aubo_scope::WaypointNode::setToolMotionParameters (
```

```
    double tool_speed,
```

```
    double tool_acceleration)
```

Creates tool motion parameters for a waypoint.

Can be used for waypoints under a MoveL or a MoveP.

参数

<i>tool_speed</i>	the tool speed to be achieved, not null .
<i>tool_speedErrorHandler</i>	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid tool speed value.
<i>tool_acceleration</i>	the tool acceleration to be used for the movement, not null .
<i>tool_accelerationErrorHandler</i>	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid tool acceleration value.

返回

the tool motion parameters for the waypoint.

setVariablePositionConfig() [1/2]

```
void arcs::aubo_scope::WaypointNode::setVariablePositionConfig ()
```

Creates a variable position configuration with default values for a waypoint
Note that these default values may change between AuboScope versions.

返回

the waypoint configuration.

setVariablePositionConfig() [2/2]

```
void arcs::aubo_scope::WaypointNode::setVariablePositionConfig (
    VariablePtr variable)
```

Creates a variable position configuration for a waypoint with a defined variable

参数

<i>variable</i>	the variable position and orientation for the waypoint, not <code>null</code> .
<i>blendParameters</i>	the blend parameters for the waypoint, not <code>null</code> .
<i>waypoint_motion_parameters</i>	the motion parameters for the waypoint, not <code>null</code> .

返回

the waypoint configuration.

11.149.5 友元及相关符号说明**DataSwitch**

```
friend class DataSwitch [friend]
```

在文件 [waypoint_node.h](#) 第 409 行定义.
引用了 [DataSwitch](#).
被这些函数引用 [DataSwitch](#).

11.149.6 类成员变量说明

`d_`

```
void* arcs::aubo_scope::WaypointNode::d_ { nullptr } [private]
```

在文件 [waypoint_node.h](#) 第 411 行定义.
该类的文档由以下文件生成:

- [include/aubo_caps/domain/program/nodes/builtin/waypoint_node.h](#)

11.150 arcs::aubo_scope::WidthCapability 类参考

宽度能力此接口表示夹爪已注册的宽度/位置能力，支持移动到用户可配置的位置（打开/关闭到可配置的宽度）。

```
#include <width_capability.h>
```

Public 成员函数

- virtual [~WidthCapability](#) ()=default
- virtual void [updateCapability](#) (double minWidth, double maxWidth, double defaultGripWidth, double defaultReleaseWidth)=0

11.150.1 详细描述

宽度能力此接口表示夹爪已注册的宽度/位置能力，支持移动到用户可配置的位置（打开/关闭到可配置的宽度）。

通过此接口，可以在能力注册后调整宽度的范围和默认值。这通常不适用于大多数夹爪，但当范围取决于另一个自定义夹爪设置（可由用户配置）时，这可能是相关的。

在文件 [width_capability.h](#) 第 31 行定义。

11.150.2 构造及析构函数说明

~WidthCapability()

```
virtual arcs::aubo_scope::WidthCapability::~WidthCapability () [virtual], [default]
```

11.150.3 成员函数说明

updateCapability()

```
virtual void arcs::aubo_scope::WidthCapability::updateCapability (
    double minWidth,
    double maxWidth,
    double defaultGripWidth,
    double defaultReleaseWidth) [pure virtual]
```

Update the range and default values for the width supported by the gripper.

Updating the range could result in an already entered value in the configuration of a gripper program node being outside the range. This will make the given program node undefined, and force the end user to adjust the value to be valid again.

参数

<i>minWidth</i>	minimum width supported by the gripper
<i>maxWidth</i>	maximum width supported by the gripper
<i>defaultGripWidth</i>	default value for the width for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
<i>defaultReleaseWidth</i>	default value for the width for a release action. This value is used for, e.g. performing a "default" release action using the toolbar
<i>unit</i>	the unit for all specified values, not null

异常

<i>InvalidCapabilityRange</i>	if <code>minWidth</code> > <code>maxWidth</code>
<i>InvalidCapabilityRange</i>	if <code>defaultGripWidth</code> or <code>defaultReleaseWidth</code> is outside the range defined by <code>minWidth</code> and <code>maxWidth</code> .

该类的文档由以下文件生成:

- include/aubo_caps/contribution/driver/gripper/capability/[width_capability.h](#)

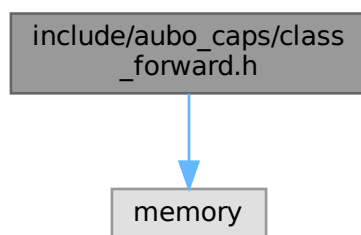
Chapter 12

文件说明

12.1 doc/SDK_overview.md 文件参考

12.2 include/aubo_caps/class_forward.h 文件参考

`#include <memory>`
class_forward.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



宏定义

- `#define ARCS_ABI_EXPORT`
- `#define ARCS_ABI_IMPORT`
- `#define ARCS_ABI_LOCAL`
- `#define ARCS_DECLARE_PTR(Name, Type)`
- `#define ARCS_DECLARE_PTR_MEMBER(Type)`
- `#define ARCS_CLASS_FORWARD(C)`
Macro that forward declares a class and defines the respective smartpointers through ARCS_DECLARE_PTR.
- `#define ARCS_STRUCT_FORWARD(C)`
Like ARCS_CLASS_FORWARD, but forward declares the type as a struct instead of a class.

12.2.1 宏定义说明

ARCS_ABI_EXPORT

`#define ARCS_ABI_EXPORT`
在文件 `class_forward.h` 第 16 行定义.

ARCS_ABI_IMPORT

`#define ARCS_ABI_IMPORT`
在文件 `class_forward.h` 第 17 行定义.

ARCS_ABI_LOCAL

`#define ARCS_ABI_LOCAL`
在文件 `class_forward.h` 第 18 行定义.

ARCS_CLASS_FORWARD

`#define ARCS_CLASS_FORWARD(
 C)`

值:

```
class C; \
ARCS_DECLARE_PTR(C, C)
```

Macro that forward declares a class and defines the respective smartpointers through `ARCS_DECLARE_PTR`.

在文件 `class_forward.h` 第 68 行定义.

ARCS_DECLARE_PTR

`#define ARCS_DECLARE_PTR(
 Name,
 Type)`

值:

```
typedef std::shared_ptr<Type> Name##Ptr; \
typedef std::shared_ptr<const Type> Name##ConstPtr; \
typedef std::weak_ptr<Type> Name##WeakPtr; \
typedef std::weak_ptr<const Type> Name##ConstWeakPtr; \
typedef std::unique_ptr<Type> Name##UniquePtr; \
typedef std::unique_ptr<const Type> Name##ConstUniquePtr
```

在文件 `class_forward.h` 第 36 行定义.

ARCS_DECLARE_PTR_MEMBER

`#define ARCS_DECLARE_PTR_MEMBER(
 Type)`

值:

```
typedef std::shared_ptr<Type> Ptr; \
typedef std::shared_ptr<const Type> ConstPtr; \
typedef std::weak_ptr<Type> WeakPtr; \
typedef std::weak_ptr<const Type> ConstWeakPtr; \
typedef std::unique_ptr<Type> UniquePtr; \
typedef std::unique_ptr<const Type> ConstUniquePtr
```

在文件 `class_forward.h` 第 54 行定义.

ARCS_STRUCT_FORWARD

`#define ARCS_STRUCT_FORWARD(
 C)`

值:

```
struct C; \
ARCS_DECLARE_PTR(C, C)
```

Like `ARCS_CLASS_FORWARD`, but forward declares the type as a struct instead of a class.

在文件 `class_forward.h` 第 77 行定义.

12.3 class_forward.h

浏览该文件的文档.

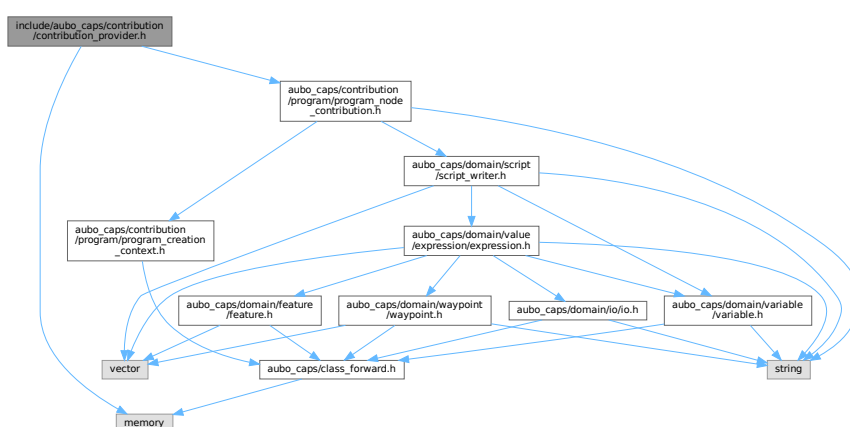
```

00001 #ifndef AUBO_SCOPE_CLASS_FORWARD_H
00002 #define AUBO_SCOPE_CLASS_FORWARD_H
00003
00004 #include <memory>
00005
00006 #if defined(ARCS_ABI_EXPORT)
00007 #elif defined(ARCS_PLATFORM_WINDOWS)
00008 #define ARCS_ABI_EXPORT __declspec(dllexport)
00009 #define ARCS_ABI_IMPORT __declspec(dllimport)
00010 #define ARCS_ABI_LOCAL
00011 #elif defined(ARCS_HAVE_VISIBILITY_ATTRIBUTE)
00012 #define ARCS_ABI_EXPORT __attribute__((visibility("default")))
00013 #define ARCS_ABI_IMPORT __attribute__((visibility("default")))
00014 #define ARCS_ABI_LOCAL __attribute__((visibility("hidden")))
00015 #else
00016 #define ARCS_ABI_EXPORT
00017 #define ARCS_ABI_IMPORT
00018 #define ARCS_ABI_LOCAL
00019 #endif
00020
00021 /**
00022  * @def ARCS_DECLARE_PTR
00023  * Macro that given a Name and a Type declares the following types:
00024  * - ${Name}Ptr = shared_ptr<${Type}>
00025  * - ${Name}ConstPtr = shared_ptr<const ${Type}>
00026  * - ${Name}WeakPtr = weak_ptr<${Type}>
00027  * - ${Name}ConstWeakPtr = weak_ptr<const ${Type}>
00028  * - ${Name}UniquePtr = unique_ptr<${Type}>
00029  * - ${Name}ConstUniquePtr = unique_ptr<const ${Type}>
00030  *
00031  * For best portability the exact type of shared_ptr declared by the macro
00032  * should be considered to be an implementation detail, liable to change in
00033  * future releases.
00034  */
00035
00036 #define ARCS_DECLARE_PTR(Name, Type) \
00037     typedef std::shared_ptr<Type> Name##Ptr; \
00038     typedef std::shared_ptr<const Type> Name##ConstPtr; \
00039     typedef std::weak_ptr<Type> Name##WeakPtr; \
00040     typedef std::weak_ptr<const Type> Name##ConstWeakPtr; \
00041     typedef std::unique_ptr<Type> Name##UniquePtr; \
00042     typedef std::unique_ptr<const Type> Name##ConstUniquePtr
00043
00044 /**
00045  * @def ARCS_DECLARE_PTR_MEMBER
00046  * The macro defines the same typedefs as ARCS_DECLARE_PTR, but shortens the
00047  * new names to their suffix.
00048  *
00049  * This can be used to create `Classname::Ptr` style names, but in most
00050  * situations in ARCS's codebase, ARCS_CLASS_FORWARD and ARCS_DECLARE_PTR
00051  * should be preferred.
00052  */
00053
00054 #define ARCS_DECLARE_PTR_MEMBER(Type) \
00055     typedef std::shared_ptr<Type> Ptr; \
00056     typedef std::shared_ptr<const Type> ConstPtr; \
00057     typedef std::weak_ptr<Type> WeakPtr; \
00058     typedef std::weak_ptr<const Type> ConstWeakPtr; \
00059     typedef std::unique_ptr<Type> UniquePtr; \
00060     typedef std::unique_ptr<const Type> ConstUniquePtr
00061
00062 /**
00063  * @def ARCS_CLASS_FORWARD
00064  * Macro that forward declares a class and defines the respective smartpointers
00065  * through ARCS_DECLARE_PTR.
00066  */
00067
00068 #define ARCS_CLASS_FORWARD(C) \
00069     class C; \
00070     ARCS_DECLARE_PTR(C, C)
00071
00072 /**
00073  * @def ARCS_STRUCT_FORWARD
00074  * Like ARCS_CLASS_FORWARD, but forward declares the type as a struct
00075  * instead of a class.
00076  */
00077 #define ARCS_STRUCT_FORWARD(C) \
00078     struct C; \
00079     ARCS_DECLARE_PTR(C, C)
00080
00081 #endif

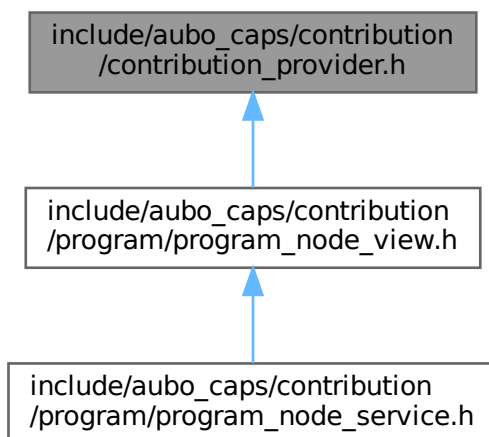
```

12.4 include/aubo_caps/contribution/contribution_provider.h 文件参考

```
#include <memory>
#include <aubo_caps/contribution/program/program_node_contribution.h>
contribution_provider.h 的引用 (Include) 关系图:
```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::ContributionProvider< T >`
贡献提供者此接口提供对贡献的访问。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

类型定义

- `template<class T>`
using `arcs::aubo_scope::ContributionProviderPtr` = `std::shared_ptr<ContributionProvider<T>>`

12.5 contribution_provider.h

[浏览该文件的文档.](#)

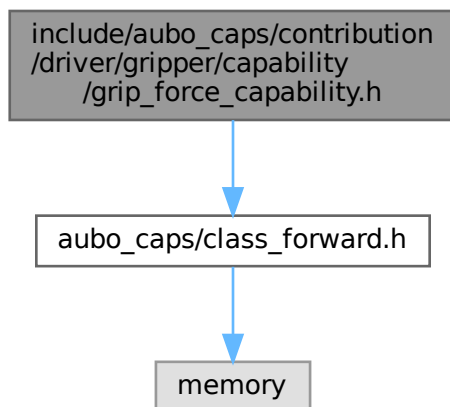
```

00001 #ifndef AUBO_SCOPE_CONTRIBUTION_PROVIDER_H
00002 #define AUBO_SCOPE_CONTRIBUTION_PROVIDER_H
00003
00004 #include <memory>
00005 #include <aubo_caps/contribution/program/program_node_contribution.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009
00010 /**
00011  * \chinese
00012  * 贡献提供者
00013  * 此接口提供对贡献的访问。
00014  *
00015  * 注意：此接口不适用于具有基于 HTML 用户界面的贡献。
00016  *
00017  * @tparam <T> 表示接口提供访问的贡献类型的（通用）类型参数
00018  * \endchinese
00019  * \english
00020  * ContributionProvider
00021  * This interfaces provides access to a contribution.
00022  *
00023  * NOTE: This interface is not relevant for contributions with a HTML-based user
00024  * interface.
00025  *
00026  * @tparam <T> the (generic) type parameter for the interface representing the
00027  * type of contribution the interface provides access to
00028  * \endenglish
00029  */
00030 template <class T>
00031 class ContributionProvider
00032 {
00033 public:
00034     virtual ~ContributionProvider() = default;
00035
00036     /**
00037      * \chinese
00038      * 获取一个贡献。在用户操作的结果中调用此方法以获取当前选中的节点。
00039      *
00040      * @return 相关的贡献
00041      * \endchinese
00042      * \english
00043      * Gets a contribution. Call this as the result of a user action to get the
00044      * currently selected node.
00045      *
00046      * @return the relevant contribution
00047      * \endenglish
00048      */
00049     virtual T *get() = 0;
00050 };
00051
00052 template <class T>
00053 using ContributionProviderPtr = std::shared_ptr<ContributionProvider<T>>;
00054
00055 } // namespace aubo_scope
00056 } // namespace arcs
00057
00058 #endif // AUBO_SCOPE_CONTRIBUTION_PROVIDER_H

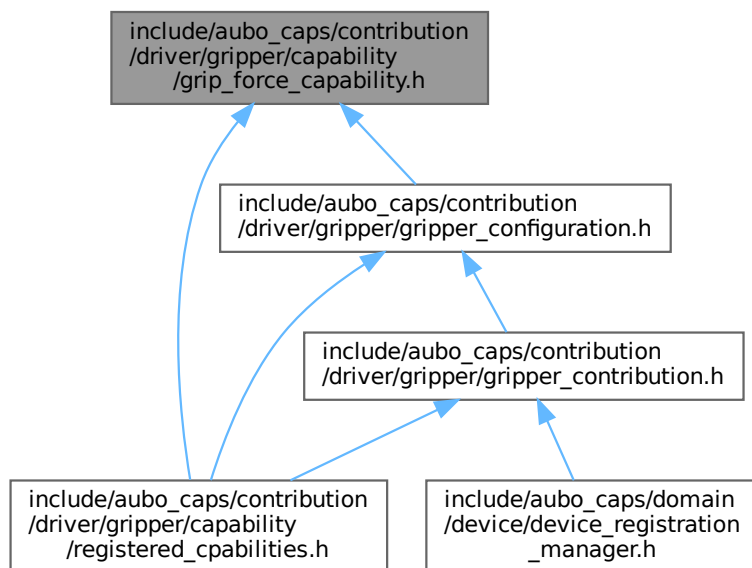
```

12.6 include/aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h 文件参考

#include <aubo_caps/class_forward.h>
grip_force_capability.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::GripForceCapability](#)
夹持力能力此接口表示夹爪已注册的力度能力，支持使用用户可配置的力度进行夹取。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`GripForceCapability`)

12.7 grip_force_capability.h

[浏览该文件的文档.](#)

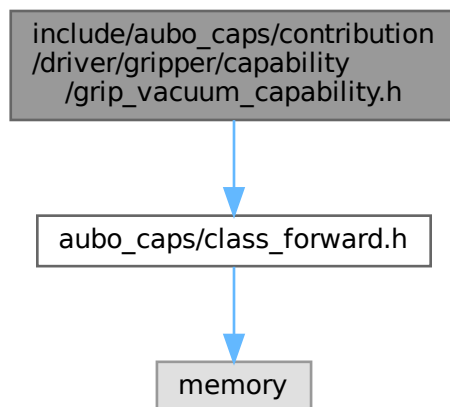
```

00001 #ifndef AUBO_SCOPE_GRIP_FORCE_CAPABILITY_H
00002 #define AUBO_SCOPE_GRIP_FORCE_CAPABILITY_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(GripForceCapability);
00009
00010         /**
00011          * \chinese
00012          * 夹持力能力
00013          * 此接口表示夹爪已注册的力度能力，支持使用用户可配置的力度进行夹取。
00014          *
00015          * 通过此接口，可以在能力注册后调整力度的范围和默认值。这通常不适用于大多数夹爪，
00016          * 但当范围取决于另一个自定义夹爪设置（可由用户配置）时，这可能是相关的。
00017          * \endchinese
00018          * \english
00019          * GripForceCapability
00020          * This interface represents a registered force capability for a gripper which
00021          * supports gripping using a user configurable force.
00022          *
00023          * Through this interface, it is possible to adjust the range and default value
00024          * for the force after the capability has been registered. This is typically not
00025          * applicable for most grippers, but can be relevant when the range depends on
00026          * another custom gripper setting (which can be configured by the user).
00027          * \endenglish
00028          */
00029         class GripForceCapability
00030         {
00031         public:
00032             virtual ~GripForceCapability() = 0;
00033
00034             /**
00035              * <p>
00036              * Update the range and default value for the force supported by the
00037              * gripper.
00038              * </p>
00039              *
00040              * Updating the range could result in an already entered value in the
00041              * configuration of a gripper program node being outside the range. This
00042              * will make the given program node undefined, and force the end user to
00043              * adjust the value to be valid again.
00044              *
00045              * @param minForce minimum force supported by the gripper
00046              * @param maxForce maximum force supported by the gripper
00047              * @param defaultGripForce default value for the force for a grip action.
00048              * This value is used for, e.g. performing a "default" grip action using the
00049              * toolbar
00050              * @param unit the unit for all specified values, <code>null</code>
00051              * @throws InvalidCapabilityRange if <code>minForce</code> > <code>maxForce</code>
00052              * or <code>defaultGripForce</code> is outside the range defined by
00053              * <code>minForce</code> and <code>maxForce</code>
00054              */
00055             virtual void updateCapability(double minForce, double maxForce,
00056                                         double defaultGripForce) = 0;
00057         };
00058     } // namespace aubo_scope
00059 } // namespace arcs
00060 #endif

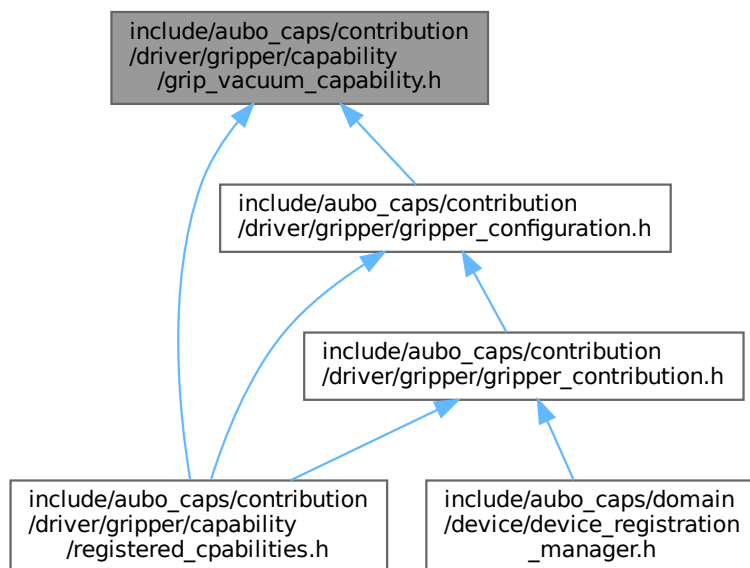
```

12.8 include/aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h 文件参考

#include <aubo_caps/class_forward.h>
grip_vacuum_capability.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::GripVacuumCapability`

真空夹持能力此接口表示夹爪已注册的真空能力, 支持使用用户可配置的真空度进行夹取。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (GripVacuumCapability)`

12.9 grip_vacuum_capability.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_GRIP_VACUUM_CAPABILITY_H
00002 #define AUBO_SCOPE_GRIP_VACUUM_CAPABILITY_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(GripVacuumCapability);
00009
00010         /**
00011          * \chinese
00012          * 真空夹持能力
00013          * 此接口表示夹爪已注册的真空能力，支持使用用户可配置的真空度进行夹取。
00014          *
00015          * 通过此接口，可以在能力注册后调整真空度的范围和默认值。这通常不适用于大多数夹爪，
00016          * 但当范围取决于另一个自定义夹爪设置（可由用户配置）时，这可能是相关的。
00017          * \endchinese
00018          * \english
00019          * GripVacuumCapability
00020          * This interface represents a registered vacuum capability for a gripper which
00021          * supports gripping using a user configurable vacuum level.
00022          *
00023          * Through this interface, it is possible to adjust the range and default value
00024          * for the vacuum level after the capability has been registered. This is
00025          * typically not applicable for most grippers, but can be relevant when the
00026          * range depends on another custom gripper setting (which can be configured by
00027          * the user).
00028          * \endenglish
00029          */
00030         class GripVacuumCapability
00031         {
00032         public:
00033             virtual ~GripVacuumCapability() = 0;
00034
00035             /**
00036              * <p>
00037              * Update the range and default value for the vacuum level supported by the
00038              * gripper.
00039              * </p>
00040              *
00041              * <p>
00042              * Updating the range could result in an already entered value in the
00043              * configuration of a gripper program node being outside the range. This
00044              * will make the given program node undefined, and force the end user to
00045              * adjust the value to be valid again.
00046              * </p>
00047              *
00048              * If the gripper works with absolute pressure, specify a positive range for
00049              * the supported vacuum level. If the gripper works with relative pressure
00050              * (vacuum), specify a negative range for the supported vacuum level.
00051              *
00052              * @param minVacuum minimum vacuum level supported by the gripper
00053              * @param maxVacuum maximum vacuum level supported by the gripper
00054              * @param defaultGripVacuum default value for the vacuum level for a grip
00055              * action. This value is used for, e.g. performing a "default" grip action
00056              * using the toolbar
00057              * @param unit the unit for all specified values, not <code>null</code>
00058              * @throws InvalidCapabilityRange if <code>minVacuum</code> >
00059              * <code>maxVacuum</code> or <code>defaultGripVacuum</code> is outside the
00060              * range defined by <code>minVacuum</code> and <code>maxVacuum</code>.
00061              */
00062             virtual void updateCapability(double minVacuum, double maxVacuum,
00063                                         double defaultGripVacuum) = 0;
00064         };
00065     } // namespace aubo_scope
00066 } // namespace arcs
00067
00068 #endif

```



```

00027 *
00028 * <p>
00029 * This interface is only relevant if the multi-gripper capability has been
00030 * registered (using the method
00031 * {@link GripperCapabilities#registerWidthCapability(double, double, double,
00032 * double, Length.Unit)}).
00033 * </p>
00034 *
00035 * <p>
00036 * Each capability <b>must</b> be registered using the {@link
00037 * GripperCapabilities} interface (when
00038 * {@link GripperContribution#configureGripper(GripperConfiguration,
00039 * GripperAPIProvider)} is called) before they can be accessed using this
00040 * interface.
00041 * </p>
00042 *
00043 * <b>NOTE:</b> If an update of the properties of a capability needs to apply
00044 * for all individual grippers, it is more convenient to perform the adjustment
00045 * simultaneously for all grippers using the capability instance (e.g.
00046 * {@link WidthCapability}) returned when the capability was registered (with
00047 * the {@link GripperCapabilities} interface).
00048 */
00049 class RegisteredCapabilities
00050 {
00051 public:
00052     virtual ~RegisteredCapabilities() = default;
00053
00054     /**
00055      * <p>
00056      * Access the width capability interface which allows for dynamic
00057      * adjustments of the capability properties exclusively for this
00058      * specific individual gripper (independently of the other grippers),
00059      * including the value range and default value.
00060      * </p>
00061      *
00062      * This is only valid, if the width capability has been registered
00063      * (using the method
00064      * {@link GripperCapabilities#registerWidthCapability(double, double,
00065      * double, double, Length.Unit)}).
00066      *
00067      * @return The width capability for this individual gripper
00068      * @throws UnsupportedOperationException if the gripper has not
00069      * registered the width capability
00070      */
00071     virtual WidthCapabilityPtr getWidthCapability() = 0;
00072
00073     /**
00074      * <p>
00075      * Access the grip force capability interface which allows for dynamic
00076      * adjustments of the capability properties exclusively for this specific
00077      * individual gripper (independently of the other grippers), including the
00078      * value range and default value.
00079      * </p>
00080      *
00081      *
00082      * This is only valid, if the force capability has been registered (using
00083      * the method
00084      * {@link GripperCapabilities#registerGrippingForceCapability(double,
00085      * double, double, Force.Unit)}).
00086      *
00087      * @return The grip force capability for this individual gripper
00088      * @throws UnsupportedOperationException if the gripper has not registered
00089      * the force capability
00090      */
00091     virtual GripForceCapabilityPtr getGripForceCapability() = 0;
00092
00093     /**
00094      * <p>
00095      * Access the speed capability interface which allows for dynamic
00096      * adjustments of the capability properties exclusively for this specific
00097      * individual gripper (independently of the other grippers), including the
00098      * value range and default value.
00099      * </p>
00100      *
00101      * This is only valid, if the speed capability has been registered (using
00102      * the method
00103      * {@link GripperCapabilities#registerSpeedCapability(double, double,
00104      * double, double, Speed.Unit)}).
00105      *
00106      * @return The speed capability for this individual gripper
00107      * @throws UnsupportedOperationException if the gripper has not registered
00108      * the speed capability
00109      */
00110     virtual SpeedCapabilityPtr getSpeedCapability() = 0;
00111
00112     /**
00113      * <p>

```

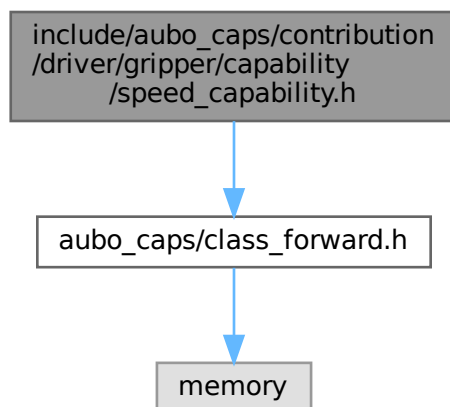
```

00114      * Access the grip vacuum capability interface which allows for dynamic
00115      * adjustments of the capability properties exclusively for this specific
00116      * individual gripper (independently of the other grippers), including the
00117      * value range and default value.
00118      * </p>
00119      *
00120      * This is only valid, if the vacuum capability has been registered (using
00121      * the method
00122      * {@link GripperCapabilities#registerGrippingVacuumCapability(double,
00123      * double, double, Pressure.Unit)}).
00124      *
00125      * @return The grip vacuum capability for this individual gripper
00126      * @throws UnsupportedOperationException if the gripper has not registered
00127      * the vacuum capability
00128      */
00129     virtual GripVacuumCapabilityPtr getGripVacuumCapability() = 0;
00130 };
00131 } // namespace aubo_scope
00132 } // namespace arcs
00133 #endif

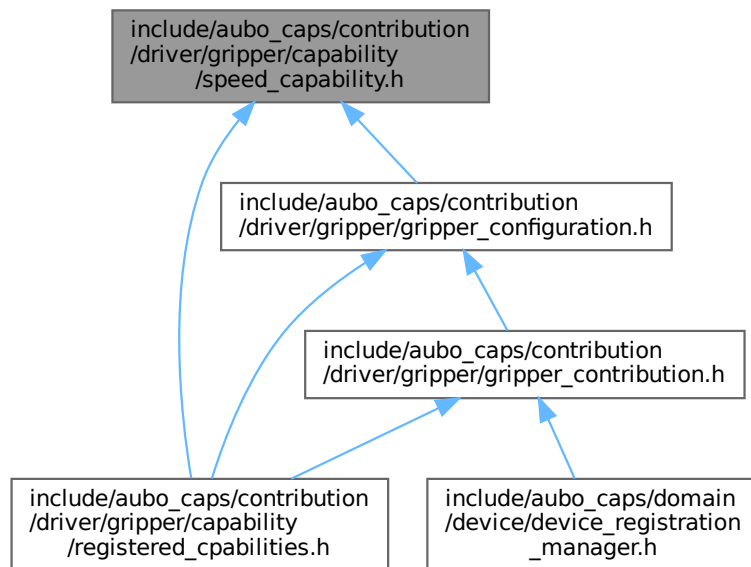
```

12.12 include/aubo_caps/contribution/driver/gripper/↵ capability/speed_capability.h 文件参考

#include <aubo_caps/class_forward.h>
speed_capability.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class `arcs::aubo_scope::SpeedCapability`
速度能力此接口表示夹爪已注册的速度能力，支持使用用户可配置的速度进行操作。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (SpeedCapability)`

12.13 speed_capability.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_SPEED_CAPABILITY_H
00002 #define AUBO_SCOPE_SPEED_CAPABILITY_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(SpeedCapability);
00009     }
00010     /**
00011      * \chinese
00012      * 速度能力
00013      * 此接口表示夹爪已注册的速度能力，支持使用用户可配置的速度进行操作。
00014      *
00015      * 通过此接口，可以在能力注册后调整速度的范围和默认值。这通常不适用于大多数夹爪，
00016      * 但当范围取决于另一个自定义夹爪设置（可由用户配置）时，这可能是相关的。
00017      * \endchinese
00018      * \english
00019      * SpeedCapability
00020      * This interface represents a registered speed capability for a gripper which
00021      * supports operating using a user configurable speed.
  
```

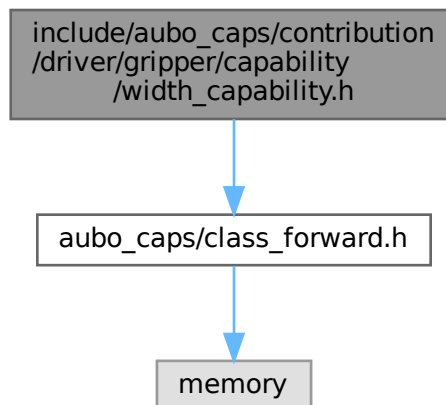
```

00022 *
00023 * Through this interface, it is possible to adjust the range and default values
00024 * for the speed after the capability has been registered. This is typically not
00025 * applicable for most grippers, but can be relevant when the range depends on
00026 * another custom gripper setting (which can be configured by the user).
00027 * \endenglish
00028 */
00029 class SpeedCapability
00030 {
00031 public:
00032     virtual ~SpeedCapability() = default;
00033
00034     /**
00035      * <p>
00036      * Update the range and default values for the speed supported by the
00037      * gripper.
00038      * </p>
00039      *
00040      * Updating the range could result in an already entered value in the
00041      * configuration of a gripper program node being outside the range. This
00042      * will make the given program node undefined, and force the end user to
00043      * adjust the value to be valid again.
00044      *
00045      * @param minSpeed minimum operating speed supported by the gripper
00046      * @param maxSpeed maximum operating speed supported by the gripper
00047      * @param defaultGripSpeed default value for the speed for a grip action.
00048      * This value is used for, e.g. performing a "default" grip action using the
00049      * toolbar
00050      * @param defaultReleaseSpeed default value for the speed for a release
00051      * action. This value is used for, e.g. performing a "default" release
00052      * action using the toolbar
00053      * @param unit the unit for all specified values, not <code>null</code>
00054      * @throws InvalidCapabilityRange if <code>minSpeed</code> >
00055      * <code>maxSpeed</code>
00056      * @throws InvalidCapabilityRange if <code>defaultGripSpeed</code> or
00057      * <code>defaultReleaseSpeed</code> is outside the range defined by
00058      * <code>minSpeed</code> and <code>maxSpeed</code>.
00059      */
00060     virtual void updateCapability(double minSpeed, double maxSpeed,
00061                                   double defaultGripSpeed,
00062                                   double defaultReleaseSpeed) = 0;
00063 };
00064 } // namespace aubo_scope
00065 } // namespace arcs
00066 #endif

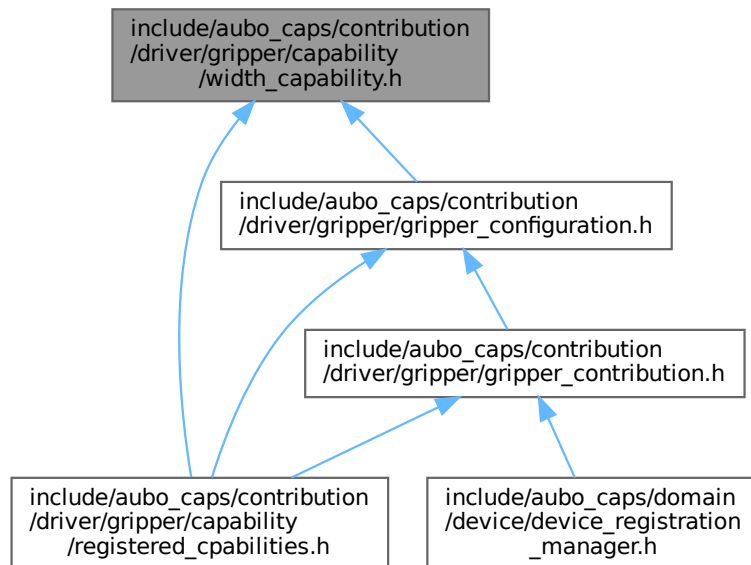
```

12.14 include/auco_caps/contribution/driver/gripper/↵ capability/width_capability.h 文件参考

#include <auco_caps/class_forward.h>
width_capability.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class `arcs::aubo_scope::WidthCapability`
宽度能力此接口表示夹爪已注册的宽度/位置能力，支持移动到用户可配置的位置（打开/关闭到可配置的宽度）。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (WidthCapability)`

12.15 width_capability.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_WIDTH_CAPABILITY_H
00002 #define AUBO_SCOPE_WIDTH_CAPABILITY_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(WidthCapability);
00009     }
00010 }
00011 /**
00012  * \chinese
00013  * 宽度能力
00014  * 此接口表示夹爪已注册的宽度/位置能力，支持移动到用户可配置的位置（打开/关闭到可配置的宽度）。
00015  *
00016  * 通过此接口，可以在能力注册后调整宽度的范围和默认值。这通常不适用于大多数夹爪，
00017  * 但当范围取决于另一个自定义夹爪设置（可由用户配置）时，这可能是相关的。
00018  * \endchinese
00019  * \english
00020  * WidthCapability
  
```

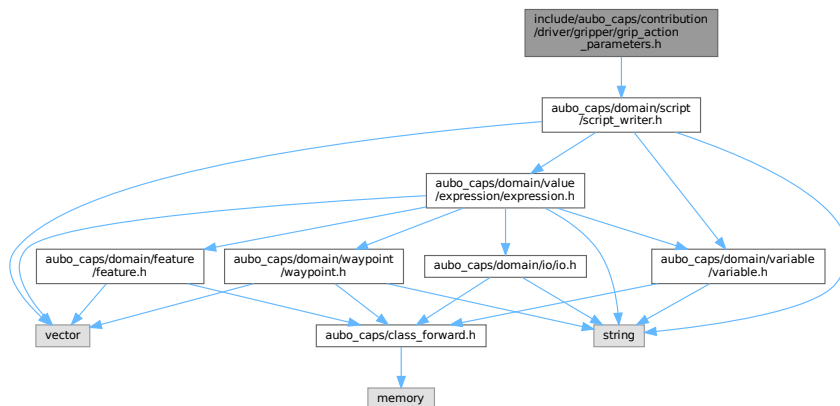
```

00021 * This interface represents a registered width/position capability for a
00022 * gripper which supports moving to a user configurable position (open/close to
00023 * a configurable width).
00024 *
00025 * Through this interface, it is possible to adjust the range and default values
00026 * for the width after the capability has been registered. This is typically not
00027 * applicable for most grippers, but can be relevant when the range depends on
00028 * another custom gripper setting (which can be configured by the user).
00029 * \endenglish
00030 */
00031 class WidthCapability
00032 {
00033 public:
00034     virtual ~WidthCapability() = default;
00035
00036     /**
00037      * <p>
00038      * Update the range and default values for the width supported by the
00039      * gripper.
00040      * </p>
00041      *
00042      * Updating the range could result in an already entered value in the
00043      * configuration of a gripper program node being outside the range. This
00044      * will make the given program node undefined, and force the end user to
00045      * adjust the value to be valid again.
00046      *
00047      * @param minWidth minimum width supported by the gripper
00048      * @param maxWidth maximum width supported by the gripper
00049      * @param defaultGripWidth default value for the width for a grip
00050      * action. This value is used for, e.g. performing a "default" grip
00051      * action using the toolbar
00052      * @param defaultReleaseWidth default value for the width for a release
00053      * action. This value is used for, e.g. performing a "default" release
00054      * action using the toolbar
00055      * @param unit the unit for all specified values, not <code>null</code>
00056      * @throws InvalidCapabilityRange if <code>minWidth</code> > <code>maxWidth</code>
00057      * @throws InvalidCapabilityRange if <code>defaultGripWidth</code> or
00058      * <code>defaultReleaseWidth</code> is outside the range defined by
00059      * <code>minWidth</code> and <code>maxWidth</code>.
00060      */
00061     virtual void updateCapability(double minWidth, double maxWidth,
00062                                   double defaultGripWidth,
00063                                   double defaultReleaseWidth) = 0;
00064 };
00065 } // namespace aubo_scope
00066 } // namespace arcs
00067 #endif

```

12.16 include/aubo_caps/contribution/driver/gripper/grip_← action_parameters.h 文件参考

#include <aubo_caps/domain/script/script_writer.h>
grip_action_parameters.h 的引用 (Include) 关系图:




```

00031      * This is only valid, if the force capability has been registered
00032      * (using the method
00033      * {@link GripperCapabilities#registerGrippingForceCapability(double,
00034      * double, double, Force.Unit)}).
00035      *
00036      * @return The force to use for the gripping action
00037      * @throws UnsupportedOperationException if the gripper has not
00038      * registered the force capability
00039      */
00040      virtual double getForce() = 0;
00041
00042      /**
00043      * <p>
00044      * Get the selected width to be used for the gripping action.
00045      * </p>
00046      *
00047      * This is only valid, if the width capability has been registered (using
00048      * the method
00049      * {@link GripperCapabilities#registerWidthCapability(double, double,
00050      * double, double, Length.Unit)}).
00051      *
00052      * @return The width to use for the gripping action
00053      * @throws UnsupportedOperationException if the gripper has not registered
00054      * the width capability
00055      */
00056      virtual double getWidth() = 0;
00057
00058      /**
00059      * <p>
00060      * Get the selected vacuum level to be used for the gripping action.
00061      * </p>
00062      *
00063      * This is only valid, if the vacuum capability has been registered (using
00064      * the method
00065      * {@link GripperCapabilities#registerGrippingVacuumCapability(double,
00066      * double, double, Pressure.Unit)}).
00067      *
00068      * @return The vacuum level to use for the gripping action
00069      * @throws UnsupportedOperationException if the gripper has not registered
00070      * the vacuum capability
00071      */
00072      virtual double getVacuum() = 0;
00073
00074      /**
00075      * <p>
00076      * Get the selected speed to be used for the gripping action.
00077      * </p>
00078      *
00079      * This is only valid, if the speed capability has been registered (using
00080      * the method
00081      * {@link GripperCapabilities#registerSpeedCapability(double, double,
00082      * double, double, Speed.Unit)}).
00083      *
00084      * @return The speed to use for the gripping action
00085      * @throws UnsupportedOperationException if the gripper has not registered
00086      * the speed capability
00087      */
00088      virtual double getSpeed() = 0;
00089
00090      /**
00091      * <p>
00092      * Use this method to determine whether the grip detection option in the
00093      * Gripper program node is on or off.
00094      * </p>
00095      *
00096      * <p>
00097      * Note that this method will report that grip detection is off, when script
00098      * code is to be generated due to the user testing a grip configuration of a
00099      * gripper program or operating the gripper using the toolbar.
00100      * </p>
00101      *
00102      * <p>
00103      * This method should only be called, if the grip detected feedback
00104      * capability has been registered (using
00105      * {@link
00106      * GripperFeedbackCapabilities#registerGripDetectedCapability(ScriptCodeGenerator)}).
00107      * </p>
00108      *
00109      * <p>
00110      * The purpose of this method is to ensure that the appropriate script code
00111      * for performing a grip action is generated when the end user has specified
00112      * a new payload value. The user-defined payload value will be applied by
00113      * AuboScope immediately after the generated grip action script code has
00114      * finished executing. Depending on the enablement state of grip detection,
00115      * the generated script should have the following properties: <ul> <li>
00116      * <i>Grip detection off:</i> The script code should not finish
00117      * earlier than when it is appropriate to apply the new payload value (i.e.

```



```

00118      * when the object has been gripped). This typically means that the script
00119      * code should wait a time period corresponding to closing the gripper's
00120      * "fingers" fully (if the gripper is vacuum operated then long enough to
00121      * achieve some level of vacuum).
00122      *      </li>
00123      *      <li>
00124      *          <i>Grip detection on:</i> The payload will only be set if the
00125      *          grip detection was successful, i.e. it will be applied immediately after
00126      *          a grip was detected. This means that it is not necessary for the script
00127      *          code to wait (since the built-in detection timeout functionality will
00128      *          handle the waiting part).
00129      *      </li>
00130      * </ul>
00131      * </p>
00132      *
00133      * @return <code>true</code> if the end user has enabled grip detection, and
00134      * <code>false</code> if grip detection is disabled.
00135      * @throws UnsupportedOperationException if the gripper has not registered
00136      * the grip detected feedback capability
00137      */
00138      virtual bool isGripDetectionEnabled() = 0;
00139 };
00140 } // namespace aubo_scope
00141 } // namespace arcs
00142 #endif

```

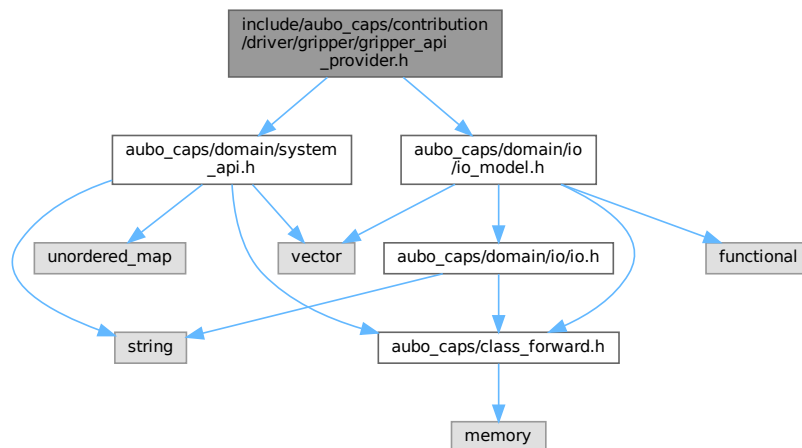
12.18 include/aubo_caps/contribution/driver/gripper/↵ gripper_api_provider.h 文件参考

```

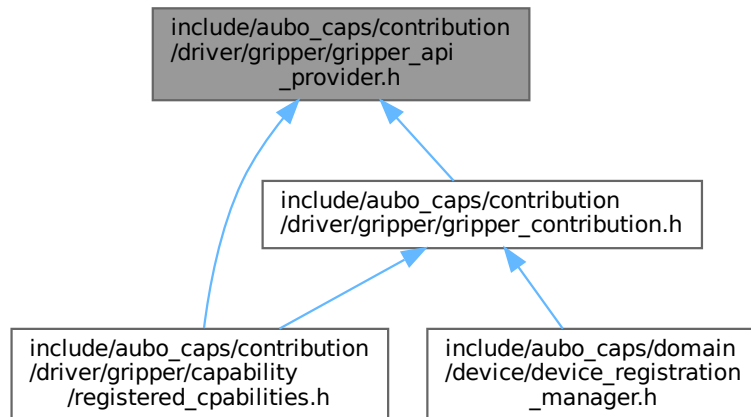
#include <aubo_caps/domain/system_api.h>
#include <aubo_caps/domain/io/io_model.h>

```

gripper_api_provider.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- `class arcs::aubo_scope::GripperApiProvider`
夹爪 API 提供者此接口提供对 *AuboScope* 中与夹爪相关的功能和服务的访问。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (GripperApiProvider)`

12.19 gripper_api_provider.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_GRIPPER_API_PROVIDER_H
00002 #define AUBO_SCOPE_GRIPPER_API_PROVIDER_H
00003
00004 #include <aubo_caps/domain/system_api.h>
00005 #include <aubo_caps/domain/io/io_model.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(GripperApiProvider);
00010
00011 /**
00012  * \chinese
00013  * 夹爪 API 提供者
00014  * 此接口提供对 AuboScope 中与夹爪相关的功能和服务的访问。
00015  * \endchinese
00016  * \english
00017  * GripperApiProvider
00018  * This interface provides access to functionality and services available from
00019  * within AuboScope relevant for gripper contributions.
00020  * \endenglish
00021  */
00022 class GripperApiProvider
00023 {
00024 public:
00025     virtual ~GripperApiProvider() = default;
00026
00027     /**
00028      * \chinese
  
```

```

00029      * @return 提供系统相关功能访问的接口
00030      * \endchinese
00031      * \english
00032      * @return An interface providing access to system related functionality.
00033      * \endenglish
00034      */
00035      virtual SystemApiPtr getSystemApi() = 0;
00036
00037      /**
00038      * \chinese
00039      * @return I/O 模型接口 (例如用于获取在设置夹爪的自定义用户输入中显示的 I/O)
00040      * \endchinese
00041      * \english
00042      * @return An I/O model interface (e.g. for getting I/Os to display in the
00043      * custom user input for setting up the gripper)
00044      * \endenglish
00045      */
00046      virtual IoModelPtr getIoModel() = 0;
00047 };
00048 } // namespace aubo_scope
00049 } // namespace arcs
00050
00051 #endif

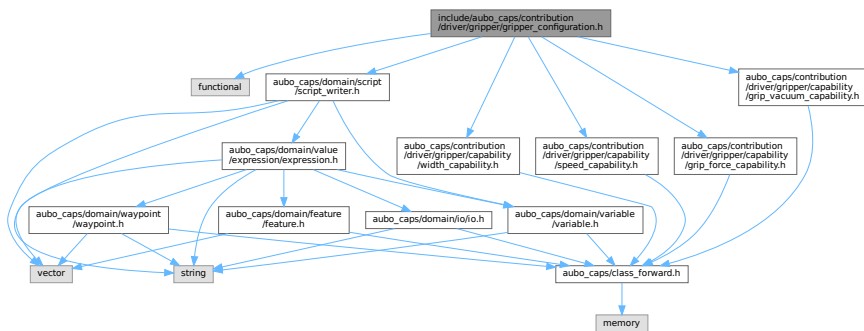
```

12.20 include/aubo_caps/contribution/driver/gripper/↵ gripper_configuration.h 文件参考

```

#include <functional>
#include <aubo_caps/domain/script/script_writer.h>
#include <aubo_caps/contribution/driver/gripper/capability/width_capability.h>
#include <aubo_caps/contribution/driver/gripper/capability/speed_capability.h>
#include <aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h>
#include <aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h>
gripper_configuration.h 的引用 (Include) 关系图:

```




```

00030      * @param maxWidth The maximum width supported by the gripper
00031      * @param defaultGripWidth default value for the width for a grip action.
00032      * This value is used for, e.g. performing a "default" grip action using the
00033      * toolbar
00034      * @param defaultReleaseWidth default value for the width for a release
00035      * action. This value is used for, e.g. performing a "default" release
00036      * action using the toolbar
00037      * @param unit the unit for all specified values, not <code>null</code>
00038      * @return a width capability allowing for dynamic adjustments of the
00039      * capability properties including the value range and default values
00040      * @throws InvalidCapabilityRange if <code>minWidth</code> >
00041      * <code>maxWidth</code>
00042      * @throws InvalidCapabilityRange if <code>defaultGripWidth</code> or
00043      * <code>defaultReleaseWidth</code> is outside the range defined by
00044      * <code>minWidth</code> and <code>maxWidth</code>.
00045      * @throws CapabilityAlreadyRegistered if this capability has already been
00046      * registered
00047      * @throws CalledOutsideGripperConfigurationPhase if this method is called
00048      * at the wrong time by a AuboCap implementing the {@link
00049      * GripperContribution} interface, i.e. called outside the scope of the
00050      * configuration of the gripper.
00051      * @throws CalledOutsideMethodScope if this method is called at the wrong
00052      * time by a (regular) AuboCap which has registered its program node
00053      * contribution/service as conformant with a AuboScope gripper (using
00054      * {@link GripperRegistrationManager#registerAsGripper(Class)}),
00055      * i.e. called outside the scope of the call to either {@link
00056      * ProgramNodeService#configureContribution(ContributionConfiguration)}
00057      * (for a Qt-based AuboCap) or {@link
00058      * ProgramNodeServiceConfigurable#configureContribution(ProgramNodeConfiguration)}
00059      * (for a HTML-based AuboCap).
00060      */
00061      virtual WidthCapabilityPtr registerWidthCapability(
00062          double minWidth, double maxWidth, double defaultGripWidth,
00063          double defaultReleaseWidth) = 0;
00064
00065      /**
00066      * Register a force capability for a gripper that supports gripping using a
00067      * user configurable force.
00068      *
00069      * @param minForce minimum force supported by the gripper
00070      * @param maxForce maximum force supported by the gripper
00071      * @param defaultGripForce default value for the force for a grip action.
00072      * This value is used for, e.g. performing a "default" grip action using the
00073      * toolbar
00074      * @param unit the unit for all specified values, <code>null</code>
00075      * @return a force capability allowing for dynamic adjustments of the
00076      * capability properties including the value range and default value
00077      * @throws InvalidCapabilityRange if <code>minForce</code> >
00078      * <code>maxForce</code> or <code>defaultGripForce</code> is outside the
00079      * range defined by <code>minForce</code> and <code>maxForce</code>.
00080      * @throws CapabilityAlreadyRegistered if this capability has already been
00081      * registered
00082      * @throws CalledOutsideGripperConfigurationPhase if this method is called
00083      * at the wrong time by a AuboCap implementing the {@link
00084      * GripperContribution} interface, i.e. called outside the scope of the
00085      * configuration of the gripper.
00086      * @throws CalledOutsideMethodScope if this method is called at the wrong
00087      * time by a (regular) AuboCap which has registered its program node
00088      * contribution/service as conformant with a AuboScope gripper (using
00089      * {@link GripperRegistrationManager#registerAsGripper(Class)}),
00090      * i.e. called outside the scope of the call to either {@link
00091      * ProgramNodeService#configureContribution(ContributionConfiguration)}
00092      * (for a Qt-based AuboCap) or {@link
00093      * ProgramNodeServiceConfigurable#configureContribution(ProgramNodeConfiguration)}
00094      * (for a HTML-based AuboCap).
00095      */
00096      virtual GripForceCapabilityPtr registerGrippingForceCapability(
00097          double minForce, double maxForce, double defaultGripForce) = 0;
00098
00099      /**
00100      * Register a speed capability for a gripper that supports operating using a
00101      * user configurable speed.
00102      *
00103      * @param minSpeed minimum speed supported by the gripper
00104      * @param maxSpeed maximum speed supported by the gripper
00105      * @param defaultGripSpeed default value for the speed for a grip action.
00106      * This value is used for, e.g. performing a "default" grip action using the
00107      * toolbar
00108      * @param defaultReleaseSpeed default value for the speed for a grip action.
00109      * This value is used for, e.g. performing a "default" release action using
00110      * the toolbar
00111      * @param unit the unit for all specified values, not <code>null</code>
00112      * @return a speed capability allowing for dynamic adjustments of the
00113      * capability properties including the value range and default values
00114      * @throws InvalidCapabilityRange if <code>minSpeed</code> >
00115      * <code>maxSpeed</code>
00116      * @throws InvalidCapabilityRange if <code>defaultGripSpeed</code> or

```

```

00117     * defaultReleaseSpeed</code> is outside the range defined by
00118     * <code>minSpeed</code> and <code>maxSpeed</code>.
00119     * @throws CapabilityAlreadyRegistered if this capability has already been
00120     * registered
00121     * @throws CalledOutsideGripperConfigurationPhase if this method is called
00122     * at the wrong time by a AuboCap implementing the {@link
00123     * GripperContribution} interface, i.e. called outside the scope of the
00124     * configuration of the gripper.
00125     * @throws CalledOutsideMethodScope if this method is called at the wrong
00126     * time by a (regular) AuboCap which has registered its program node
00127     * contribution/service as conformant with a AuboScope gripper (using
00128     * {@link GripperRegistrationManager#registerAsGripper(Class)}),
00129     * i.e. called outside the scope of the call to either {@link
00130     * ProgramNodeService#configureContribution(ContributionConfiguration)}
00131     * (for a Qt-based AuboCap) or {@link
00132     * ProgramNodeServiceConfigurable#configureContribution(ProgramNodeConfiguration)}
00133     * (for a HTML-based AuboCap).
00134     */
00135     virtual SpeedCapabilityPtr registerSpeedCapability(
00136         double minSpeed, double maxSpeed, double defaultGripSpeed,
00137         double defaultReleaseSpeed) = 0;
00138
00139     /**
00140     * <p>
00141     * Register a vacuum capability for a gripper that supports gripping using a
00142     * user configurable vacuum level.
00143     * </p>
00144     *
00145     * If the gripper works with absolute pressure, specify a positive range for
00146     * the supported vacuum level. If the gripper works with relative pressure
00147     * (vacuum), specify a negative range for the supported vacuum level.
00148     *
00149     * @param minVacuum minimum vacuum level supported by the gripper
00150     * @param maxVacuum maximum vacuum level supported by the gripper
00151     * @param defaultGripVacuum default value for the vacuum level for a grip
00152     * action. This value is used for, e.g. performing a "default" grip action
00153     * using the toolbar
00154     * @param unit the unit for all specified values, not <code>null</code>
00155     * @return a vacuum capability allowing for dynamic adjustments of the
00156     * capability properties including the value range and default value
00157     * @throws InvalidCapabilityRange if <code>minVacuum</code> >
00158     * <code>maxVacuum</code> or <code>defaultGripVacuum</code> is outside the
00159     * range defined by <code>minVacuum</code> and <code>maxVacuum</code>.
00160     * @throws CapabilityAlreadyRegistered if this capability has already been
00161     * registered
00162     * @throws CalledOutsideGripperConfigurationPhase if this method is called
00163     * at the wrong time by a AuboCap implementing the {@link
00164     * GripperContribution} interface, i.e. called outside the scope of the
00165     * configuration of the gripper.
00166     * @throws CalledOutsideMethodScope if this method is called at the wrong
00167     * time by a (regular) AuboCap which has registered its program node
00168     * contribution/service as conformant with a AuboScope gripper (using
00169     * {@link GripperRegistrationManager#registerAsGripper(Class)}),
00170     * i.e. called outside the scope of the call to either {@link
00171     * ProgramNodeService#configureContribution(ContributionConfiguration)}
00172     * (for a Qt-based AuboCap) or {@link
00173     * ProgramNodeServiceConfigurable#configureContribution(ProgramNodeConfiguration)}
00174     * (for a HTML-based AuboCap).
00175     */
00176     virtual GripVacuumCapabilityPtr registerGrippingVacuumCapability(
00177         double minVacuum, double maxVacuum, double defaultGripVacuum) = 0;
00178
00179     /**
00180     * <p>
00181     * Register this feedback capability if the gripper can inform AuboScope
00182     * whether it has detected that an object has been gripped, after a grip
00183     * action has been triggered. This will enable the end user to configure the
00184     * actions to take when the gripper has detected that an object has been
00185     * gripped and/or when the grip action timed out.
00186     * </p>
00187     *
00188     * <p>
00189     * The provided implementation of the {@link ScriptCodeGenerator} interface
00190     * must generate the script code for determining if a grip was detected. The
00191     * return value of the script code <b>must</b> be a boolean, i.e. 'True' or
00192     * 'False'.
00193     * </p>
00194     *
00195     * <b>Note:</b> When this capability has been registered, the
00196     * {@link GripperContribution#generateGripActionScript(ScriptWriter,
00197     * GripActionParameters)} method should generate different grip action
00198     * script code depending on whether the end user has enabled or disabled the
00199     * grip detection option (in the Gripper program node). For more details,
00200     * see Javadoc for
00201     * {@link GripActionParameters#isGripDetectionEnabled()}.
00202     *
00203     * @param scriptCodeGenerator script generator that generates the script

```

```
00204      * code for determining if a grip was detected. When the script code needs
00205      * to be generated, the method
00206      *
00207      * {@link
00208      *   ScriptCodeGenerator#generateScript(ScriptWriter, Object)} will be called
00209      * (by AuboScope). The resulting script code will be embedded in a script
00210      * function and <must> return a boolean, i.e. 'True' or 'False'.
00211      * @throws CapabilityAlreadyRegistered if this capability has already been
00212      * registered
00213      * @throws CalledOutsideGripperConfigurationPhase if this method is called
00214      * at the wrong time, i.e. outside the scope of the {@link
00215      *   GripperContribution#configureGripper(GripperConfiguration,
00216      *   GripperAPIProvider)} method.
00217      */
00218 virtual void registerGripDetectedCapability(
00219     std::function<void(ScriptWriterPtr scriptWriter)>
00220     scriptCodeGenerator) = 0;
00221 /**
00222  * <p>
00223  * Register this feedback capability if the gripper can inform AuboScope
00224  * whether it has detected an object has been released, after a release
00225  * action has been triggered. This will enable the end user to configure the
00226  * actions to take when the gripper has detected that an object has been
00227  * released and/or when the release action timed out.
00228  * </p>
00229  *
00230  * <p>
00231  * The provided implementation of the {@link ScriptCodeGenerator} interface
00232  * must generate the script code for determining if a release was detected.
00233  * The return value of the script code <must> be a boolean, i.e. 'True'
00234  * or 'False'.
00235  * </p>
00236  *
00237  * <b>Note:</b> When this capability has been registered, the
00238  * {@link GripperContribution#generateReleaseActionScript(ScriptWriter,
00239  *   ReleaseActionParameters)} method should generate different release action
00240  * script code depending on whether the end user has enabled or disabled the
00241  * release detection option (in the Gripper program node). For more details,
00242  * see Javadoc for
00243  * {@link GripActionParameters#isGripDetectionEnabled()}.
00244  *
00245  * @param scriptCodeGenerator script generator that generates the script
00246  * code for determining if a release was detected. When the script code
00247  * needs to be generated, the method
00248  *
00249  * {@link
00250  *   ScriptCodeGenerator#generateScript(ScriptWriter, Object)} will be called
00251  * (by AuboScope). The resulting script code will be embedded in a script
00252  * function and <must> return a boolean, i.e. 'True' or 'False'.
00253  * @throws CapabilityAlreadyRegistered if this capability has already been
00254  * registered
00255  * @throws CalledOutsideGripperConfigurationPhase if this method is called
00256  * at the wrong time, i.e. outside the scope of the {@link
00257  *   GripperContribution#configureGripper(GripperConfiguration,
00258  *   GripperAPIProvider)} method.
00259  */
00259 virtual void registerReleaseDetectedCapability(
00260     std::function<void(ScriptWriterPtr scriptWriter)>
00261     scriptCodeGenerator) = 0;
00262 };
00263 } // namespace aubo_scope
00264 } // namespace arcs
00265
00266 #endif
```

12.22 include/aubo_caps/contribution/driver/gripper/↵ gripper_contribution.h 文件参考

```
#include <aubo_caps/contribution/driver/gripper/gripper_api_provider.h>
#include <aubo_caps/contribution/driver/gripper/gripper_configuration.h>
#include <aubo_caps/contribution/driver/gripper/release_action_parameters.h>
#include <aubo_caps/contribution/driver/gripper/grip_action_parameters.h>
#include <aubo_caps/domain/system_api.h>
#include <aubo_caps/domain/script/script_writer.h>
```



```

00017 * \chinese
00018 * 夹具贡献
00019 * 此接口定义了向 AuboScope 添加夹具设备功能的 API。
00020 *
00021 * 实现此接口将创建程序节点贡献、安装贡献和工具栏贡献，向终端用户暴露夹具功能。
00022 * 请注意，工具栏贡献在 CB3 机器人上不受支持。
00023 *
00024 * 程序节点贡献允许终端用户编程夹取和释放操作。终端用户可用的配置选项基于夹具
00025 * 支持的属性和功能。至少，所有夹具必须支持"默认"夹取和释放操作。所有其他功能
00026 * 是可选的，可以在调用 {@link GripperContribution#configureGripper(GripperConfiguration,
00027 * GripperAPIProvider)} 方法时注册。
00028 *
00029 * 安装节点贡献可在需要终端用户提供任何自定义输入来设置夹具时使用（例如指定 IP 地址）。
00030 * 相应的生成 UI 将在安装节点中可访问。此功能是可选的，因为此类配置属性是特定于
00031 * 每个夹具的。当调用
00032 * {@link GripperContribution#configureInstallation(CustomUserInputConfiguration,
00033 * SystemConfiguration, TCPConfiguration, GripperAPIProvider)} 方法时，提供创建
00034 * 基于用户输入的自定义配置的访问权限。如果没有定义自定义用户输入、文本等，安装节点
00035 * 将不会在 AuboScope 中可见。前言脚本代码（由
00036 * {@link #generatePreambleScript(ScriptWriter)} 生成）仍将添加到机器人程序中。
00037 *
00038 * 工具栏贡献为终端用户提供快速访问，以使用"默认"夹具操作参数执行夹取和释放操作。
00039 * \endchinese
00040 * \english
00041 * GripperContribution
00042 * This interface defines an API for adding functionality for gripping devices
00043 * to AuboScope.
00044 *
00045 * Implementing this interface will create a program node contribution, an
00046 * installation contribution and a toolbar contribution exposing gripper
00047 * functionality to the end user. Note that the toolbar contribution is not
00048 * supported on CB3 robots.
00049 *
00050 * The program node contribution allows the end user to program grip and release
00051 * actions. The configuration options available to the end user are based on the
00052 * properties and capabilities supported by the gripper. As a minimum, all
00053 * grippers must support "default" grip and release actions. All other
00054 * capabilities are optional to support and can be registered when the {@link
00055 * GripperContribution#configureGripper(GripperConfiguration,
00056 * GripperAPIProvider)} method is called.
00057 *
00058 * The installation node contribution can be used, if any custom input from the
00059 * end user is required to setup the gripper, e.g. specifying an IP address. The
00060 * corresponding generated UI will be accessible in the installation node. This
00061 * functionality is optional, since such configuration properties are specific
00062 * for each gripper. Access to create a custom configuration based on user
00063 * inputs is provided when the method
00064 * {@link
00065 * GripperContribution#configureInstallation(CustomUserInputConfiguration,
00066 * SystemConfiguration, TCPConfiguration, GripperAPIProvider)} is called. If no
00067 * custom user inputs, texts, etc. are defined, the installation node will not
00068 * be visible in AuboScope. Preamble script code (generated by {@link
00069 * #generatePreambleScript(ScriptWriter)}) will still be added to the robot
00070 * program.
00071 *
00072 * The toolbar contribution provides the end user quick access to perform grip
00073 * and release actions using "default" gripper action parameters.
00074 * \endenglish
00075 */
00076 class GripperContribution
00077 {
00078 public:
00079     virtual ~GripperContribution() = default;
00080
00081     /**
00082      * \chinese
00083      * 此方法必须返回夹具贡献的标题。标题显示在以下位置：
00084      * <ul>
00085      * <li> 程序标签页的"结构"区域中此类夹具程序节点的显示（可从此处插入夹具程序节点）</li>
00086      * <li> 此夹具贡献的程序节点画面 </li>
00087      * <li> 程序树中此夹具贡献程序节点的标题 </li>
00088      * <li> 安装标签页左侧导航中此夹具贡献的显示 </li>
00089      * <li> 此夹具贡献的安装节点画面 </li>
00090      * <li> 此夹具的工具栏贡献。注意：工具栏贡献在 CB3 机器人上不可用 </li>
00091      * </ul>
00092      *
00093      * <p> 此方法的返回值不能为 <code>null</code>，也不能为空字符串。</p>
00094      * 此方法在启动时调用一次。
00095      *
00096      * @param locale AuboScope 的当前语言环境，可用于支持多种语言的标题
00097      * @return 此夹具贡献的标题，不能为 <code>null</code> 也不能为空字符串
00098      * \endchinese
00099      * \english
00100      * This method must return the title of the gripper contribution. The title
00101      * is displayed in: <ul> <li>The Structure section of the Program Tab for
00102      * this type of gripper program node (from which the Gripper program node
00103      * can be inserted)</li> <li>The program node screen for this gripper

```

```

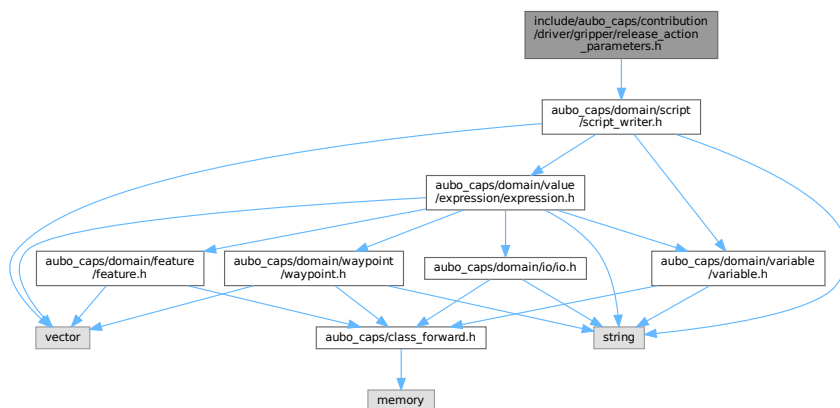
00104      * contribution.</li> <li>The Program Tree as title for the program node for
00105      * this gripper contribution</li> <li>The left-hand side navigation of the
00106      * Installation Tab for this gripper contribution.</li> <li>The installation
00107      * node screen for this gripper contribution.</li> <li>The toolbar
00108      * contribution for this gripper. Note: The toolbar contribution is not
00109      * available on CB3 robots.</li>
00110      * </ul>
00111      *
00112      * <p>
00113      * The return value of this method cannot be <code>null</code> nor an empty
00114      * string.
00115      * </p>
00116      *
00117      * This method is called once upon startup.
00118      *
00119      * @param locale The current locale of AuboScope. Can be used for supporting
00120      * titles in several languages.
00121      * @return The title of this gripper contribution, not <code>null</code> nor
00122      * an empty string.
00123      * \endenglish
00124      */
00125 virtual std::string getTitle(/*Locale locale*/) = 0;
00126
00127 /**
00128      * \chinese
00129      * <p> 当调用此方法时, 使用 configuration 参数注册或设置夹爪的可选属性和功能。</p>
00130      * <p> 此方法在夹爪配置阶段调用, 在贡献注册后、新安装被加载或创建时调用。</p>
00131      * <b> 注意: </b> 如果夹爪仅支持基本的"默认" 夹取和释放操作, 请将此方法的实现留空。
00132      *
00133      * @param gripperConfiguration 可用于注册或设置夹爪属性和功能的配置实例
00134      * @param apiProvider 提供对 AuboScope 中可用于设置夹爪功能的相关功能和服务访问
00135      * \endchinese
00136      * \english
00137      * <p>
00138      * When this method is called, use the configuration parameter to register
00139      * or setup optional properties and capabilities of the gripper.
00140      * </p>
00141      *
00142      * <p>
00143      * The method is called in the Gripper Configuration phase after this
00144      * contribution has been registered when a new installation is loaded or
00145      * created.
00146      * </p>
00147      *
00148      * <b>Note:</b> If the gripper only supports basic "default" grip and
00149      * release actions, leave the implementation of this method empty.
00150      *
00151      * @param gripperConfiguration A configuration instance that can be used for
00152      * registering or setting up the properties and capabilities of the gripper.
00153      * @param apiProvider Provides access to functionality and services
00154      * available from within AuboScope which can be relevant for setting up the
00155      * gripper capabilities (e.g. the {@link SystemAPI} interface with
00156      * functionality for querying information about the robot).
00157      * \endenglish
00158      */
00159 virtual void configureGripper(GripperConfigurationPtr gripperConfiguration,
00160                               GripperApiProviderPtr apiProvider) = 0;
00161
00162 /**
00163      * \chinese
00164      * <p> 当调用此方法时, 必须生成使用夹爪执行夹取动作的脚本代码。</p>
00165      * <p> 终端用户定义/配置的已注册可选爪功能参数将作为脚本生成的输入。</p>
00166      * <p> 此方法在以下场景中调用: </p>
00167      * <ul>
00168      * <li> 当配置了夹取动作的夹爪程序节点应作为要执行的机器人程序的一部分生成脚本代码时 </li>
00169      * <li> 当终端用户测试夹爪程序节点的夹取配置时 </li>
00170      * <li> 当使用工具栏操作夹爪时 </li>
00171      * </ul>
00172      *
00173      * <b> 注意: </b>
00174      * <ul>
00175      * <li> 夹取物体后应用新的总负载 <b> 不是 </b> 夹爪的职责, 因为这是 AuboScope 自动处理的。</li>
00176      * <li> 当终端用户指定了新的负载时, 用户定义的负载值将在夹取动作的生成脚本代码完成
00177      * 执行后立即由 AuboScope 应用。因此脚本代码不应在适当应用新负载值之前完成。
00178      * </li>
00179      * </ul>
00180      *
00181      * @param scriptWriter 使用此脚本编写器实例生成夹取脚本代码
00182      * @param parameters 终端用户定义/配置的夹爪动作参数
00183      * \endchinese
00184      * \english
00185      * <p>
00186      * When this method is called, the script code for performing a grip action
00187      * with the gripper must be generated.
00188      * </p>
00189      *
00190      * ...

```

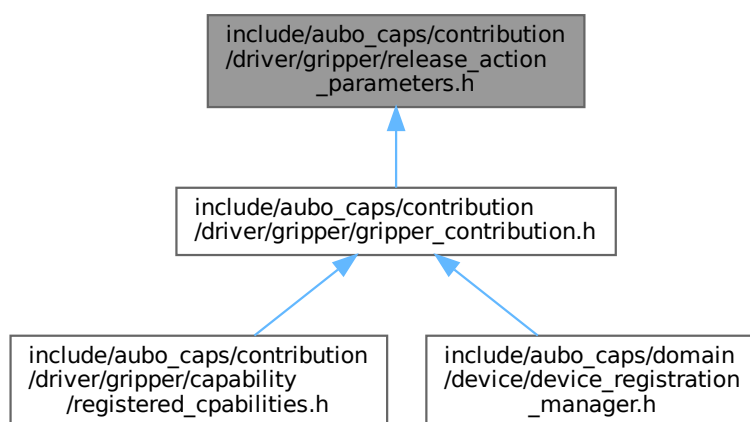
```
00191      * @param scriptWriter Use this script writer instance to generate the
00192      * script code for gripping
00193      * @param parameters The parameters for the gripper action
00194      * defined/configured by the end user
00195      * \endenglish
00196      */
00197 virtual void generateGripActionScript(
00198     ScriptWriterPtr scriptWriter, GripActionParametersPtr parameters) = 0;
00199
00200 /**
00201  * \chinese
00202  * <p> 当调用此方法时，必须生成使用夹爪执行释放动作的脚本代码。</p>
00203  * <p> 终端用户定义/配置的已注册可选夹爪功能参数将作为脚本生成的输入。</p>
00204  * <p> 此方法在以下场景中调用：</p>
00205  * <ul>
00206  * <li> 当配置了释放动作的夹爪程序节点应作为要执行的机器人程序的一部分生成脚本代码时 </li>
00207  * <li> 当终端用户测试夹爪程序节点的释放配置时 </li>
00208  * <li> 当使用工具栏操作夹爪时 </li>
00209  * </ul>
00210  *
00211  * <b> 注意：</b>
00212  * <ul>
00213  * <li> 释放物体后应用新的总负载 <b> 不是 </b> 夹爪的职责，因为这是 AuboScope 自动处理的。</li>
00214  * <li> 当终端用户指定了新的负载时，用户定义的负载值将在释放动作的生成脚本代码完成
00215  * 执行后立即由 AuboScope 应用。因此脚本代码不应在适当应用新负载值之前完成。</li>
00216  * </ul>
00217  *
00218  * @param scriptWriter 使用此脚本编写器实例生成释放夹爪的脚本代码
00219  * @param parameters 终端用户定义/配置的夹爪动作参数
00220  * \endchinese
00221  * \english
00222  * <p>
00223  * When this method is called, the script code for performing a release
00224  * action with the gripper must be generated.
00225  * </p>
00226  * ...
00227  * @param scriptWriter Use this script writer instance to generate the
00228  * script code for releasing the gripper
00229  * @param parameters The parameters for the gripper action
00230  * defined/configured by the end user
00231  * \endenglish
00232  */
00233 virtual void generateReleaseActionScript(
00234     ScriptWriterPtr scriptWriter,
00235     ReleaseActionParametersPtr parameters) = 0;
00236 };
00237 } // namespace aubo_scope
00238 } // namespace arcs
00239 #endif
```

12.24 include/aubo_caps/contribution/driver/gripper/release_↵ action_parameters.h 文件参考

#include <aubo_caps/domain/script/script_writer.h>
release_action_parameters.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::ReleaseActionParameters`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ReleaseActionParameters)`

12.25 release_action_parameters.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_RELEASE_ACTION_PARAMETERS_H
00002 #define AUBO_SCOPE_RELEASE_ACTION_PARAMETERS_H
00003
00004 #include <aubo_caps/domain/script/script_writer.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008
00009 ARCS_CLASS_FORWARD(ReleaseActionParameters);
00010 /**
00011  * <p>
00012  * This interface represents parameters for a release action defined/configured
00013  * by the end user.
00014  * </p>
00015  *
00016  * These parameters are used when generating script code for a release action
00017  * (see
00018  * {@link GripperContribution#generateReleaseActionScript(ScriptWriter,
00019  * ReleaseActionParameters)}).
00020  */
00021 class ReleaseActionParameters
00022 {
00023 public:
00024     virtual ~ReleaseActionParameters() = default;
00025
00026     /**
00027      * <p>
00028      * Get the selected width to be used for releasing the gripper.
00029      * </p>
00030      *

```

```

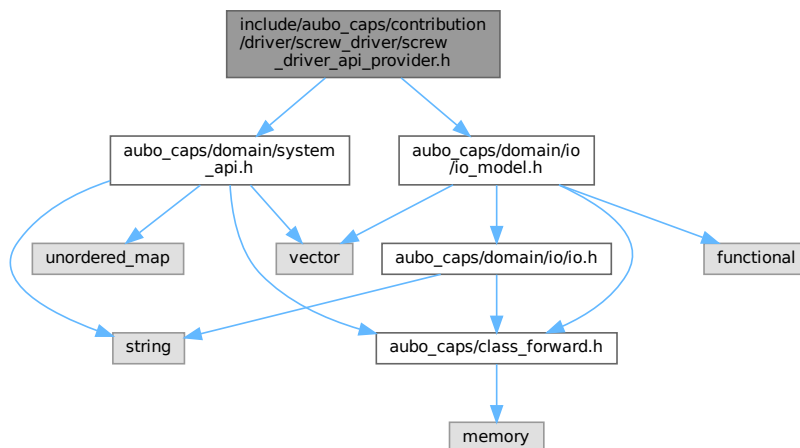
00031      * This is only valid, if the width capability has been registered (using
00032      * the method
00033      * {@link GripperCapabilities#registerWidthCapability(double, double,
00034      * double, double, Length.Unit)}}).
00035      *
00036      * @return The width to use for the release action
00037      * @throws UnsupportedOperationException if the gripper has not registered
00038      * the width capability
00039      */
00040      virtual double getWidth() = 0;
00041
00042      /**
00043      * <p>
00044      * Get the selected speed to be used for releasing the gripper.
00045      * </p>
00046      *
00047      * This is only valid, if the speed capability has been registered (using
00048      * the method
00049      * {@link GripperCapabilities#registerSpeedCapability(double, double,
00050      * double, double, Speed.Unit)}}).
00051      *
00052      * @return The speed to use for the release action
00053      * @throws UnsupportedOperationException if the gripper has not registered
00054      * the speed capability
00055      */
00056      virtual double getSpeed() = 0;
00057
00058      /**
00059      * <p>
00060      * Use this method to determine whether the release detection option in the
00061      * Gripper program node is on or off.
00062      * </p>
00063      *
00064      * <p>
00065      * Note that this method will report that release detection is off, when
00066      * script code is to be generated due to the user testing a release
00067      * configuration of a gripper program or operating the gripper using the
00068      * toolbar.
00069      * </p>
00070      *
00071      * <p>
00072      * This method should only be called, if the release detected feedback
00073      * capability has been registered (using
00074      * {@link
00075      * GripperFeedbackCapabilities#registerReleaseDetectedCapability(ScriptCodeGenerator)}}).
00076      * </p>
00077      *
00078      * <p>
00079      * The purpose of this method is to ensure that the appropriate script code
00080      * for performing a release action is generated when the end user has
00081      * specified a new payload value. The user-defined payload value will be
00082      * applied by AutoScope immediately after the generated release action
00083      * script code has finished executing. Depending on the enablement state of
00084      * release detection, the generated script should have the following
00085      * properties: <ul> <li> <i>Release detection off:</i> The script code
00086      * should not finish earlier than when it is appropriate to apply the new
00087      * payload value (i.e. when the object has been released). This typically
00088      * means that the script code should wait a time period corresponding to
00089      * opening the gripper's "fingers" fully (if the gripper is vacuum operated
00090      * then long enough to achieve some level of vacuum).
00091      * </li>
00092      * <li>
00093      * <i>Release detection on:</i> The payload will only be set if the
00094      * release detection was successful, i.e. it will be applied immediately
00095      * after a release was detected. This means that it is not necessary for the
00096      * script code to wait (since the built-in detection timeout
00097      * functionality will handle the waiting part).
00098      * </li>
00099      * </ul>
00100      * </p>
00101      *
00102      * @return <code>true</code> if the end user has enabled release detection,
00103      * and <code>false</code> if release detection is disabled.
00104      * @throws UnsupportedOperationException if the gripper has not registered
00105      * the release detected feedback capability
00106      */
00107      virtual bool isReleaseDetectionEnabled() = 0;
00108 };
00109 } // namespace aubo_scope
00110 } // namespace arcs
00111 #endif

```

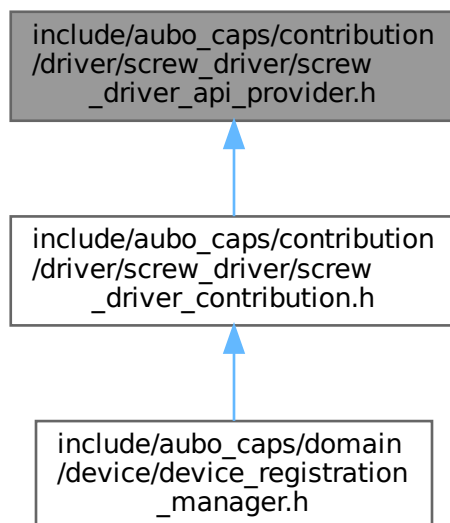
12.26 include/aubo_caps/contribution/driver/screw_driver/screw_driver_api_provider.h 文件参考

```
#include <aubo_caps/domain/system_api.h>
#include <aubo_caps/domain/io/io_model.h>
```

screw_driver_api_provider.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- `class arcs::aubo_scope::ScrewdriverApiProvider`

螺丝刀 API 提供者此接口提供对 `AuboScope` 中与螺丝刀贡献相关的功能和服务的访问。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`ScrewdriverApiProvider`)

12.27 screw_driver_api_provider.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_SCREW_DRIVER_API_PROVIDER_H
00002 #define AUBO_SCOPE_SCREW_DRIVER_API_PROVIDER_H
00003
00004 #include <aubo_caps/domain/system_api.h>
00005 #include <aubo_caps/domain/io/io_model.h>
00006
00007 namespace arcs {
00008     namespace aubo_scope {
00009         ARCS_CLASS_FORWARD(ScrewdriverApiProvider);
00010
00011         /**
00012          * \chinese
00013          * 螺丝刀 API 提供者
00014          * 此接口提供对 AuboScope 中与螺丝刀贡献相关的功能和服务的访问。
00015          * \endchinese
00016          * \english
00017          * ScrewdriverApiProvider
00018          * This interface provides access to functionality and services available from
00019          * within AuboScope relevant for screwdriver contributions.
00020          * \endenglish
00021          */
00022         class ScrewdriverApiProvider
00023         {
00024         public:
00025             virtual ~ScrewdriverApiProvider() = default;
00026
00027             /**
00028              * \chinese
00029              * @return 提供系统相关功能访问的接口
00030              * \endchinese
00031              * \english
00032              * @return an interface providing access to system related functionality.
00033              * \endenglish
00034              */
00035             virtual SystemApiPtr getSystemAPI() = 0;
00036
00037             /**
00038              * \chinese
00039              * @return I/O 模型接口 (例如用于获取在设置螺丝刀的自定义用户输入中显示的 I/O)
00040              * \endchinese
00041              * \english
00042              * @return an I/O model interface (e.g. for getting I/Os to display in the
00043              * custom user input for setting up the screwdriver)
00044              * \endenglish
00045              */
00046             virtual IoModelPtr getIOModel() = 0;
00047         };
00048     } // namespace aubo_scope
00049 } // namespace arcs
00050 #endif

```

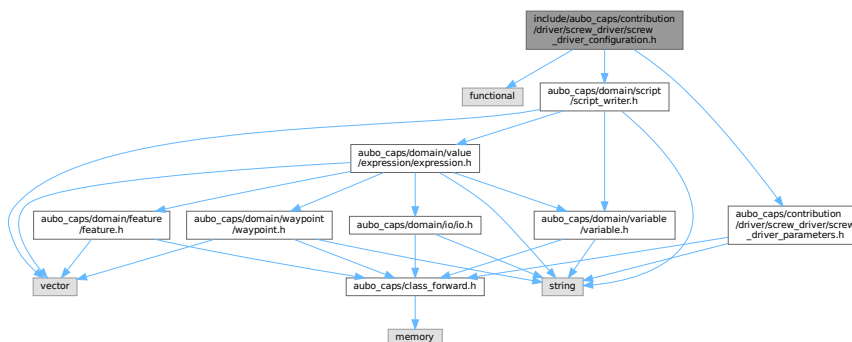
12.28 include/aubo_caps/contribution/driver/screw_driver/↔ screw_driver_configuration.h 文件参考

```

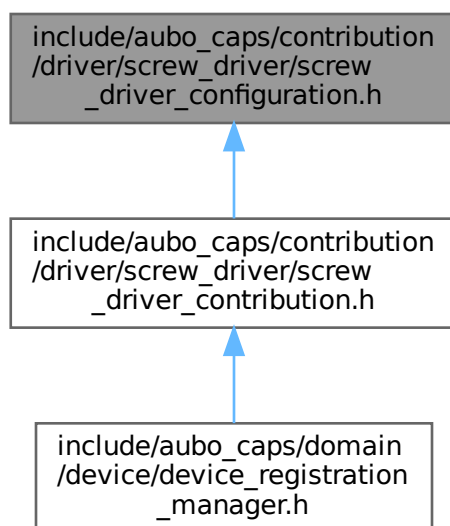
#include <functional>
#include <aubo_caps/domain/script/script_writer.h>
#include <aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h>

```

screw_driver_configuration.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::ProgramSelectionParameters`
- class `arcs::aubo_scope::PrepareToStartScrewdriverParameters`
- class `arcs::aubo_scope::FeedScrewParameters`
- class `arcs::aubo_scope::DriveScrewNotOKParameters`
- class `arcs::aubo_scope::DriveScrewOKParameters`
- class `arcs::aubo_scope::ScrewdriverReadyParameters`

This interface provides the parameters relevant for generating script for the Screwdriver Ready feedback operation capability (registered using `ScrewdriverFeedbackCapabilities#registerScrewdriverReadyCapability` (Script

- class `arcs::aubo_scope::ScrewdriverConfiguration`

This interface provides access to register or setup various properties and capabilities of a screwdriver.

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ProgramSelectionParameters)`
- `arcs::aubo_scope::ARCS_CLASS_FORWARD (PrepareToStartScrewdriverParameters)`
- `arcs::aubo_scope::ARCS_CLASS_FORWARD (FeedScrewParameters)`
- `arcs::aubo_scope::ARCS_CLASS_FORWARD (DriveScrewNotOKParameters)`
- `arcs::aubo_scope::ARCS_CLASS_FORWARD (DriveScrewOKParameters)`
- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ScrewdriverReadyParameters)`
- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ScrewdriverConfiguration)`

12.29 screw_driver_configuration.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_SCREW_DRIVER_CONFIGURATION_H
00002 #define AUBO_SCOPE_SCREW_DRIVER_CONFIGURATION_H
00003
00004 #include <functional>
00005 #include <aubo_caps/domain/script/script_writer.h>
00006 #include <aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010
00011 ARCS_CLASS_FORWARD(ProgramSelectionParameters);
00012 /**
00013  * <p>
00014  * This interface provides parameters relevant for generating script code for
00015  * the Program Selection capability (registered using {@link
00016  * ScrewdriverCapabilities#registerProgramSelectionCapability(ScrewdriverProgramListProvider,
00017  * ScriptCodeGenerator)}).
00018  * </p>
00019  *
00020  * These parameters are passed when AuboScope calls the implementation of the
00021  * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method
00022  * responsible for the generating the script code.
00023  */
00024 class ProgramSelectionParameters
00025 {
00026 public:
00027     virtual ~ProgramSelectionParameters() = default;
00028
00029     /**
00030      * @return The selected screwdriver program to use for the operation.
00031      * @throws UnsupportedOperationException if the screwdriver has not
00032      * registered this capability
00033      */
00034     virtual ScrewdriverProgram getScrewdriverProgram() = 0;
00035
00036     /**
00037      * @return The operation type selected by the end user.
00038      * @throws UnsupportedOperationException if the screwdriver has not
00039      * registered this capability
00040      */
00041     virtual ScrewdriverOperationType getScrewdriverOperationType() = 0;
00042 };
00043
00044 ARCS_CLASS_FORWARD(PrepareToStartScrewdriverParameters);
00045 /**
00046  * <p>
00047  * This interface provides parameters relevant for generating script code for
00048  * the Prepare To Start Screwdriver capability (registered using {@link
00049  * ScrewdriverCapabilities#registerPrepareToStartScrewdriverCapability(ScriptCodeGenerator)}).
00050  * </p>
00051  *
00052  * <p>
00053  * These parameters are passed when AuboScope calls the implementation of the
00054  * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method
00055  * responsible for the generating the script code.
00056  * </p>
00057  *
00058  * The interface is intentionally empty, but could have parameters in future
00059  * releases.
00060  */

```

```

00061 class PrepareToStartScrewdriverParameters
00062 {
00063 public:
00064     virtual ~PrepareToStartScrewdriverParameters() = default;
00065 };
00066
00067 ARCS_CLASS_FORWARD(FeedScrewParameters);
00068 /**
00069  * <p>
00070  * This interface provides parameters relevant for generating script code for
00071  * the Feed Screw capability (registered using {@link
00072  * ScrewdriverCapabilities#registerFeedScrewCapability(ScriptCodeGenerator)}).
00073  * </p>
00074  *
00075  * <p>
00076  * These parameters are passed when AuboScope calls the implementation of the
00077  * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method
00078  * responsible for the generating the script code.
00079  * </p>
00080  *
00081  * The interface is intentionally empty, but could have parameters in future
00082  * releases.
00083  */
00084 class FeedScrewParameters
00085 {
00086 public:
00087     virtual ~FeedScrewParameters() = default;
00088 };
00089
00090 ARCS_CLASS_FORWARD(DriveScrewNotOKParameters);
00091 /**
00092  * <p>
00093  * This interface provides parameters relevant for generating script code for
00094  * the Drive Screw Not OK feedback capability (registered using {@link
00095  * ScrewdriverFeedbackCapabilities#registerDriveScrewNotOKCapability(ScriptCodeGenerator)}).
00096  * </p>
00097  *
00098  * <p>
00099  * These parameters are passed when AuboScope calls the implementation of the
00100  * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method
00101  * responsible for the generating the script code.
00102  * </p>
00103  *
00104  * The interface is intentionally empty, but could have parameters in future
00105  * releases.
00106  */
00107 class DriveScrewNotOKParameters
00108 {
00109 public:
00110     virtual ~DriveScrewNotOKParameters() = default;
00111 };
00112
00113 ARCS_CLASS_FORWARD(DriveScrewOKParameters);
00114 /**
00115  * <p>
00116  * This interface provides parameters relevant for generating script code for
00117  * the Drive Screw OK feedback capability (registered using {@link
00118  * ScrewdriverFeedbackCapabilities#registerDriveScrewOKCapability(ScriptCodeGenerator)}).
00119  * </p>
00120  *
00121  * <p>
00122  * These parameters are passed when AuboScope calls the implementation of the
00123  * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method
00124  * responsible for the generating the script code.
00125  * </p>
00126  *
00127  * The interface is intentionally empty, but could have parameters in future
00128  * releases.
00129  */
00130 class DriveScrewOKParameters
00131 {
00132 public:
00133     virtual ~DriveScrewOKParameters() = default;
00134 };
00135
00136 ARCS_CLASS_FORWARD(ScrewdriverReadyParameters);
00137 /**
00138  * This interface provides the parameters relevant for generating script for the
00139  * Screwdriver Ready feedback operation capability (registered using {@link
00140  * ScrewdriverFeedbackCapabilities#registerScrewdriverReadyCapability(ScriptCodeGenerator)}).
00141  *
00142  * <p>
00143  * These parameters are passed when AuboScope calls the implementation of the
00144  * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method
00145  * responsible for the generating the script code.
00146  * </p>
00147  *

```

```

00148 * The interface is intentionally empty, but could have parameters in future
00149 * releases.
00150 */
00151 class ScrewdriverReadyParameters
00152 {
00153 public:
00154     virtual ~ScrewdriverReadyParameters() = default;
00155 };
00156
00157 ARCS_CLASS_FORWARD(ScrewdriverConfiguration);
00158 /**
00159 * This interface provides access to register or setup various properties and
00160 * capabilities of a screwdriver. It is optional to register these capabilities
00161 * and properties.
00162 */
00163 class ScrewdriverConfiguration
00164 {
00165 public:
00166     virtual ~ScrewdriverConfiguration() = default;
00167
00168     /**
00169      * Add a screwdriver program to the list of programs.
00170      *
00171      * @param screwdriverProgram program to add, not <code>null</code>
00172      * @throws DuplicateScrewdriverProgramID if the ids of the screwdriver
00173      * programs (returned by the method {@link ScrewdriverProgram#getId()})
00174      * are not unique.
00175      * @throws NameOfScrewdriverProgramCannotBeNullOrEmpty if the display name
00176      * of a program (returned by the method {@link
00177      * ScrewdriverProgram#getDisplayName()}) is <code>null</code> or an empty
00178      * string.
00179      */
00180     virtual void addScrewdriverProgram(
00181         ScrewdriverProgramPtr screwdriverProgram) = 0;
00182
00183     /**
00184      * Add a list of screwdriver programs to the list of programs.
00185      *
00186      * @param screwdriverProgramList list of programs to add, not
00187      * <code>null</code>
00188      * @throws DuplicateScrewdriverProgramID if the ids of the screwdriver
00189      * programs (returned by the method
00190      * {@link ScrewdriverProgram#getId()})
00191      * are not unique.
00192      * @throws NameOfScrewdriverProgramCannotBeNullOrEmpty if the display name
00193      * of a program (returned by the method
00194      * {@link
00195      * ScrewdriverProgram#getDisplayName()}) is <code>null</code> or an empty
00196      * string.
00197      */
00198     virtual void addScrewdriverPrograms(
00199         std::vector<ScrewdriverProgramPtr> screwdriverProgramList) = 0;
00200
00201     /**
00202      * <p>
00203      * Register a program selection capability for a screwdriver that supports
00204      * selecting between a set of screwdriver programs, typically defined on an
00205      * external control box for the screwdriver.
00206      * </p>
00207      *
00208      * <p>
00209      * The provided implementation of the {@link ScriptCodeGenerator} interface
00210      * must generate the script code for selecting the specified screwdriver
00211      * program. The return value of the script code must be a boolean, i.e.
00212      * 'True' or 'False'. 'True' must be returned if selecting the program
00213      * succeeded and 'False' must be returned if the program selection failed.
00214      * </p>
00215      *
00216      * The generated script code is executed before the script code for starting
00217      * the screwdriver (could be from a different node), i.e the script code
00218      * generated by
00219      * {@link
00220      * ScrewdriverContribution#generateStartScrewdriverScript(ScriptWriter,
00221      * ScrewdriverParameters)}).
00222      *
00223      * @param programListProvider provider of the list of screwdriver programs,
00224      * not <code>null</code>
00225      * @param scriptCodeGenerator script code generator responsible for
00226      * generating the script code for selecting the specified screwdriver
00227      * program. When the script code needs to be generated, the
00228      * {@link
00229      * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method will be
00230      * called (by AutoScope). Return value for the generated script code must be
00231      * a boolean, i.e. 'True' or 'False'.
00232      * @throws CapabilityAlreadyRegistered if this capability has already been
00233      * registered
00234      * @throws CalledOutsideScrewdriverConfigurationPhase if this method is

```

```

00235     * called at the wrong time, i.e. outside the scope of the {@link
00236     * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00237     * ScrewdriverApiProvider)}} method.
00238     */
00239 virtual void registerProgramSelectionCapability(
00240     std::function<void(ScriptWriterPtr, ProgramSelectionParametersPtr)>
00241     scriptCodeGenerator) = 0;
00242
00243 /**
00244     * Register an operation type capability for a screwdriver that
00245     * supports/requires the parameter for the screwdriving operation type (i.e.
00246     * if the operation is Tighten or Loosen).
00247     *
00248     * @throws CapabilityAlreadyRegistered if this capability has already been
00249     * registered
00250     * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00251     * called at the wrong time, i.e. outside the scope of the {@link
00252     * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00253     * ScrewdriverApiProvider)}} method.
00254     */
00255 virtual void registerOperationTypeCapability() = 0;
00256
00257 /**
00258     * <p>
00259     * Register a capability/requirement for a screwdriver that needs to
00260     * generate script code for preparing the screwdriver for a screwdriving
00261     * operation (before starting the screw driver).
00262     * </p>
00263     *
00264     * <p>
00265     * The provided implementation of the {@link ScriptCodeGenerator} interface
00266     * must generate the script code for preparing the screwdriver. The return
00267     * value of the script code must be a boolean, i.e. 'True' or 'False'.
00268     * 'True' must be returned if the preparation of the screwdriver was
00269     * successful and 'False' must be returned if the preparation failed.
00270     * </p>
00271     *
00272     * The generated script code is executed before the script code for starting
00273     * the screwdriver (could be from a different node), i.e the script code
00274     * generated by
00275     * {@link
00276     * ScrewdriverContribution#generateStartScrewdriverScript(ScriptWriter,
00277     * ScrewdriverParameters)}}.
00278     *
00279     * @param scriptCodeGenerator script code generator responsible for
00280     * generating the script code for preparing the screwdriver. When the script
00281     * code needs to be generated, the
00282     *
00283     * {@link
00284     * ScriptCodeGenerator#generateScript(ScriptWriter, Object)}} method will be
00285     * called (by AuboScope). Return value for the generated script code must be
00286     * a boolean, i.e. 'True' or 'False'.
00287     * @throws CapabilityAlreadyRegistered if this capability has already been
00288     * registered
00289     * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00290     * called at the wrong time, i.e. outside the scope of the {@link
00291     * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00292     * ScrewdriverApiProvider)}} method.
00293     */
00294 virtual void registerPrepareToStartScrewdriverCapability(
00295     std::function<void(ScriptWriterPtr,
00296         PrepareToStartScrewdriverParametersPtr)>
00297     scriptCodeGenerator) = 0;
00298
00299 /**
00300     * <p>
00301     * Register a capability for a screwdriver that supports automatic screw
00302     * feeding.
00303     * </p>
00304     *
00305     * <p>
00306     * The provided implementation of the {@link ScriptCodeGenerator} interface
00307     * must generate the script code for feeding a screw to the screwdriver. The
00308     * return value of the script code must be a boolean, i.e. 'True' or
00309     * 'False'. 'True' must be returned if the screw feed operation was
00310     * successful and 'False' must be returned if the operation failed.
00311     * </p>
00312     *
00313     * The generated script code is executed before the script code for
00314     * preparing the screwdriver, i.e the script code generated by {@link
00315     * #registerPrepareToStartScrewdriverCapability(ScriptCodeGenerator)}}.
00316     *
00317     * @param scriptCodeGenerator script code generator responsible for
00318     * generating the script code for feeding a screw to the screwdriver. When
00319     * the script code needs to be generated, the
00320     *
00321     * {@link
00322     * ScriptCodeGenerator#generateScript(ScriptWriter, Object)}} method will be
00323     * called (by AuboScope). Return value for the generated script code must be

```

```

00322     * a boolean, i.e. 'True' or 'False'
00323     * @throws CapabilityAlreadyRegistered if this capability has already been
00324     * registered
00325     * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00326     * called at the wrong time, i.e. outside the scope of the {@link
00327     * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00328     * ScrewdriverApiProvider)} method.
00329     */
00330     virtual void registerFeedScrewCapability(
00331         std::function<void(ScriptWriterPtr, FeedScrewParametersPtr)>
00332         scriptCodeGenerator) = 0;
00333
00334     /**
00335     * <p>
00336     * Register a feedback capability for a screwdriver that is capable of
00337     * providing information about whether or not an ongoing screwdriving
00338     * operation (driving the screw) ended successfully (ended in OK-state).
00339     * </p>
00340     *
00341     * <p>
00342     * <b>Note:</b> This capability must only be used to provide feedback about
00343     * the success of the screwing operation. It should not be used to provide
00344     * feedback about failures. To support feedback about when the operation
00345     * fails, register the Drive Screw Not Ok feedback capability the using the
00346     * {@link #registerDriveScrewNotOkCapability(ScriptCodeGenerator)} method.
00347     * </p>
00348     *
00349     * <p>
00350     * The provided implementation of the {@link ScriptCodeGenerator} interface
00351     * must generate the script code for determining if the screwing operation
00352     * succeeded. The return value of the script code must be a boolean, i.e.
00353     * 'True' or 'False'. 'True' must be returned if the screwing operation was
00354     * successful, otherwise 'False' must be returned.
00355     * </p>
00356     *
00357     * <p>
00358     * <b>Note:</b> A return value of 'False' must not be used to indicate that
00359     * the operation failed, but simply that a success was not (yet) detected.
00360     * If the screwdriver can provide feedback on whether or not the screwing
00361     * operation failed, the Drive Screw Not Ok feedback capability must be
00362     * used.
00363     * </p>
00364     *
00365     * <b>Example:</b> <br>
00366     * The digital tool input 0 will go high if the operation ended in OK-state.
00367     * The script code is: <pre> return get_tool_digital_in(0)
00368     * </pre>
00369     *
00370     * So the implementation of {@link
00371     * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} could be: <pre>
00372     * scriptWriter.appendLine("return get_tool_digital_in(0)");
00373     * </pre>
00374     *
00375     * @param scriptCodeGenerator script code generator responsible for
00376     * generating the script code for determining if the screwing operation
00377     * succeeded. When the script code needs to be generated, the
00378     *      {@link
00379     *      ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method will be
00380     *      called (by AutoScope). Return value for the
00381     *      generated script code must be a boolean, i.e. 'True' or 'False'.
00382     * @throws CapabilityAlreadyRegistered if this capability has already been
00383     * registered
00384     * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00385     * called at the wrong time, i.e. outside the scope of the {@link
00386     * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00387     * ScrewdriverApiProvider)} method.
00388     */
00389     virtual void registerDriveScrewOKCapability(
00390         std::function<void(ScriptWriterPtr, DriveScrewOKParametersPtr)>
00391         scriptCodeGenerator) = 0;
00392
00393     /**
00394     * <p>
00395     * Register a feedback capability for a screwdriver that is capable of
00396     * providing information about whether or not an ongoing screwdriving
00397     * operation (driving the screw) failed, i.e. ended in Not Ok (NOK) state.
00398     * </p>
00399     *
00400     * <p>
00401     * <b>Note:</b> This capability must only be used to provide feedback about
00402     * the failure of the screwing operation. It should not be used to provide
00403     * feedback about successes. To support feedback about when the operation
00404     * succeeds, register the Drive Screw Ok feedback capability the using the
00405     * {@link #registerDriveScrewOKCapability(ScriptCodeGenerator)} method.
00406     * </p>
00407     *
00408     * <p>

```

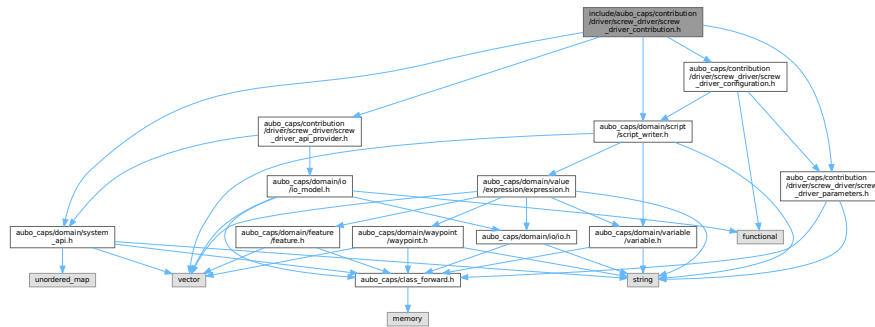
```

00409      * The provided implementation of the {@link ScriptCodeGenerator} interface
00410      * must generate the script code for determining if the screwing operation
00411      * ended in failure. The return value of the script code must be a boolean,
00412      * i.e. 'True' or 'False'. 'True' must be returned if the screwing operation
00413      * failed, otherwise 'False' must be returned.
00414      * </p>
00415      *
00416      * <p>
00417      * <b>Note:</b> A return value of 'False' must not be used to indicate that
00418      * the operation succeeded, but simply that a failure was not (yet)
00419      * detected. If the screwdriver can provide feedback on whether or not the
00420      * screwing operation succeeded, the Drive Screw Ok feedback capability
00421      * should be used.
00422      * </p>
00423      *
00424      * <b>Example:</b> <br>
00425      * The digital tool input 1 will go high if the operation ended in NOT
00426      * OK-state. The script code is: <pre> return get_tool_digital_in(1)
00427      * </pre>
00428      * So the implementation of {@link
00429      * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} could be: <pre>
00430      *   scriptWriter.appendLine("return get_tool_digital_in(1)");
00431      * </pre>
00432      *
00433      *
00434      * @param scriptCodeGenerator script code generator responsible for
00435      * generating the script code for determining if the screwing operation
00436      * failed. When the script code needs to be generated, the {@link
00437      * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method will be
00438      * called (by AutoScope). Return value for the
00439      * generated script code must be a boolean, i.e. 'True' or 'False'.
00440      * @throws CapabilityAlreadyRegistered if this capability has already been
00441      * registered
00442      * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00443      * called at the wrong time, i.e. outside the scope of the {@link
00444      * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00445      * ScrewdriverApiProvider)} method.
00446      */
00447 virtual void registerDriveScrewNotOKCapability(
00448     std::function<void(ScriptWriterPtr, DriveScrewNotOKParametersPtr)>
00449     scriptCodeGenerator) = 0;
00450
00451 /**
00452  * <p>
00453  * Register a feedback capability for a screwdriver that is capable of
00454  * providing information about whether or not the screwdriver device is
00455  * ready to operate (receive commands). This capability is used (by
00456  * AutoScope) in a program to ensure that the screwdriver is ready before
00457  * sending commands to it.
00458  * </p>
00459  *
00460  * <p>
00461  * The provided implementation of the {@link ScriptCodeGenerator} interface
00462  * must generate the script code for determining if the screwdriver is
00463  * ready. The return value of the script code must be a boolean, i.e. 'True'
00464  * or 'False'. 'True' must be returned if the screwdriver is ready,
00465  * otherwise 'False' must be returned.
00466  * </p>
00467  *
00468  * The generated script code is executed before the script code for the
00469  * program selected capability (if that capability has been registered).
00470  *
00471  * @param scriptCodeGenerator script code generator responsible for
00472  * generating the script code for determining if the screwdriver is ready.
00473  * When the script code needs to be generated, the
00474  * {@link
00475  * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method will be
00476  * called (by AutoScope). Return value for the
00477  * generated script code must be a boolean, i.e. 'True' or 'False'.
00478  * @throws CapabilityAlreadyRegistered if this capability has already been
00479  * registered
00480  * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00481  * called at the wrong time, i.e. outside the scope of the {@link
00482  * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00483  * ScrewdriverApiProvider)} method.
00484  */
00485 virtual void registerScrewdriverReadyCapability(
00486     std::function<void(ScriptWriterPtr, ScrewdriverReadyParametersPtr)>
00487     scriptCodeGenerator) = 0;
00488 };
00489 } // namespace aubo_scope
00490 } // namespace arcs
00491
00492 #endif

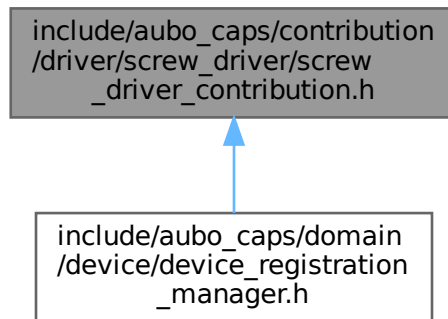
```

12.30 include/aubo_caps/contribution/driver/screw_driver/↵ screw_driver_contribution.h 文件参考

```
#include <aubo_caps/domain/system_api.h>
#include <aubo_caps/domain/script/script_writer.h>
#include <aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h>
#include <aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h>
#include <aubo_caps/contribution/driver/screw_driver/screw_driver_api_provider.h>
screw_driver_contribution.h 的引用 (Include) 关系图:
```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ScrewdriverContribution](#)
螺丝刀贡献此接口定义了向 *AuboScope* 添加螺丝刀功能的 *API*。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ScrewdriverContribution](#))

12.31 screw_driver_contribution.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_SCREW_DRIVER_CONTRIBUTION_H
00002 #define AUBO_SCOPE_SCREW_DRIVER_CONTRIBUTION_H
00003
00004 #include <aubo_caps/domain/system_api.h>
00005 #include <aubo_caps/domain/script/script_writer.h>
00006 #include <aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h>
00007 #include <aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h>
00008 #include <aubo_caps/contribution/driver/screw_driver/screw_driver_api_provider.h>
00009
00010 namespace arcs {
00011 namespace aubo_scope {
00012 ARCS_CLASS_FORWARD(ScrewdriverContribution);
00013
00014 /**
00015  * \chinese
00016  * 螺丝刀贡献
00017  * 此接口定义了向 AuboScope 添加螺丝刀功能的 API。
00018  *
00019  * 实现此接口将把螺丝刀贡献"集成"(挂钩)到内置(原生)AuboScope 螺丝刀程序节点
00020  * 和螺丝刀安装画面中。这将向终端用户暴露螺丝刀支持的螺丝驱动功能。
00021  *
00022  * <b>注意:</b> 内置(原生)AuboScope 螺丝刀程序节点和螺丝刀贡献在 CB3 机器人上
00023  * 均不受支持。如果 AuboCap 在 CB3 机器人上注册了螺丝刀贡献, AuboCap 将无法激活。
00024  *
00025  * 螺丝刀程序节点允许终端用户编程螺丝驱动操作。终端用户可用的参数基于螺丝刀支持的
00026  * 属性和功能。至少,所有螺丝刀必须支持"默认"拧紧(启动螺丝刀)和停止(停止螺丝刀)
00027  * 操作。所有其他功能是可选的,可以在调用
00028  * {@link ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00029  * ScrewdriverApiProvider)} 方法时注册。
00030  *
00031  * 螺丝刀安装画面中的集成允许用户选择螺丝刀来进行拧紧操作编程(使用螺丝刀程序节点)。
00032  * 它还提供了定义终端用户所需的任何自定义输入来设置螺丝刀的可能性,例如指定 IP 地址。
00033  * 此功能是可选的,因为此类配置属性是特定于每个螺丝刀设备的。相应的生成自定义 UI 将在
00034  * 用户选择螺丝刀时在画面中可访问,并替换默认的螺丝刀安装 UI。当调用
00035  * {@link ScrewdriverContribution#configureInstallation(CustomUserInputConfiguration,
00036  * SystemConfiguration, TCPConfiguration, ScrewdriverApiProvider)} 方法时,提供创建
00037  * 基于用户输入的自定义配置的访问权限。
00038  * \endchinese
00039  * \english
00040  * ScrewdriverContribution
00041  * This interface defines an API for adding functionality for screwdrivers to
00042  * AuboScope.
00043  *
00044  * Implementing this interface will "integrate" (hook) this screwdriving
00045  * contribution into the builtin (native) AuboScope Screwdriving program node
00046  * and Screwdriving installation screen. This will expose the screwdriving
00047  * functionality supported by the screwdriver to the end user.
00048  *
00049  * <b>Note:</b> Both the builtin (native) AuboScope Screwdriving program node
00050  * and Screwdriver contributions are not supported on CB3 robots. If a AuboCap
00051  * registers a Screwdriver contribution on a CB3 robot, the AuboCap will fail to
00052  * be activated.
00053  *
00054  * The Screwdriving program node allows the end user to program screwdriving
00055  * operations. The configuration options available to the end user are based on
00056  * the properties and capabilities supported by the screwdriver. As a minimum,
00057  * all screwdrivers must support "default" screw (start screwdriver) and stop
00058  * (stop screwdriver) operations. All other capabilities are optional to support
00059  * and can be registered when the method
00060  * {@link ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00061  * ScrewdriverApiProvider)} is called.
00062  *
00063  * The integration in the Screwdriving installation screen allows the user to
00064  * select the screwdriver for programming screwing operations (using the
00065  * Screwdriving program node). It also offers the possibility to define any
00066  * custom inputs from the end user that are required to setup the screwdriver,
00067  * e.g. specifying an IP address. This functionality is optional, since such
00068  * configuration properties are specific for each screwdriver device. The
00069  * corresponding generated custom UI will be accessible in the screen when the
00070  * screwdriver is selected (by the end user) and replace the default
00071  * Screwdriving installation UI. Access to create a custom configuration based
00072  * on user inputs is provided when the method
00073  * {@link
00074  * ScrewdriverContribution#configureInstallation(CustomUserInputConfiguration,
00075  * SystemConfiguration, TCPConfiguration, ScrewdriverApiProvider)} is called.
00076  * \endenglish
00077  */
00078 class ScrewdriverContribution
00079 {
00080 public:
00081     virtual ~ScrewdriverContribution() = default;
00082
00083     /**

```



```

00084     * \chinese
00085     * 此方法必须返回螺丝刀贡献的标题。标题显示在以下位置：
00086     * <ul>
00087     * <li> 内置 AuboScope 螺丝驱动安装画面顶部选择螺丝刀的位置 </li>
00088     * <li> 内置 AuboScope 螺丝驱动程序节点的画面（如果已在螺丝驱动安装画面中选择了
00089     * 此螺丝刀）</li>
00090     * </ul>
00091     * 此方法在启动时调用一次。
00092     *
00093     * @param locale AuboScope 的当前语言环境，可用于支持多种语言的标题
00094     * @return 此螺丝刀贡献的标题，不能为 <code>null</code> 也不能为空字符串
00095     * \endchinese
00096     * \english
00097     * This method must return the title of the screwdriver contribution. The
00098     * title is displayed in: <ul> <li>The builtin (native) AuboScope
00099     * Screwdriing installation screen at the top where the screwdriver is
00100     * selected
00101     * </li>
00102     * <li>The screen for the builtin (native) AuboScope Screwdriing
00103     * program node (if this screwdriver has been selected in the Screwdriing
00104     * installation screen)
00105     * </li>
00106     * </ul>
00107     *
00108     * This method is called once upon startup.
00109     *
00110     * @param locale the current locale of AuboScope. Can be used for supporting
00111     * titles in several languages.
00112     * @return the title of this screw driver contribution, not
00113     * <code>null</code> nor an empty string.
00114     * \endenglish
00115     */
00116 virtual std::string getTitle(/*Locale locale*/) = 0;
00117
00118 /**
00119     * \chinese
00120     * <p> 当调用此方法时，使用 configuration 参数注册或设置螺丝刀的可选属性和功能。</p>
00121     * <p> 此方法在螺丝刀配置阶段调用，在贡献注册后、新安装被加载或创建时调用。</p>
00122     * <b> 注意：</b> 如果螺丝刀仅支持基本的"默认"拧紧和停止操作，请将此方法的实现留空。
00123     *
00124     * @param screwdriverConfiguration 可用于注册或设置螺丝刀属性和功能的配置实例
00125     * @param apiProvider 提供对 AuboScope 中可用于设置螺丝刀功能的相关功能和服务访问
00126     * \endchinese
00127     * \english
00128     * <p>
00129     * When this method is called, use the configuration parameter to register
00130     * or setup optional properties and capabilities of the screwdriver.
00131     * </p>
00132     * ...
00133     * @param screwdriverConfiguration a configuration instance that can be used
00134     * for registering or setting up the properties and capabilities of the
00135     * screwdriver.
00136     * @param apiProvider provides access to functionality and services
00137     * available from within AuboScope which can be relevant for setting up the
00138     * screwdriver capabilities (e.g. the {@link SystemAPI} interface with
00139     * functionality for querying information about the robot).
00140     * \endenglish
00141     */
00142 virtual void configureScrewdriver(
00143     ScrewdriverConfiguration screwdriverConfiguration,
00144     ScrewdriverAPIProviderPtr apiProvider);
00145
00146 /**
00147     * \chinese
00148     * <p> 当调用此方法时，必须生成驱动螺丝的脚本代码。</p>
00149     * 终端用户定义/配置的已注册可选螺丝刀功能/要求的参数将作为脚本代码生成的输入。
00150     *
00151     * @param scriptWriter 使用此脚本编写器实例生成驱动螺丝的脚本代码
00152     * @param parameters 终端用户定义/配置的螺丝驱动操作参数
00153     * \endchinese
00154     * \english
00155     * <p>
00156     * When this method is called, the script code for driving the screw must be
00157     * generated.
00158     * </p>
00159     * The relevant parameters for the registered optional screwdriver
00160     * capabilities/requirements defined/configured by the end user are provided
00161     * as input to the script code generation.
00162     *
00163     * @param scriptWriter use this script writer instance to generate the
00164     * script code for driving the screw
00165     * @param parameters the parameters for the screwdriing operation
00166     * defined/configured by the end user
00167     * \endenglish
00168     */
00169 virtual void generateStartScrewdriverScript(
00170     ScriptWriterPtr scriptWriter, ScrewdriverParametersPtr parameters) = 0;

```

```

00171
00172  /**
00173   * \chinese
00174   * <p> 当调用此方法时，必须生成停止螺丝刀脚本代码（即螺丝刀必须停止当前的
00175   * 螺丝驱动操作）。</p>
00176   * 终端用户定义/配置的已注册可选螺丝刀功能/要求的参数将作为脚本代码生成的输入。
00177   *
00178   * @param scriptWriter 使用此脚本编写器实例生成停止螺丝刀的脚本代码
00179   * @param parameters 终端用户定义/配置的螺丝驱动操作参数
00180   * \endchinese
00181   * \english
00182   * <p>
00183   * When this method is called, the script code for stopping the screw driver
00184   * must be generated (i.e. the screwdriver must stop the current
00185   * screwdriving operation).
00186   * </p>
00187   * The relevant parameters for the registered optional screwdriver
00188   * capabilities/requirements defined/configured by the end user are provided
00189   * as input to the script code generation.
00190   *
00191   * @param scriptWriter use this script writer instance to generate the
00192   * script code for stopping the screwdriver
00193   * @param parameters the parameters for the screwdriving operation
00194   * defined/configured by the end user
00195   * \endenglish
00196   */
00197  virtual void generateStopScrewdriverScript(
00198      ScriptWriterPtr scriptWriter, ScrewdriverParametersPtr parameters) = 0;
00199  ;
00200 };
00201 } // namespace aubo_scope
00202 } // namespace arcs
00203
00204 #endif

```

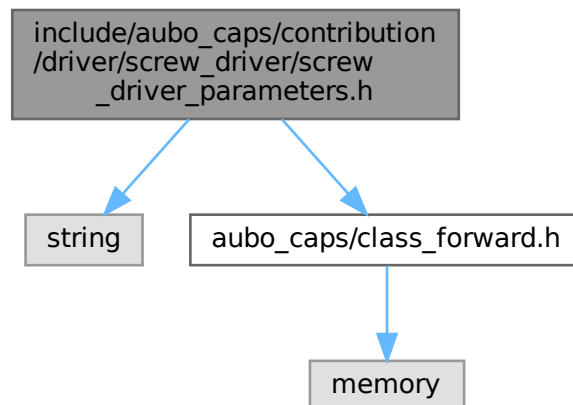
12.32 include/aubo_caps/contribution/driver/screw_driver/↵ screw_driver_parameters.h 文件参考

```

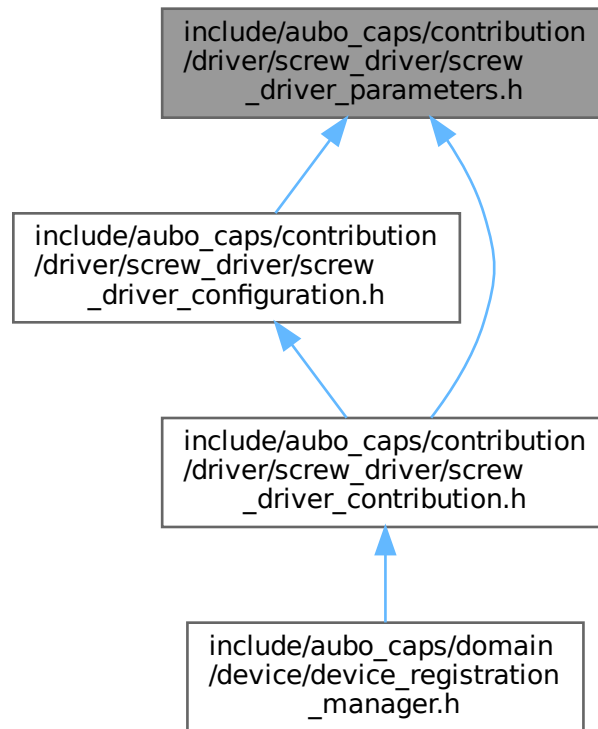
#include <string>
#include <aubo_caps/class_forward.h>

```

screw_driver_parameters.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ScrewdriverProgram](#)
This abstract class represents a screwdriver program, typically defined on an external control box for the screwdriver.
- class [arcs::aubo_scope::ScrewdriverParameters](#)
This interface represents parameters for a screwdriver operation defined/configured by the end user.

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

枚举

- enum [arcs::aubo_scope::ScrewdriverOperationType](#) { [arcs::aubo_scope::TIGHTEN](#), [arcs::aubo_scope::LOOSEN](#) }
- This interface represents the type of screwing operation performed by the screwdriver.*

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (ScrewdriverProgram)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (ScrewdriverParameters)

12.33 screw_driver_parameters.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_SCREW_DIRVER_PARAMETERS_H
00002 #define AUBO_SCOPE_SCREW_DIRVER_PARAMETERS_H
00003
00004 #include <string>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009
00010 ARCS_CLASS_FORWARD(ScrewdriverProgram);
00011 /**
00012  * This abstract class represents a screwdriver program, typically defined on an
00013  * external control box for the screwdriver.
00014  */
00015 class ScrewdriverProgram
00016 {
00017 public:
00018     virtual ~ScrewdriverProgram() = 0;
00019
00020     /**
00021      * <p>
00022      * This method must return an identifier to uniquely identify this
00023      * screwdriver program. The identifier will be used for persistence and
00024      * resolving the correct screwdriver program. Therefore the identifier must
00025      * remain constant over time and versions of the screwdriver AuboCap.
00026      * </p>
00027      *
00028      * This method must be implemented.
00029      *
00030      * @return a unique identifier string which will remain constant over time
00031      * and versions of the screwdriver AuboCap.
00032      */
00033     virtual std::string getId() = 0;
00034
00035     /**
00036      * <p>
00037      * This method should return the name of the screwdriver program. The name
00038      * will be displayed in the UI, e.g. in a combo box. The name returned is
00039      * allowed to change over versions of the driver AuboCap as well as be
00040      * different depending on the current language selected in AuboScope, in
00041      * case translations are supported.
00042      * </p>
00043      *
00044      * This method must be implemented.
00045      *
00046      * @return the name of the screwdriver program to display in the UI.
00047      */
00048     virtual std::string getDisplayName() = 0;
00049 };
00050
00051 /**
00052  * This interface represents the type of screwing operation performed by the
00053  * screwdriver.
00054  */
00055 enum ScrewdriverOperationType
00056 {
00057
00058     /**
00059      * The screwdriver operation will follow the screw in a tightening direction
00060      * (in).
00061      */
00062     TIGHTEN,
00063
00064     /**
00065      * The screwdriver operation will follow the screw in a loosening direction
00066      * (out).
00067      */
00068     LOOSEN
00069 };
00070
00071 ARCS_CLASS_FORWARD(ScrewdriverParameters);
00072
00073 /**
00074  * This interface represents parameters for a screwdriver operation
00075  * defined/configured by the end user.
00076  */
00077 class ScrewdriverParameters
00078 {
00079 public:
00080     virtual ~ScrewdriverParameters() = default;
00081
00082     /**
00083      * <p>

```

```

00084      * Get the selected screwdriver program to be used for the screwdriver
00085      * operation.
00086      * </p>
00087      *
00088      * This is only valid, if the program selection capability has been
00089      * registered (using the method
00090      * {@link
00091      * ScrewdriverCapabilities#registerProgramSelectionCapability(ScrewdriverProgramListProvider,
00092      * ScriptCodeGenerator)}}).
00093      *
00094      * @return The selected screwdriver program to use for the operation.
00095      * @throws UnsupportedOperationException if the screwdriver has not
00096      * registered the program selection capability
00097      */
00098      virtual ScrewdriverProgram getScrewdriverProgram() = 0;
00099
00100      /**
00101      * <p>
00102      * Get the selected screwdriver operation type.
00103      * </p>
00104      *
00105      * This is only valid, if the operation type capability has been registered
00106      * (using the method
00107      * {@link ScrewdriverCapabilities#registerOperationTypeCapability()}).
00108      *
00109      * @return The operation type selected by the user.
00110      * @throws UnsupportedOperationException if the screwdriver has not
00111      * registered the operation type capability
00112      */
00113      virtual ScrewdriverOperationType getScrewdriverOperationType() = 0;
00114      ;
00115 };
00116 } // namespace aubo_scope
00117 } // namespace arcs
00118
00119 #endif

```

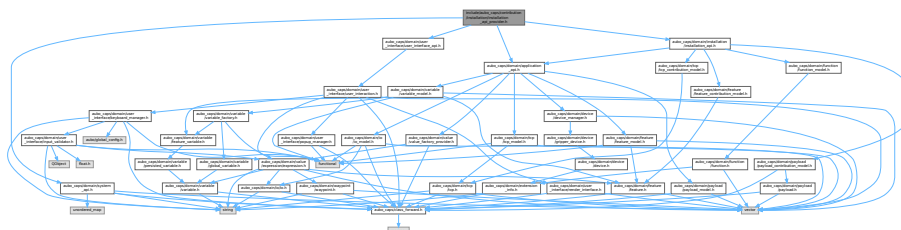
12.34 include/auco_caps/contribution/installation/↵ installation_api_provider.h 文件参考

```

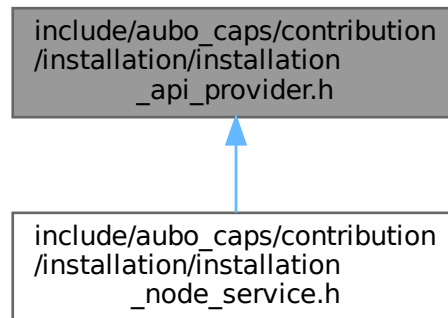
#include <auco_caps/domain/application_api.h>
#include <auco_caps/domain/system_api.h>
#include <auco_caps/domain/user_interface/user_interface_api.h>
#include <auco_caps/domain/installation/installation_api.h>

```

installation_api_provider.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- `class arcs::aubo_scope::InstallationApiProvider`

安装 API 提供者提供与安装节点相关的功能和服务访问。这包括 *AuboScope* 内部已有的功能, 以及在 *AuboScope* 中创建要使用的额外模型元素。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (InstallationApiProvider)`

12.35 installation_api_provider.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_INSTALLATION_API_PROVIDER_H
00002 #define AUBO_SCOPE_INSTALLATION_API_PROVIDER_H
00003
00004 #include <aubo_caps/domain/application_api.h>
00005 #include <aubo_caps/domain/system_api.h>
00006 #include <aubo_caps/domain/user_interface/user_interface_api.h>
00007 #include <aubo_caps/domain/installation/installation_api.h>
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011 ARCS_CLASS_FORWARD (InstallationApiProvider);
00012
00013 /**
00014  * \chinese
00015  * 安装 API 提供者
00016  * 提供与安装节点相关的功能和服务访问。这包括 AuboScope 内部已有的功能, 以及在
00017  * AuboScope 中创建要使用的额外模型元素。
00018  * \endchinese
00019  * \english
00020  * InstallationApiProvider
00021  * Provides access to functionality and services relevant for installation
00022  * nodes. This includes what is available from within AuboScope, as well as
00023  * creating additional model elements to be used within AuboScope.
00024  * \endenglish
00025  */
00026 class InstallationApiProvider
00027 {
00028 public:
00029     InstallationApiProvider (InstallationApiProvider &f);
00030     InstallationApiProvider (InstallationApiProvider &&f);
  
```

```

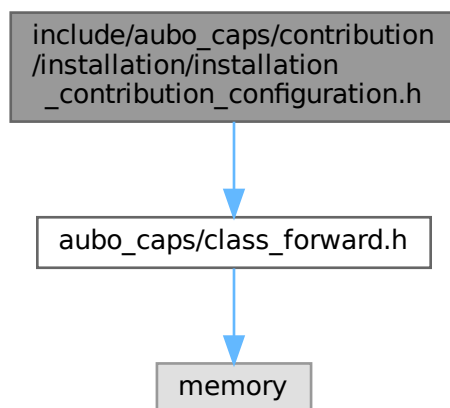
00031     virtual ~InstallationApiProvider();
00032
00033     /**
00034      * \chinese
00035      * 提供对系统相关功能的访问
00036      *
00037      * @return SystemAPI 实例
00038      * \endchinese
00039      * \english
00040      * Provides access to system related functionality
00041      *
00042      * @return an instance of SystemAPI
00043      * \endenglish
00044      */
00045     SystemApiPtr getSystemApi();
00046
00047     /**
00048      * \chinese
00049      * 提供对用户界面和终端用户交互相关功能的访问
00050      *
00051      * @return UserInterfaceAPI 实例
00052      * \endchinese
00053      * \english
00054      * Provides access to functionality related to user interface and end user
00055      * interaction
00056      *
00057      * @return an instance of UserInterfaceAPI
00058      * \endenglish
00059      */
00060     UserInterfaceApiPtr getUserInterfaceApi();
00061
00062     /**
00063      * \chinese
00064      * 提供对安装相关功能的访问
00065      *
00066      * @return InstallationAPI 实例
00067      * \endchinese
00068      * \english
00069      * Provides access to functionality relevant related to the installation
00070      *
00071      * @return an instance of InstallationAPI
00072      * \endenglish
00073      */
00074     InstallationApiPtr getInstallationApi();
00075
00076 private:
00077     friend class DataSwitch;
00078     InstallationApiProvider();
00079     void *d_{ nullptr };
00080 };
00081
00082 } // namespace aubo_scope
00083 } // namespace arcs
00084
00085 #endif // INSTALLATIONAPIPROVIDER_H

```

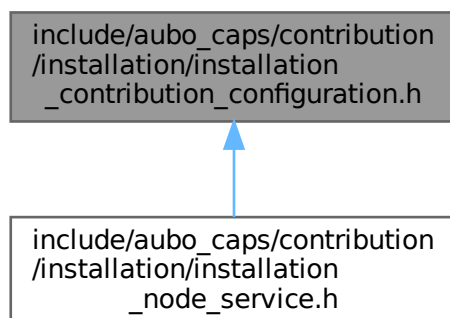
12.36 include/aubo_caps/contribution/installation/↵ installation_contribution_configuration.h 文件参考

```
#include <aubo_caps/class_forward.h>
```

installation_contribution_configuration.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::InstallationContributionConfiguration`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (InstallationContributionConfiguration)`

12.37 installation_contribution_configuration.h

[浏览该文件的文档.](#)

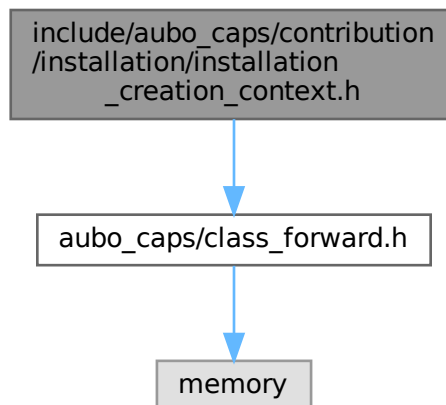
```

00001 #ifndef AUBO_SCOPE_INSTALLATION_CONTRIBUTION_CONFIGURATION_H
00002 #define AUBO_SCOPE_INSTALLATION_CONTRIBUTION_CONFIGURATION_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008
00009 ARCS_CLASS_FORWARD(InstallationContributionConfiguration);
00010
00011 class InstallationContributionConfiguration
00012 {
00013 public:
00014     virtual ~InstallationContributionConfiguration() = default;
00015     // Intentionally left blank for future extension
00016 };
00017
00018 } // namespace aubo_scope
00019 } // namespace arcs
00020
00021 #endif // AUBO_SCOPE_INSTALLATION_CONTRIBUTION_CONFIGURATION_H

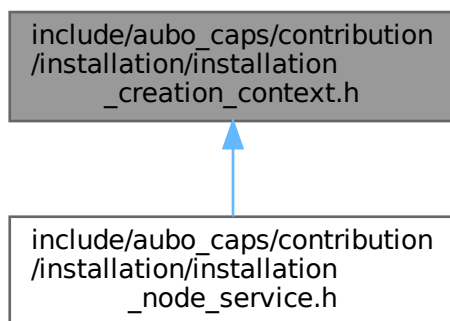
```

12.38 include/aubo_caps/contribution/installation/↵ installation_creation_context.h 文件参考

#include <aubo_caps/class_forward.h>
installation_creation_context.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::InstallationCreationContext`
安装创建上下文提供安装节点创建时所在上下文的相关信息。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (InstallationCreationContext)`

12.39 installation_creation_context.h

[浏览该文件的文档.](#)

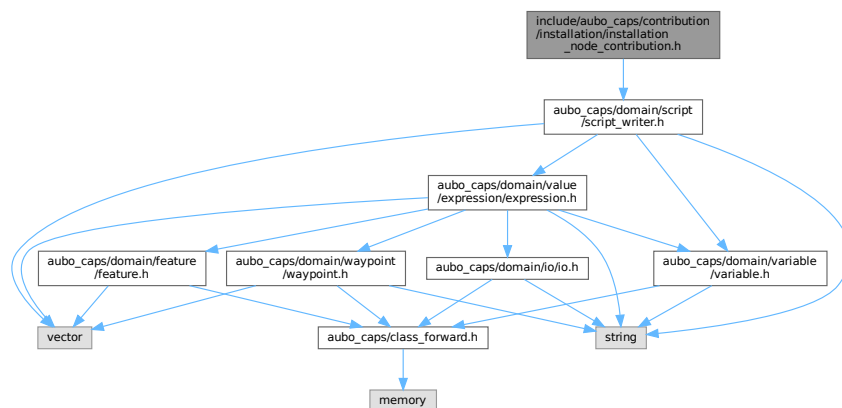
```

00001 #ifndef AUBO_SCOPE_INSTALLATION_CREATION_CONTEXT_H
00002 #define AUBO_SCOPE_INSTALLATION_CREATION_CONTEXT_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008
00009 ARCS_CLASS_FORWARD (InstallationCreationContext);
00010
00011 /**
00012  * \chinese
00013  * 安装创建上下文
00014  * 提供安装节点创建时所在上下文的相关信息。
00015  * \endchinese
00016  * \english
00017  * InstallationCreationContext
00018  * Provides information about the context in which an installation node is
00019  * created.
00020  * \endenglish
00021  */
00022 class InstallationCreationContext
00023 {
00024 public:
00025     enum NodeCreationType
00026     {
00027         NEW,
00028         LOAD
00029     };
00030
00031     InstallationCreationContext (InstallationCreationContext &f);
  
```

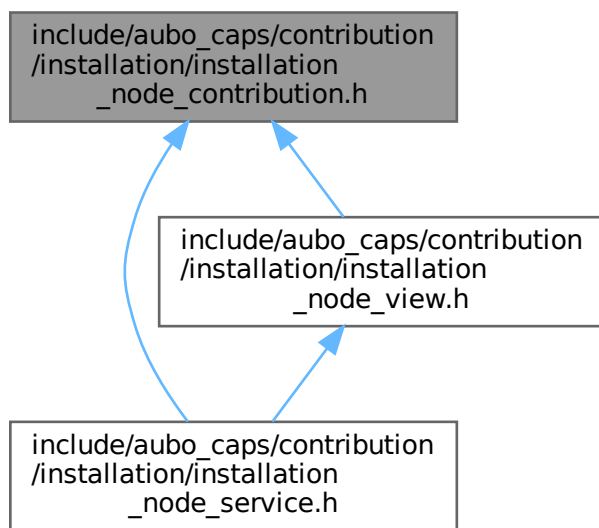
```
00032     InstallationContext(InstallationContext &&f);
00033     ~InstallationContext();
00034
00035     NodeCreationType getNodeCreationType();
00036
00037     std::string getCreatedVersion();
00038     std::string getLastSavedVersion();
00039
00040 private:
00041     friend class DataSwitch;
00042     InstallationContext();
00043     void *d_{ nullptr };
00044 };
00045
00046 } // namespace aubo_scope
00047 } // namespace arcs
00048
00049 #endif // AUBO_SCOPE_INSTALLATION_CREATION_CONTEXT_H
```

12.40 include/aubo_caps/contribution/installation/↵ installation_node_contribution.h 文件参考

#include <aubo_caps/domain/script/script_writer.h>
installation_node_contribution.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::InstallationNodeContribution`
安装节点贡献安装节点贡献的 API。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (InstallationNodeContribution)`

12.41 installation_node_contribution.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_INSTALLATION_NODE_CONTRIBUTION_H
00002 #define AUBO_SCOPE_INSTALLATION_NODE_CONTRIBUTION_H
00003
00004 #include <aubo_caps/domain/script/script_writer.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(InstallationNodeContribution);
00009
00010 /**
00011  * \chinese
00012  * 安装节点贡献
00013  * 安装节点贡献的 API。
00014  * \endchinese
00015  * \english
00016  * InstallationNodeContribution
00017  * API for installation node contributions.
00018  * \endenglish
00019  */
00020 class InstallationNodeContribution
00021 {

```

```

00022 public:
00023     virtual ~InstallationNodeContribution() = default;
00024
00025     /**
00026      * \chinese
00027      * 当终端用户每次在安装标签页中打开此 AuboCap 贡献时调用。
00028      * \endchinese
00029      * \english
00030      * Called each time the end user opens this AuboCap contribution in the
00031      * Installation Tab.
00032      * \endenglish
00033      */
00034     virtual void openView() = 0;
00035
00036     /**
00037      * \chinese
00038      * 当终端用户每次在安装标签页中退出此 AuboCap 贡献时调用。
00039      * \endchinese
00040      * \english
00041      * Called each time the end user exits this AuboCap contribution in the
00042      * Installation Tab.
00043      * \endenglish
00044      */
00045     virtual void closeView() = 0;
00046
00047     /**
00048      * \chinese
00049      * 定义添加到程序启动时执行的脚本代码开头的脚本代码。
00050      *
00051      * @param script_writer 用于向程序脚本代码前言添加脚本命令
00052      * \endchinese
00053      * \english
00054      * Defines script code that is added to the beginning of the script code
00055      * executed when a program is launched.
00056      *
00057      * @param script_writer serves to add script commands to program script code
00058      * preamble.
00059      * \endenglish
00060      */
00061     virtual void generateScript(ScriptWriterPtr script_writer) = 0;
00062 };
00063
00064 } // namespace aubo_scope
00065 } // namespace arcs
00066
00067 #endif // AUBO_SCOPE_INSTALLATIONNODECONTRIBUTION_H

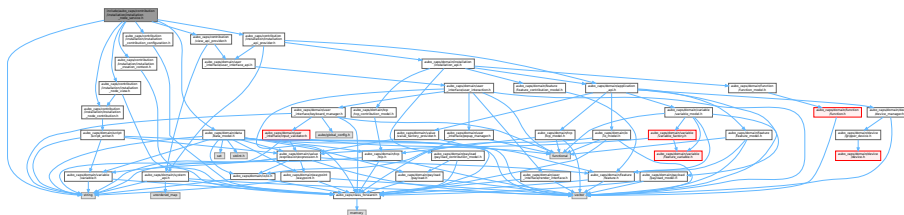
```

12.42 include/aubo_caps/contribution/installation/↵ installation_node_service.h 文件参考

```

#include <string>
#include <aubo_caps/domain/data/data_model.h>
#include <aubo_caps/contribution/view_api_provider.h>
#include <aubo_caps/contribution/installation/installation_node_view.h>
#include <aubo_caps/contribution/installation/installation_node_contribution.h>
#include <aubo_caps/contribution/installation/installation_creation_context.h>
#include <aubo_caps/contribution/installation/installation_contribution_configuration.↵
h>
#include <aubo_caps/contribution/installation/installation_api_provider.h>
installation_node_service.h 的引用 (Include) 关系图:

```



类

- `class arcs::aubo_scope::InstallationNodeService`

安装节点服务定义了 *AuboScope* 中定义和添加安装节点及对应界面（用户界面基于 *Qt*）所需的 *API*。

命名空间

- `namespace arcs`
- `namespace arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (InstallationNodeService)`

12.43 installation__node__service.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_INSTALLATION_SERVICE_H
00002 #define AUBO_SCOPE_INSTALLATION_SERVICE_H
00003
00004 #include <string>
00005 #include <aubo_caps/domain/data/data_model.h>
00006 #include <aubo_caps/contribution/view_api_provider.h>
00007 #include <aubo_caps/contribution/installation/installation_node_view.h>
00008 #include <aubo_caps/contribution/installation/installation_node_contribution.h>
00009 #include <aubo_caps/contribution/installation/installation_creation_context.h>
00010 #include <aubo_caps/contribution/installation/installation_contribution_configuration.h>
00011 #include <aubo_caps/contribution/installation/installation_api_provider.h>
00012
00013 namespace arcs {
00014 namespace aubo_scope {
00015 ARCS_CLASS_FORWARD(InstallationNodeService);
00016
00017 /**
00018  * \chinese
00019  * 安装节点服务
00020  * 定义了在 AuboScope 中定义和添加安装节点及对应界面（用户界面基于 Qt）所需的 API。
00021  * \endchinese
00022  * \english
00023  * InstallationNodeService
00024  * Defines an API required for defining and adding to AuboScope an installation
00025  * node and corresponding screen where the user interface is Qt-based.
00026  * \endenglish
00027  */
00028 class InstallationNodeService
00029 {
00030 public:
00031     virtual ~InstallationNodeService() = default;
00032
00033     /**
00034      * \chinese
00035      * @param locale AuboScope 的当前语言环境，可用于支持多种语言的标题
00036      * @return 在安装标签页左侧导航中显示的文本以及对应安装节点画面的标题。
00037      * 在启动时调用一次。
00038      * \endchinese
00039      * \english
00040      * @param locale the current locale of AuboScope. Can be used for supporting
00041      * titles in several languages.
00042      * @return The text displayed for this installation contribution in the
00043      * left-hand side navigation of the Installation Tab as well as the title of
00044      * the corresponding installation node screen. Called once at start up.
00045      * \endenglish
00046      */
00047     virtual std::string getTitle() = 0;
00048
00049     /**
00050      * \chinese
00051      * @return 此安装贡献显示的图标。在启动时调用一次。
00052      * \endchinese
00053      * \english
00054      * @return The icon displayed for this installation contribution. Called once
00055      * at start up.
00056      * \endenglish
00057      */
00058     virtual std::string getIcon() = 0;
00059
00060     /**
00061      * \chinese
00062      * 此方法在服务注册后调用一次。修改 configuration 参数来配置您的贡献。

```

```

00063      * 配置对象将已经具有匹配大多数用例的默认属性值。
00064      * <p>
00065      * ContributionConfiguration 对象的值将在方法调用后立即读取一次。
00066      * 稍后更改值将不会产生任何效果，因此不要存储对配置对象的引用。
00067      * </p>
00068      * 如果默认值合适，请将此方法留空。
00069      *
00070      * @param configuration 具有默认值的可修改贡献配置
00071      * \endchinese
00072      * \english
00073      * This method is called once after this service is registered. Modify the
00074      * configuration parameter to configure your contribution. The configuration
00075      * object will already have default values for its properties matching most
00076      * use cases. <p> The values of the ContributionConfiguration object will be
00077      * read once immediately after this method call. Changing values at a later
00078      * stage will have no effect, so do not store a reference to the
00079      * configuration object.
00080      * </p>
00081      * If the default values are appropriate, leave this method empty.
00082      *
00083      * @param configuration a modifiable ContributionConfiguration with default
00084      * values
00085      * \endenglish
00086      */
00087 virtual void configureContribution(
00088     InstallationContributionConfigurationPtr configuration) = 0;
00089
00090 /**
00091      * \chinese
00092      * 创建一个新的视图实例，实现安装节点画面的 UI。在加载或创建新安装时调用一次。
00093      *
00094      * @param api_provider 提供对 AuboScope 中与用户界面和终端用户交互相关的
00095      * 功能和服务访问
00096      * @return 视图
00097      * \endchinese
00098      * \english
00099      * Creates a new View instance which implements the UI for your installation
00100      * node screen. Called once when a new installation is loaded or created.
00101      *
00102      * @param api_provider Provides access to functionality and services
00103      * available from within AuboScope related to user interface and end user
00104      * interaction
00105      * @return the view
00106      * \endenglish
00107      */
00108 virtual InstallationNodeViewPtr createView(
00109     ViewApiProviderPtr api_provider) = 0;
00110
00111 /**
00112      * \chinese
00113      * <p> 创建一个新的安装节点实例。</p>
00114      * <p> 返回的节点必须使用提供的数据模型对象来检索和存储其中包含的数据。
00115      * 加载现有安装时，将忽略安装节点构造函数对提供的数据模型所做的所有修改。
00116      * 数据模型对象在同一 AuboCap 贡献的所有安装节点之间共享。</p>
00117      *
00118      * @param api_provider 提供对 AuboScope 中与安装节点相关的功能和服务访问
00119      * @param view 由 createView 创建的视图
00120      * @param model 新安装节点实例的所有配置数据存储和检索的数据模型对象
00121      * @param context 创建此节点的上下文
00122      * @return 新构建的安装节点贡献实例
00123      * \endchinese
00124      * \english
00125      * <p>
00126      * Creates a new installation node instance.
00127      * </p>
00128      * <p>
00129      * The returned node must use the supplied data model object to retrieve and
00130      * store the data contained in it. All modifications to the supplied data
00131      * model from the installation node constructor are ignored when an existing
00132      * installation is loaded. The data model object is shared between all
00133      * installation nodes contributed by the same AuboCap.
00134      * </p>
00135      *
00136      * @param api_provider Provides access to functionality and services
00137      * available from within AuboScope relevant for the installation node
00138      * @param view the View created by {@link #createView(ViewAPIProvider)}
00139      * @param model object where all configuration data of the new installation
00140      * node instance is to be stored in and retrieved from
00141      * @param context the context in which this node is created
00142      * @return the newly constructed installation node contribution instance
00143      * \endenglish
00144      */
00145 virtual InstallationNodeContributionPtr createInstallationNode(
00146     InstallationApiProviderPtr api_provider, InstallationNodeViewPtr view,
00147     DataModelPtr model, InstallationContextPtr context) = 0;
00148 };
00149

```

```

00150 } // namespace aubo_scope
00151 } // namespace arcs
00152
00153 #endif

```

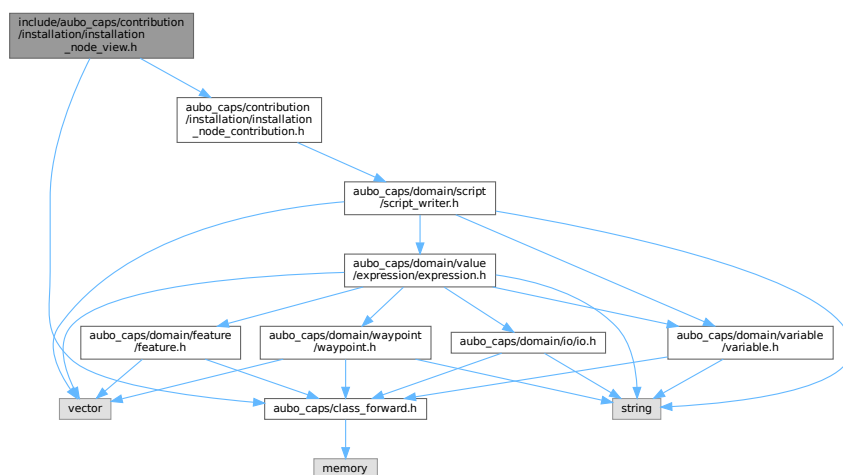
12.44 include/auco_caps/contribution/installation/↵ installation_node_view.h 文件参考

```

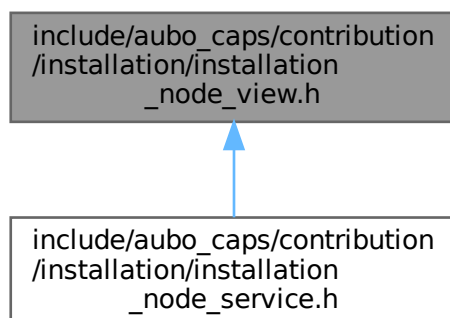
#include <auco_caps/class_forward.h>
#include <auco_caps/contribution/installation/installation_node_contribution.h>

```

installation_node_view.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::auco_scope::InstallationNodeView`

安装节点视图安装节点视图的 API, 定义了对应 `InstallationNodeContribution` 的 UI。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

类型定义

- typedef long [WidgetHandle](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([InstallationNodeView](#))

12.44.1 类型定义说明

WidgetHandle

typedef long [WidgetHandle](#)

在文件 [installation_node_view.h](#) 第 7 行定义.

12.45 installation_node_view.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_INSTALLATION_NODE_VIEW_H
00002 #define AUBO_SCOPE_INSTALLATION_NODE_VIEW_H
00003
00004 #include <aubo_caps/class_forward.h>
00005 #include <aubo_caps/contribution/installation/installation_node_contribution.h>
00006
00007 typedef long WidgetHandle;
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011 ARCS_CLASS_FORWARD(InstallationNodeView);
00012
00013 /**
00014  * \chinese
00015  * 安装节点视图
00016  * 安装节点视图的 API，定义了对应 {@link InstallationNodeContribution} 的 UI。
00017  *
00018  * @param <C> 表示此视图实现 UI 的 {@link InstallationNodeContribution} 类型
00019  *           的（通用）类型参数
00020  * \endchinese
00021  * \english
00022  * InstallationNodeView
00023  * API for an installation node View which defines the UI for the corresponding
00024  * {@link InstallationNodeContribution}
00025  *
00026  * @param <C> the (generic) type parameter for the interface representing the
00027  * type of the
00028  *           {@link InstallationNodeContribution} for which this view
00029  * implements the UI
00030  * \endenglish
00031  */
00032 class InstallationNodeView
00033 {
00034 public:
00035     virtual ~InstallationNodeView() = default;
00036
00037     /**
00038      * \chinese
00039      * 在提供的面板上为对应的 InstallationNodeContribution 构建 UI。使用贡献进行
00040      * 逻辑和数据模型操作。
00041      *
00042      * @param panel 构建 UI 的面板
00043      * @param contribution 对应的安装节点贡献
00044      * \endchinese
00045      * \english
00046      * Build the UI for the corresponding {@link InstallationNodeContribution}
00047      * on the provided panel. Use the contribution for your logic and data model
00048      * manipulations.
00049      *
00050      * @param panel the panel to build the UI on
00051      * @param contribution the corresponding installation node contribution
00052      * \endenglish
00053      */

```

```

00054     virtual void buildUi(WidgetHandle panel,
00055                          InstallationNodeContributionPtr contribution) = 0;
00056 };
00057
00058 } // namespace aubo_scope
00059 } // namespace arcs
00060
00061 #endif // AUBO_SCOPE_INSTALLATION_NODE_VIEW_H

```

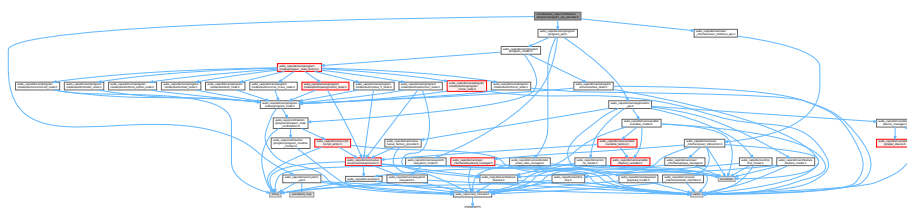
12.46 include/aubo_caps/contribution/program/program__api_provider.h 文件参考

```

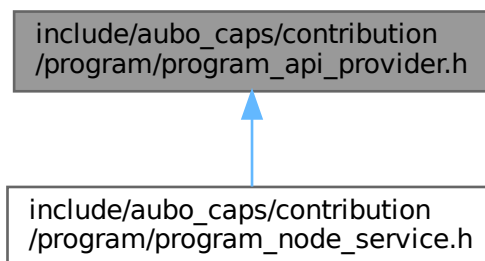
#include <aubo_caps/domain/system_api.h>
#include <aubo_caps/domain/program/program_api.h>
#include <aubo_caps/domain/user_interface/user_interface_api.h>

```

program_api_provider.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ProgramApiProvider](#)

程序 API 提供者提供与程序节点相关的功能和服务访问。这包括 *AuboScope* 内部已有的功能，以及在 *AuboScope* 中创建要使用的额外模型元素。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ProgramApiProvider\)](#)

12.47 program_api_provider.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_PROGRAM_API_PROVIDER_H
00002 #define AUBO_SCOPE_PROGRAM_API_PROVIDER_H
00003
00004 #include <aubo_caps/domain/system_api.h>
00005 #include <aubo_caps/domain/program/program_api.h>
00006 #include <aubo_caps/domain/user_interface/user_interface_api.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010
00011 ARCS_CLASS_FORWARD(ProgramApiProvider);
00012
00013 /**
00014  * \chinese
00015  * 程序 API 提供者
00016  * 提供与程序节点相关的功能和服务访问。这包括 AuboScope 内部已有的功能，以及在
00017  * AuboScope 中创建要使用的额外模型元素。
00018  * \endchinese
00019  * \english
00020  * ProgramApiProvider
00021  * Provides access to functionality and services relevant for program nodes.
00022  * This includes what is available from within AuboScope, as well as creating
00023  * additional model elements to be used within AuboScope.
00024  * \endenglish
00025  */
00026 class ProgramApiProvider
00027 {
00028 public:
00029     ProgramApiProvider(ProgramApiProvider &f);
00030     ProgramApiProvider(ProgramApiProvider &&f);
00031     virtual ~ProgramApiProvider();
00032
00033     /**
00034      * \chinese
00035      * 提供对系统相关功能的访问
00036      *
00037      * @return SystemAPI 实例
00038      * \endchinese
00039      * \english
00040      * Provides access to system related functionality
00041      *
00042      * @return an instance of SystemAPI
00043      * \endenglish
00044      */
00045     SystemApiPtr getSystemApi();
00046
00047     /**
00048      * \chinese
00049      * 提供对用户界面和终端用户交互相关功能的访问
00050      *
00051      * @return UserInterfaceAPI 实例
00052      * \endchinese
00053      * \english
00054      * Provides access to functionality related to user interface and end user
00055      * interaction
00056      *
00057      * @return an instance of UserInterfaceAPI
00058      * \endenglish
00059      */
00060     UserInterfaceApiPtr getUserInterfaceApi();
00061
00062     /**
00063      * \chinese
00064      * 提供对程序相关功能的访问
00065      *
00066      * @return ProgramAPI 实例
00067      * \endchinese
00068      * \english
00069      * Provides access to functionality relevant related to programs
00070      *
00071      * @return an instance of ProgramAPI
00072      * \endenglish
00073      */
00074     ProgramApiPtr getProgramApi();
00075
00076 private:
00077     ProgramApiProvider();
00078     friend class DataSwitch;
00079     void *d_{ nullptr };
00080 };
00081
00082 } // namespace aubo_scope
00083 } // namespace arcs

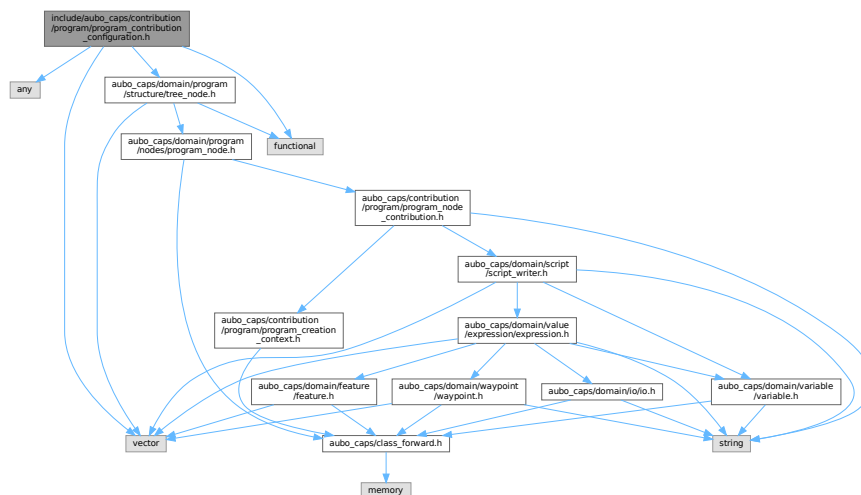
```

```
00084
00085 #endif // AUBO_SCOPE_PROGRAM_API_PROVIDER_H
```

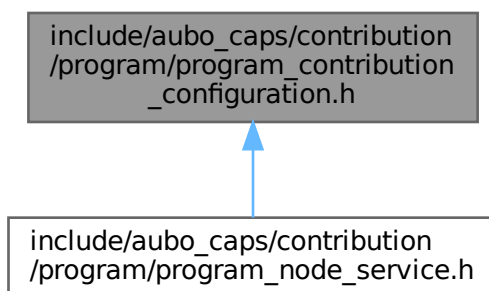
12.48 include/aubo_caps/contribution/program/program_↵ contribution_configuration.h 文件参考

```
#include <any>
#include <vector>
#include <functional>
#include <aubo_caps/domain/program/structure/tree_node.h>
```

program_contribution_configuration.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ProgramContributionConfiguration](#)
程序贡献配置提供对安装节点贡献属性进行配置的访问。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`ProgramContributionConfiguration`)

12.49 program_contribution_configuration.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_CONTRIBUTION_CONFIGURATION_H
00002 #define AUBO_SCOPE_CONTRIBUTION_CONFIGURATION_H
00003
00004 #include <any>
00005 #include <vector>
00006 #include <functional>
00007
00008 #include <aubo_caps/domain/program/structure/tree_node.h>
00009
00010 namespace arcs {
00011 namespace aubo_scope {
00012
00013 ARCS_CLASS_FORWARD(ProgramContributionConfiguration);
00014
00015 /**
00016  * \chinese
00017  * 程序贡献配置
00018  * 提供对安装节点贡献属性进行配置的访问。
00019  * \endchinese
00020  * \english
00021  * ProgramContributionConfiguration
00022  * Provides access to configure the properties of an installation node
00023  * contribution.
00024  * \endenglish
00025  */
00026 class ProgramContributionConfiguration
00027 {
00028 public:
00029     ProgramContributionConfiguration(ProgramContributionConfiguration &f);
00030     ProgramContributionConfiguration(ProgramContributionConfiguration &&f);
00031     virtual ~ProgramContributionConfiguration();
00032
00033     /**
00034      * If set to <code>true</code> it is possible for the program node to have
00035      * child nodes.
00036      *
00037      * <p>
00038      * The default value is <code>false</code>.
00039      * </p>
00040      *
00041      * @param childrenAllowed determines whether this node can have child nodes
00042      */
00043     void setChildrenAllowed(bool childrenAllowed);
00044     void setChildrenAllowedOnly(const std::vector<std::string> &childrenList);
00045
00046     /**
00047      * If set to <code>false</code> this program node will not appear in the
00048      * Program Tab in AuboScope and can not be inserted in the program tree by
00049      * the end user. It can only be inserted programmatically using the
00050      * {@link ProgramNodeFactory}. This
00051      * also means, it can not be inserted using copy/paste of that node
00052      * directly. It can however be inserted by copy/paste of an ancestor of such
00053      * a node.
00054      *
00055      * <p>
00056      * The default value is <code>true</code>.
00057      * </p>
00058      *
00059      * @param userInsertable determines whether the end user can insert this
00060      * node
00061      */
00062     void setUserInsertable(bool userInsertable);
00063
00064     /**
00065      * If set to <code>true</code> it is not possible for the end user to create
00066      * new program nodes of this type. Loading of existing programs will however
00067      * still be supported for this program node.
00068      *
00069      * <p>

```

```

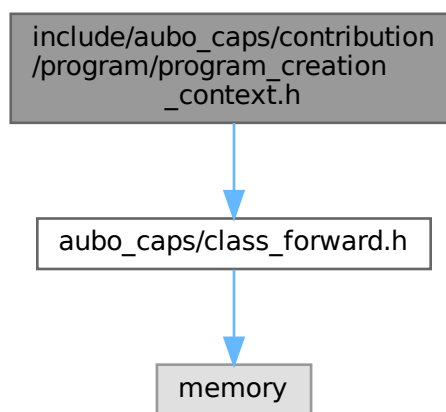
00070      * The default value is <code>false</code>.
00071      * </p>
00072      *
00073      * @param deprecated determines whether this node is deprecated
00074      */
00075      void setDeprecated(bool deprecated);
00076
00077      /**
00078      * Set the display order id for this type of program node contribution. This
00079      * makes it possible to specify how all the program node contributions from
00080      * a AuboCap should be ordered/sorted in AuboScope.
00081      *
00082      * <p>
00083      * All program node contributions from a AuboCap will be grouped together
00084      * and sorted (in ascending order) according to their display order id.
00085      * </p>
00086      *
00087      * <p>
00088      * The default value is 0.0. If the display order id for all program node
00089      * contributions in a AuboCap have the default value, the standard built-in
00090      * sorting in AuboScope will be used.
00091      * </p>
00092      *
00093      * @param displayOrderId Determines the display order of the AuboCap program
00094      * node type
00095      */
00096      void setDisplayOrderId(double displayOrderId);
00097
00098      /**
00099      * <p>
00100      * Configure whether or not the end user is allowed to set a breakpoint on
00101      * or single step the AuboCap program node.
00102      * </p>
00103      *
00104      * <p>
00105      * The default value is <code>true</code>.
00106      * </p>
00107      *
00108      * @param isBreakpointAllowed <code>true</code> if it should be allowed to
00109      * set a breakpoint on or single step this AuboCap program node. If it
00110      * should be disallowed, the value must be <code>false</code>.
00111      */
00112      void setAllowBreakpointOnNode(bool isBreakpointAllowed);
00113
00114      /**
00115      * <p>
00116      * Configure whether or not the end user is allowed to set a breakpoint on
00117      * or single step any of the child nodes in the sub-tree under the AuboCap
00118      * program node. Note that it will only be possible to set a breakpoint on a
00119      * child node if the child node itself supports breakpoints.
00120      * </p>
00121      *
00122      * <p>
00123      * The default value is <code>false</code>.
00124      * </p>
00125      *
00126      * @param isBreakpointOnChildNodesAllowed <code>true</code> if it should be
00127      * allowed to set a breakpoint on or single step any of the child nodes in
00128      * the sub-tree under this AuboCap program node. If it should be disallowed,
00129      * the value must be <code>false</code>.
00130      */
00131      void setAllowBreakpointOnChildNodesInSubtree(
00132          bool isBreakpointOnChildNodesAllowed);
00133
00134      /**
00135      * <p>
00136      * Configure whether or not the end user can start the program from the
00137      * AuboCap program node when it is selected in the program tree.
00138      * </p>
00139      *
00140      * <p>
00141      * The default value is <code>true</code>.
00142      * </p>
00143      *
00144      * @param isStartFromNodeAllowed <code>true</code> if it should be allowed
00145      * to start the program directly from this AuboCap program node. If it
00146      * should be disallowed, the value must be <code>false</code>.
00147      *
00148      */
00149      void setAllowStartFromNode(bool isStartFromNodeAllowed);
00150
00151      /**
00152      * <p>
00153      * Configure whether or not the end user can start the program directly from
00154      * any selected child node in the sub-tree under the AuboCap program node.
00155      * Note that it will only be possible to start from a selected child node if
00156      * the child node itself supports starting from the child node.

```

```
00157      * </p>
00158      *
00159      * <p>
00160      * The default value is <code>>false</code>.
00161      * </p>
00162      *
00163      * @param isStartFromChildNodesAllowed <code>true</code> if it should be
00164      * allowed to start the program directly from any child program node. If it
00165      * should be disallowed, the value must be <code>>false</code>..
00166      */
00167 void setAllowStartFromChildNodesInSubtree(
00168     bool isStartFromChildNodesAllowed);
00169
00170 /**
00171  * Configure parent node of this node. End user cannot insert this node to
00172  *
00173  * The default value is <code>""</code>.
00174  * @param parentNodeName
00175  */
00176 void setAllowInsertOnlyIntoNodes(
00177     const std::vector<std::string> &parentNodeNames);
00178
00179 private:
00180     ProgramContributionConfiguration();
00181     friend class DataSwitch;
00182     void *d_{ nullptr };
00183 };
00184
00185 } // namespace aubo_scope
00186 } // namespace arcs
00187
00188 #endif // AUBO_SCOPE_CONTRIBUTION_CONFIGURATION_H
```

12.50 include/aubo_caps/contribution/program/program_↵ creation_context.h 文件参考

#include <aubo_caps/class_forward.h>
program_creation_context.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- `class arcs::aubo_scope::ProgramCreationContext`
程序创建上下文提供安装节点创建时所在上下文的相关信息。

命名空间

- `namespace arcs`
- `namespace arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ProgramCreationContext)`

12.51 program_creation_context.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_PROGRAM_CREATION_CONTEXT_H
00002 #define AUBO_SCOPE_PROGRAM_CREATION_CONTEXT_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(ProgramCreationContext);
00009
00010 /**
00011  * \chinese
00012  * 程序创建上下文
00013  * 提供安装节点创建时所在上下文的相关信息。
00014  * \endchinese
00015  * \english
00016  * ProgramCreationContext
00017  * Provides information about the context in which an installation node is
00018  * created.
00019  * \endenglish
00020  */
00021 class ProgramCreationContext
00022 {
00023 public:
00024     enum NodeCreationType
00025     {
00026         NEW,
00027         LOAD
00028     };
00029
00030     ProgramCreationContext(ProgramCreationContext &f);
00031     ProgramCreationContext(ProgramCreationContext &&f);
00032     ~ProgramCreationContext();
00033
00034     NodeCreationType getNodeCreationType();
00035
00036     std::string getCreatedVersion();
00037     std::string getLastSavedVersion();
00038
00039 private:
00040     friend class DataSwitch;
00041     ProgramCreationContext();
00042     void *d_{ nullptr };
00043 };
00044
00045 } // namespace aubo_scope
00046 } // namespace arcs
00047
00048 #endif // AUBO_SCOPE_PROGRAM_CREATION_CONTEXT_H

```

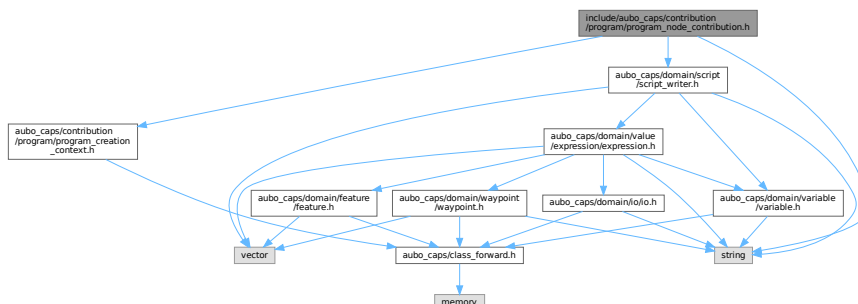
12.52 include/aubo_caps/contribution/program/program_node_contribution.h 文件参考

```

#include <string>
#include <aubo_caps/contribution/program/program_creation_context.h>

```


#include <aubo_caps/domain/script/script_writer.h>
 program_node_contribution.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ProgramNodeContribution](#)
 程序节点贡献定义了 *AuboScope* 中指定 *AuboCap* 程序节点所需的 API。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ProgramNodeContribution\)](#)

12.53 program_node_contribution.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_PROGRAM_NODE_CONTRIBUTION_H
00002 #define AUBO_SCOPE_PROGRAM_NODE_CONTRIBUTION_H
00003
00004 #include <string>
00005 #include <aubo_caps/contribution/program/program_creation_context.h>
00006 #include <aubo_caps/domain/script/script_writer.h>
00007
00008 namespace arcs {
00009     namespace aubo_scope {
00010         ARCS_CLASS_FORWARD(ProgramNodeContribution);
00011
00012         /**
00013          * \chinese
00014          * 程序节点贡献
00015          * 定义了 AuboScope 中指定 AuboCap 程序节点所需的 API。
00016          * \endchinese
00017          * \english
00018          * ProgramNodeContribution
00019          * Defines an API required for specifying a AuboCap Program Node within
00020          * AuboScope.
00021          * \endenglish
00022          */
00023         class ProgramNodeContribution
00024         {
00025         public:
00026             virtual ~ProgramNodeContribution() = default;
00027

```

```

00028  /**
00029  * \chinese
00030  * 当此节点在程序树中被选中时调用。
00031  * \endchinese
00032  * \english
00033  * Called when this node is selected in the program tree.
00034  * \endenglish
00035  */
00036  virtual void openView() = 0;
00037
00038  /**
00039  * \chinese
00040  * 当此节点在程序树中被取消选中或导航到其他视图时调用。
00041  * \endchinese
00042  * \english
00043  * Called when this node is unselected in the program tree or when
00044  * navigating to another view.
00045  * \endenglish
00046  */
00047  virtual void closeView() = 0;
00048
00049  /**
00050  * \chinese
00051  * @return 程序树中为此程序节点显示的文本。
00052  * @throws IllegalStateException 如果数据模型或子程序子树被修改
00053  * \endchinese
00054  * \english
00055  * @return The text displayed in the Program Tree for the program node.
00056  * @throws IllegalStateException if the data model or child program
00057  * sub-tree is modified
00058  * \endenglish
00059  */
00060  virtual std::string getTitle() = 0;
00061
00062  /**
00063  * \chinese
00064  * <p>
00065  * 如果此程序节点完整，应返回 <code>true</code>，否则返回 <code>false</code>。
00066  * </p>
00067  *
00068  * <p>
00069  * 未定义的程序节点将在程序树中以黄色标记。只要程序包含一个或多个未定义的
00070  * 程序节点，用户就无法启动程序。
00071  * </p>
00072  *
00073  * 如果此程序节点在其 DataModel 中存储了不可解析的实体（例如
00074  * PersistedVariable 或 TCP），则该程序节点将被视为未定义。
00075  *
00076  * @return 如果此程序节点完整则返回 <code>true</code>，否则返回
00077  * <code>false</code>。
00078  * @throws IllegalStateException 如果数据模型或子程序子树被修改
00079  * \endchinese
00080  * \english
00081  * <p>
00082  * Should return <code>true</code> if this program node is complete, and
00083  * <code>false</code> otherwise.
00084  * </p>
00085  *
00086  * <p>
00087  * Undefined program nodes will be marked with yellow in the program
00088  * tree. The user cannot start the program as long as it contains one or
00089  * more undefined program nodes.
00090  * </p>
00091  *
00092  * If this program node stores an unresolvable entity (e.g. {@link
00093  * variable/PersistedVariable} or {@link tcp/TCP}) in its DataModel, the
00094  * program node will be treated as undefined.
00095  *
00096  * @return <code>true</code> if this program node is complete and otherwise
00097  * <code>false</code>.
00098  * @throws IllegalStateException if the data model or child program
00099  * sub-tree is modified.
00100  * \endenglish
00101  */
00102  virtual bool isDefined() = 0;
00103
00104  /**
00105  * \chinese
00106  * 定义此程序节点在程序中要生成的脚本代码。
00107  *
00108  * @param script_writer 用于添加定义此程序节点行为的脚本命令
00109  * \endchinese
00110  * \english
00111  * Defines the script code to be generated by this program node within
00112  * the program.
00113  *
00114  * @param script_writer serves to add script commands that define the behaviour

```

```

00115     * of this program node.
00116     * \endenglish
00117     */
00118     virtual void generateScript(ScriptWriterPtr script_writer) = 0;
00119
00120     /**
00121     * \chinese
00122     * @brief postAction 发送 HTTP POST 请求
00123     * @param pattern URL 模式
00124     * @param body 请求体
00125     * @return 响应内容
00126     * \endchinese
00127     * \english
00128     * @brief post
00129     * @param pattern
00130     * @param body
00131     * @return
00132     * \endenglish
00133     */
00134     virtual std::string postAction(const char *pattern, const std::string &body)
00135     {
00136         return "";
00137     }
00138
00139     /**
00140     * \chinese
00141     * @brief getAction 发送 HTTP GET 请求
00142     * @param pattern URL 模式
00143     * @param body 请求体
00144     * @return 响应内容
00145     * \endchinese
00146     * \english
00147     * @brief GET
00148     * @param pattern
00149     * @param body
00150     * @return
00151     * \endenglish
00152     */
00153     virtual std::string getAction(const char *pattern, const std::string &body)
00154     {
00155         return "";
00156     }
00157 };
00158
00159 } // namespace aubo_scope
00160 } // namespace arcs
00161
00162 #endif // AUBO_SCOPE_PROGRAM_NODE_CONTRIBUTION_H

```

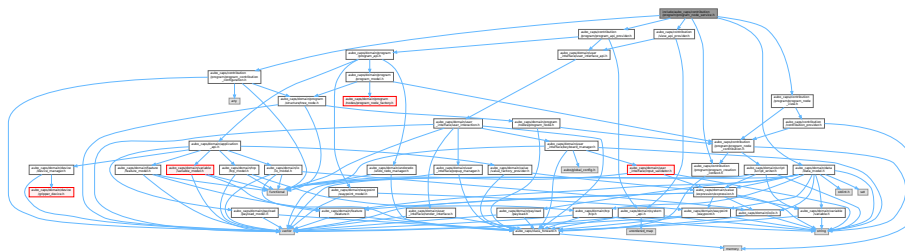
12.54 include/aubo_caps/contribution/program/program_↵ node_service.h 文件参考

```

#include <aubo_caps/contribution/view_api_provider.h>
#include <aubo_caps/contribution/program/program_node_view.h>
#include <aubo_caps/contribution/program/program_creation_context.h>
#include <aubo_caps/contribution/program/program_api_provider.h>
#include <aubo_caps/contribution/program/program_contribution_configuration.h>
#include <aubo_caps/domain/data/data_model.h>

```

program_node_service.h 的引用 (Include) 关系图:



类

- class arcs::aubo_scope::ProgramNodeService

程序节点服务定义了向 *AuboScope* 添加新类型程序节点所需的 *API*。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ProgramNodeService)`

12.55 program_node_service.h

浏览该文件的文档.

```
00001 #ifndef AUBO_SCOPE_PROGRAM_NODE_SERVICE_H
00002 #define AUBO_SCOPE_PROGRAM_NODE_SERVICE_H
00003
00004 #include <aubo_caps/contribution/view_api_provider.h>
00005 #include <aubo_caps/contribution/program/program_node_view.h>
00006 #include <aubo_caps/contribution/program/program_creation_context.h>
00007 #include <aubo_caps/contribution/program/program_api_provider.h>
00008 #include <aubo_caps/contribution/program/program_contribution_configuration.h>
00009 #include <aubo_caps/domain/data/data_model.h>
00010
00011 namespace arcs {
00012 namespace aubo_scope {
00013 ARCS_CLASS_FORWARD(ProgramNodeService);
00014
00015 /**
00016  * \chinese
00017  * 程序节点服务
00018  * 定义了向 AuboScope 添加新类型程序节点所需的 API。
00019  * \endchinese
00020  * \english
00021  * ProgramNodeService
00022  * Defines an API required for adding a new type of program node to AuboScope.
00023  * \endenglish
00024  */
00025 class ProgramNodeService
00026 {
00027 public:
00028     virtual ~ProgramNodeService() = default;
00029
00030     /**
00031      * \chinese
00032      * 获取此类程序节点的唯一标识符。该标识符将在存储使用此程序节点的程序时使用。
00033      *
00034      * @return 唯一标识符
00035      * \endchinese
00036      * \english
00037      * Get the unique identifier for this kind of program node. The identifier
00038      * will be used when storing programs that use the program node.
00039      *
00040      * @return the unique identifier.
00041      * \endenglish
00042      */
00043     virtual std::string getId() = 0;
00044
00045     /**
00046      * \chinese
00047      * @return 通过此工厂创建的程序节点在结构标签页中显示的文本
00048      * \endchinese
00049      * \english
00050      * @return The text displayed in the Structure Tab for program nodes created
00051      * by this factory.
00052      * \endenglish
00053      */
00054     virtual std::string getTitle() = 0;
00055
00056     /**
00057      * \chinese
00058      * @return 通过此工厂创建的程序节点显示的图标
00059      * \endchinese
00060      * \english
00061      * @return The icon displayed for program nodes created by this factory.
00062      * \endenglish
00063      */
00064     virtual std::string getIcon() = 0;
00065
00066     /**
```

```

00067     * \chinese
00068     * 配置贡献，设置程序节点的属性和行为。
00069     * @param configuration 贡献配置对象
00070     * \endchinese
00071     * \english
00072     * configureContribution
00073     * @param configuration
00074     * \endenglish
00075     */
00076     virtual void configureContribution(
00077         ProgramContributionConfigurationPtr configuration) = 0;
00078
00079     /**
00080     * \chinese
00081     * 创建一个新的视图实例，实现程序节点画面的 UI。在加载或创建新程序时调用一次。
00082     *
00083     * @param api_provider 提供对 AuboScope 中与用户界面和终端用户交互相关的
00084     * 功能和服务访问
00085     * @return 视图
00086     * \endchinese
00087     * \english
00088     * Creates a new View instance which implements the UI for your program node
00089     * screen. Called once when a new program is loaded or created.
00090     *
00091     * @param api_provider Provides access to functionality and services
00092     * available from within AuboScope related to user interface and end user
00093     * interaction
00094     * @return the view
00095     * \endenglish
00096     */
00097     virtual ProgramNodeViewPtr createView(ViewApiProviderPtr api_provider) = 0;
00098
00099     /**
00100     * \chinese
00101     * <p>
00102     * 创建一个新的程序节点贡献实例。
00103     * </p>
00104     *
00105     * <p>
00106     * 返回的节点必须使用提供的数据模型对象来检索和存储其中包含的数据。
00107     * 加载或复制/粘贴现有程序节点时，将忽略程序节点构造函数对提供的数据模型
00108     * 所做的所有修改。对模型对象的每次更改都将在程序树中注册为一个单独的
00109     * 撤销/重做事件。
00110     * </p>
00111     *
00112     * @param api_provider 提供对 AuboScope 中可用的域数据相关功能的访问
00113     * @param view 程序节点视图
00114     * @param model 新程序节点实例的所有配置数据存储和检索的数据模型对象
00115     * @param context 创建上下文
00116     * @return 新构建的程序节点贡献实例
00117     * \endchinese
00118     * \english
00119     * <p>
00120     * Creates a new program node contribution instance.
00121     * </p>
00122     *
00123     * <p>
00124     * The returned node must use the supplied data model object to retrieve and
00125     * store the data contained in it. All modifications to the supplied data
00126     * model from the program node constructor are ignored when an existing
00127     * program node is loaded or copy/pasted. Every change to the model object
00128     * is registered as a separate undo/redo event in the program tree.
00129     * </p>
00130     *
00131     * @param api_provider the AuboCap API object with access to domain data
00132     * @param view the View created by createView
00133     * @param model object where all configuration data of the new program node
00134     * instance is to be stored in and retrieved from
00135     * @param context the creation context
00136     * @return the newly constructed program node contribution instance
00137     * \endenglish
00138     */
00139     virtual ProgramNodeContributionPtr createNode(
00140         ProgramApiProviderPtr api_provider, ProgramNodeViewPtr view,
00141         DataModelPtr model, ProgramCreationContextPtr context) = 0;
00142 };
00143
00144 } // namespace aubo_scope
00145 } // namespace arcs
00146
00147 #endif // AUBO_SCOPE_PROGRAM_NODE_SERVICE_H

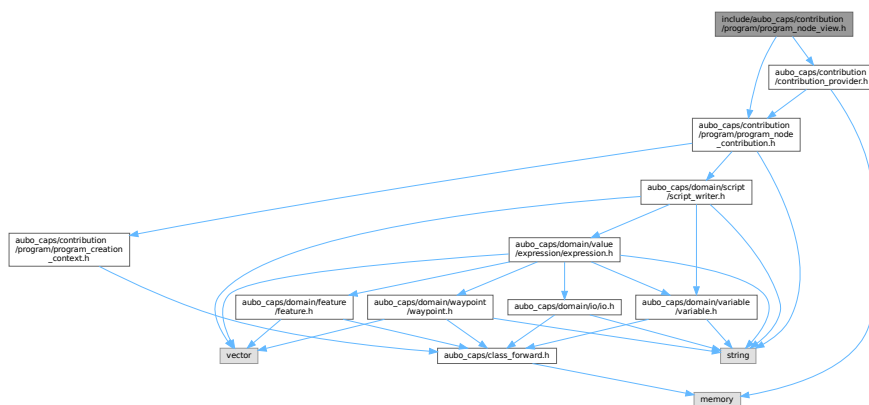
```

12.56 include/aubo_caps/contribution/program/program_node_view.h 文件参考

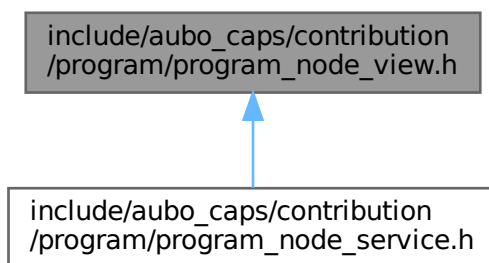
```
#include <aubo_caps/contribution/program/program_node_contribution.h>
```

```
#include <aubo_caps/contribution/contribution_provider.h>
```

program_node_view.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ProgramNodeView](#)

程序节点视图程序节点视图的 API, 定义了对应 *ProgramNodeContribution* 的 UI。 *ProgramNodeView* 实例是多个贡献之间的共享实体。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

类型定义

- typedef long [arcs::aubo_scope::WidgetHandle](#)

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ProgramNodeView)`

12.57 program_node_view.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_PROGRAM_NODE_VIEW_H
00002 #define AUBO_SCOPE_PROGRAM_NODE_VIEW_H
00003
00004 #include <aubo_caps/contribution/program/program_node_contribution.h>
00005 #include <aubo_caps/contribution/contribution_provider.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 typedef long WidgetHandle;
00010 ARCS_CLASS_FORWARD(ProgramNodeView);
00011
00012 /**
00013  * \chinese
00014  * 程序节点视图
00015  * 程序节点视图的 API，定义了对应 {@link ProgramNodeContribution} 的 UI。
00016  * {@link ProgramNodeView} 实例是多个贡献之间的共享实体。
00017  * \endchinese
00018  * \english
00019  * ProgramNodeView
00020  * API for a program node View which defines the UI for the corresponding {@link
00021  * ProgramNodeContribution}. The instance of {@link ProgramNodeView} is a
00022  * shared entity between several contributions.
00023  * \endenglish
00024  */
00025 class ProgramNodeView
00026 {
00027 public:
00028     virtual ~ProgramNodeView() = default;
00029
00030     /**
00031      * \chinese
00032      * 在提供的面板上为对应的 ProgramNodeContribution 构建 UI。使用贡献进行
00033      * 逻辑和数据模型操作。
00034      *
00035      * @param panel 构建 UI 的面板
00036      * @param provider 当前选中的对应程序节点贡献的提供者
00037      * \endchinese
00038      * \english
00039      * Build the UI for the corresponding {@link ProgramNodeContribution} on the
00040      * provided panel. Use the contribution for your logic and data model
00041      * manipulations.
00042      *
00043      * @param panel the panel to build the UI on
00044      * @param provider the provider for corresponding program node contribution
00045      * currently selected (in the program tree)
00046      * \endenglish
00047      */
00048     virtual void buildUi(
00049         WidgetHandle panel,
00050         ContributionProviderPtr<ProgramNodeContribution> provider) = 0;
00051 };
00052
00053 } // namespace aubo_scope
00054 } // namespace arcs
00055
00056 #endif // AUBO_SCOPE_PROGRAM_NODE_VIEW_H

```

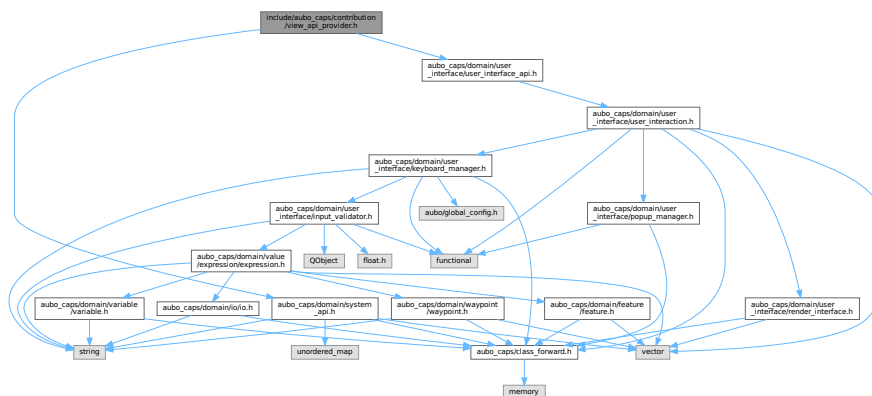
12.58 include/aubo_caps/contribution/view_api_provider.h
文件参考

```

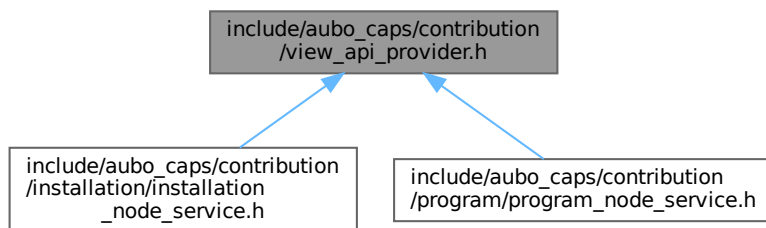
#include <aubo_caps/domain/system_api.h>
#include <aubo_caps/domain/user_interface/user_interface_api.h>

```

view_api_provider.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ViewApiProvider](#)

视图 API 提供者提供与安装或程序节点贡献的视图 (UI) 相关的功能访问。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ViewApiProvider\)](#)

12.59 view_api_provider.h

浏览该文件的文档.

```
00001 #ifndef AUBO_SCOPE_VIEW_API_PROVIDER_H
00002 #define AUBO_SCOPE_VIEW_API_PROVIDER_H
00003
00004 #include <aubo_caps/domain/system_api.h>
00005 #include <aubo_caps/domain/user_interface/user_interface_api.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(ViewApiProvider);
00010
```



```

00011 /**
00012  * \chinese
00013  * 视图 API 提供者
00014  * 提供与安装或程序节点贡献的视图 (UI) 相关的功能访问。
00015  *
00016  * <b> 注意: </b> 此接口不适用于具有基于 HTML 用户界面的 AuboCap 程序和安装节点。
00017  * \endchinese
00018  * \english
00019  * ViewApiProvider
00020  * Provides access to functionality relevant for the View (UI) of an
00021  * installation or a program node contribution.
00022  *
00023  * <b>NOTE:</b> This interface is not relevant for AuboCap program and
00024  * installation nodes with a HTML-based user interface.
00025  * \endenglish
00026  */
00027 class ViewApiProvider
00028 {
00029 public:
00030     ViewApiProvider();
00031     ViewApiProvider(ViewApiProvider &f);
00032     ViewApiProvider(ViewApiProvider &&f);
00033     virtual ~ViewApiProvider();
00034
00035     /**
00036     * \chinese
00037     * 提供对系统相关功能的访问。
00038     *
00039     * @return SystemAPI 实例
00040     * \endchinese
00041     * \english
00042     * Provides access to system related functionality.
00043     *
00044     * @return an instance of SystemAPI.
00045     * \endenglish
00046     */
00047     SystemApiPtr getSystemApi();
00048
00049     /**
00050     * \chinese
00051     * 提供对用户界面和用户交互相关功能的访问。
00052     *
00053     * @return UserInterfaceAPI 实例
00054     * \endchinese
00055     * \english
00056     * Provides access to functionality related to user interface and user
00057     * interaction.
00058     *
00059     * @return an instance of UserInterfaceAPI.
00060     * \endenglish
00061     */
00062     UserInterfaceApiPtr getUserInterfaceApi();
00063
00064 private:
00065     friend class DataSwitch;
00066     void *d_{ nullptr };
00067 };
00068
00069 } // namespace aubo_scope
00070 } // namespace arcs
00071
00072 #endif // AUBO_SCOPE_VIEW_API_PROVIDER_H

```

12.60 include/aubo_caps/domain/application_api.h 文件参考

```

#include <aubo_caps/domain/io/io_model.h>
#include <aubo_caps/domain/device/device_manager.h>
#include <aubo_caps/domain/feature/feature_model.h>
#include <aubo_caps/domain/payload/payload_model.h>
#include <aubo_caps/domain/tcp/tcp_model.h>
#include <aubo_caps/domain/variable/variable_model.h>
#include <aubo_caps/domain/value/value_factory_provider.h>

```



```

00005 #include <aubo_caps/domain/device/device_manager.h>
00006 #include <aubo_caps/domain/feature/feature_model.h>
00007 #include <aubo_caps/domain/payload/payload_model.h>
00008 #include <aubo_caps/domain/tcp/tcp_model.h>
00009 #include <aubo_caps/domain/variable/variable_model.h>
00010 #include <aubo_caps/domain/value/value_factory_provider.h>
00011
00012 namespace arcs {
00013
00014 namespace aubo_sdk {
00015     ARCS_CLASS_FORWARD(RobotProxy);
00016 }
00017
00018 namespace aubo_scope {
00019     ARCS_CLASS_FORWARD(ApplicationApi);
00020     ARCS_CLASS_FORWARD(InstallationNodeContribution);
00021
00022 /**
00023  * \chinese
00024  * @defgroup ApplicationApi ApplicationApi(应用程序)
00025  * @ingroup ApplicationApi
00026  * 应用程序接口
00027  * 提供对各种不同应用程序和上下文相关的功能与服务的访问。这包括 AuboScope 内部已有的功
00028  * 能，以及在 AuboScope 中创建要使用的额外模型元素。
00029  * \endchinese
00030  * \english
00031  * @defgroup ApplicationApi ApplicationApi
00032  * @ingroup ApplicationApi
00033  * ApplicationApi
00034  * Provides access to functionality and services which are relevant for various
00035  * different applications and contexts. This includes what is available from
00036  * within AuboScope, as well as creating additional model elements to be used
00037  * within AuboScope.
00038  * \endenglish
00039  */
00040
00041 class ARCS_ABI_EXPORT ApplicationApi
00042 {
00043 public:
00044     ApplicationApi(ApplicationApi &f);
00045     ApplicationApi(ApplicationApi &&f);
00046     virtual ~ApplicationApi();
00047
00048 /**
00049  * @ingroup ApplicationApi
00050  * @ref IoModel
00051  * \chinese
00052  * 获取 IO 模型接口，用于操作机器人的输入和输出。
00053  * @return 用于操作机器人输入和输出的接口。
00054  * \endchinese
00055  * \english
00056  * Returns an interface for working with the inputs and outputs of the robot.
00057  * @return An interface for working with the inputs and outputs of the robot.
00058  * \endenglish
00059  */
00060     IoModelPtr getIoModel();
00061
00062 /**
00063  * @ingroup ApplicationApi
00064  * @ref FeatureModel
00065  * \chinese
00066  * 获取特征模型接口，用于操作当前安装中的特征。
00067  * @return 用于操作当前安装中特征的接口。
00068  * \endchinese
00069  * \english
00070  * Returns an interface for working with the features of the current
00071  * installation.
00072  * @return An interface for working with the features of the current
00073  * installation.
00074  * \endenglish
00075  */
00076     FeatureModelPtr getFeatureModel();
00077
00078 /**
00079  * @ingroup ApplicationApi
00080  * @ref TcpModel
00081  * \chinese
00082  * 获取 TCP 模型接口，用于访问当前安装中的 TCP。
00083  * @return 用于访问当前安装中 TCP 的接口。
00084  * \endchinese
00085  * \english
00086  * Returns an interface for accessing the TCPs of the current installation.
00087  * @return An interface for accessing the TCPs of the current installation.
00088  * \endenglish
00089  */
00090     TcpModelPtr getTcpModel();
00091

```

```

00092  /**
00093   * @ingroup ApplicationApi
00094   * @ref PayloadModel
00095   * \chinese
00096   * 获取负载模型接口，用于访问当前 AuboScope 安装中的负载。
00097   *
00098   * @return 用于访问当前 AuboScope 安装中负载的接口。
00099   * \endchinese
00100   * \english
00101   * Gets an interface which provides access the payloads in the current
00102   * AuboScope installation.
00103   *
00104   * @return An interface for accessing the payloads in the current AuboScope
00105   * installation.
00106   * \endenglish
00107   */
00108  PayloadModelPtr getPayloadModel();
00109
00110  /**
00111   * @ingroup ApplicationApi
00112   * @ref VariableModel
00113   * \chinese
00114   * 获取变量模型接口，用于操作变量。
00115   * @return 用于操作变量的接口。
00116   * \endchinese
00117   * \english
00118   * Returns an interface for working with variables.
00119   * @return An interface for working with variables.
00120   * \endenglish
00121   */
00122  VariableModelPtr getVariableModel();
00123
00124  /**
00125   * @ingroup ApplicationApi
00126   * @ref ValueFactory
00127   * \chinese
00128   * 获取值工厂接口，用于访问能够创建值对象的各种工厂。
00129   * @return 用于访问能够创建值对象的各种工厂的接口。
00130   * \endchinese
00131   * \english
00132   * Returns an interface for accessing various factories capable of creating
00133   * value objects.
00134   * @return An interface for accessing various factories capable of creating
00135   * value objects.
00136   * \endenglish
00137   */
00138  ValueFactoryPtr getValueFactory();
00139
00140  // /**
00141  //  * @return An interface with access to various queryable-only system
00142  //  resources
00143  //  */
00144  // ResourceModel getResourceModel();
00145
00146  /**
00147   * \chinese
00148   * 此方法用于获取指定的 {@link InstallationNodeContribution} 实例。
00149   *
00150   * @param installationType 要返回的安装节点贡献的类。
00151   * @param <T> 用于指定 {@link InstallationNodeContribution} 的泛型。
00152   * @return 安装节点实例。
00153   * \endchinese
00154   * \english
00155   * This method can be used to get a specific {@link
00156   * InstallationNodeContribution} instance.
00157   *
00158   * @param installationType The class of the installation node
00159   * contribution to return.
00160   * @param <T> The generic for specifying the {@link
00161   * InstallationNodeContribution}.
00162   * @return The installation node instance.
00163   * \endenglish
00164   */
00165  template <typename T>
00166  std::shared_ptr<T> getInstallationNode()
00167  {
00168      return std::dynamic_pointer_cast<T>(
00169          getInstallationNode(typeid(T).name()));
00170  }
00171  InstallationNodeContributionPtr getInstallationNode(
00172      const char *typeidOfClass);
00173
00174  /**
00175   * @ingroup ApplicationApi
00176   * @ref DeviceManager
00177   * \chinese
00178   * 此方法用于获取指定的 {@link DeviceManager} 实例。

```

```

00179      *
00180      * @param deviceManagerClass 要返回的设备管理器类。
00181      * @param <T> 要返回的类的泛型类型。
00182      * @return 用于注册和操作特定设备的接口。
00183      * \endchinese
00184      * \english
00185      * This method can be used to get a specific {@link DeviceManager}
00186      * instance.
00187      *
00188      * @param deviceManagerClass The class for the device manager to return.
00189      * @param <T> The generic type of class to return.
00190      * @return An interface for registering and working with a specific
00191      * device.
00192      * \endenglish
00193      */
00194 DeviceManagerPtr getDeviceManager();
00195
00196 /**
00197  * \chinese
00198  * 获取机器人代理，提供与机器人底层 SDK 的交互接口。
00199  * @return 机器人代理指针。
00200  * \endchinese
00201  * \english
00202  * Returns the robot proxy, providing an interaction interface with the
00203  * underlying robot SDK.
00204  * @return The robot proxy pointer.
00205  * \endenglish
00206  */
00207 aubo_sdk::RobotProxyPtr getRobotProxy();
00208
00209 protected:
00210     ApplicationApi();
00211
00212 private:
00213     friend class DataSwitch;
00214     void *d_{ nullptr };
00215 };
00216
00217 } // namespace aubo_scope
00218 } // namespace arcs
00219
00220 #endif // AUBO_SCOPE_APPLICATION_API_H

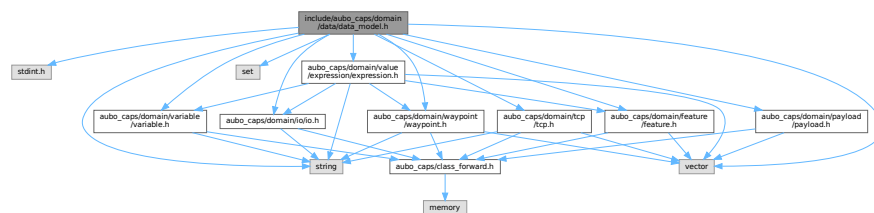
```

12.62 include/aubo_caps/domain/data/data_model.h 文件参考

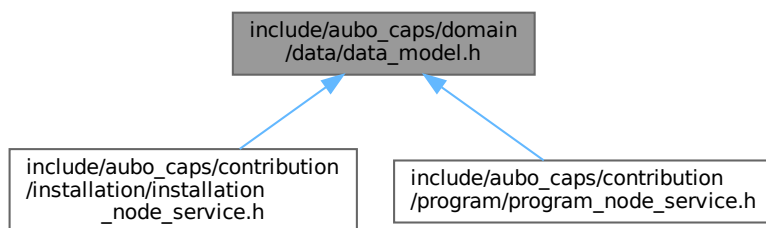
```

#include <stdint.h>
#include <string>
#include <vector>
#include <set>
#include <aubo_caps/domain/feature/feature.h>
#include <aubo_caps/domain/variable/variable.h>
#include <aubo_caps/domain/tcp/tcp.h>
#include <aubo_caps/domain/io/io.h>
#include <aubo_caps/domain/waypoint/waypoint.h>
#include <aubo_caps/domain/payload/payload.h>
#include <aubo_caps/domain/value/expression/expression.h>
data_model.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::DataModel`

数据模型此接口用于存储和管理表示当前配置的数据,例如 *ProgramNodeContribution* 或 *InstallationNodeContribution* 的配置。提供了在字典中添加、删除、获取和修改值的方法。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (DataModel)`

12.63 data_model.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_DATA_MODEL_H
00002 #define AUBO_SCOPE_DATA_MODEL_H
00003
00004 #include <stdint.h>
00005 #include <string>
00006 #include <vector>
00007 #include <set>
00008
00009 #include <aubo_caps/domain/feature/feature.h>
00010 #include <aubo_caps/domain/variable/variable.h>
00011 #include <aubo_caps/domain/tcp/tcp.h>
00012 #include <aubo_caps/domain/io/io.h>
00013 #include <aubo_caps/domain/waypoint/waypoint.h>
00014 #include <aubo_caps/domain/payload/payload.h>
00015 #include <aubo_caps/domain/value/expression/expression.h>
00016
00017 namespace arcs {
00018 namespace aubo_scope {
00019 ARCS_CLASS_FORWARD(DataModel);
00020
00021 /**
00022  * \chinese
00023  * 数据模型
00024  * 此接口用于存储和管理表示当前配置的数据,例如 {@link ProgramNodeContribution}
00025  * 或 {@link InstallationNodeContribution} 的配置。提供了在字典中添加、删除、获取
00026  * 和修改值的方法。
00027  *
00028  * 为已使用的键设置新值将覆盖当前值。无论值的类型如何都会发生(例如,在键
00029  * <i>myBool</i> 下存储值 <code>true</code>,之后又在同一键下存储一个
00030  * <code>Angle</code> 对象,将会用提供的 <code>Angle</code> 对象覆盖
00031  * <code>true</code> 值)。
00032  *
00033  * 每个 AuboCap 安装画面都有一个底层的 <code>DataModel</code> 对象。该对象随每个
00034  * AuboScope 安装一起保存和加载。
00035  *
00036  * 同样,每个贡献的程序节点实例都有一个底层的 <code>DataModel</code>
  
```

```

00037 * 对象。该对象随节点所在程序一起保存和加载。基于 HTML 的程序节点贡献中，对
00038 * <code>DataModel</code> 对象的所有修改都支持撤销/重做操作。基于 Qt 的 AuboStudio
00039 * 插件必须使用 {@link UndoRedoManager} 将更改记录到撤销/重做栈中。
00040 *
00041 * 检索对象时，键和对象类型都必须与之前存储的匹配。这意味着如果使用键
00042 * <i>myAngle</i> 存储了一个 <code>TCP</code> 对象，然后尝试使用键 <i>myAngle</i>
00043 * 通过 {@link #get(std::string key, Angle default_value)} 检索它，将不会返回存储
00044 * 的值，因为类型不匹配。相反，将返回提供的 <code>default_value</code>。
00045 * \endchinese
00046 * \english
00047 * DataModel
00048 * This interface is used for storing and managing data that represents the
00049 * current configuration of, e.g. a
00050 * {@link ProgramNodeContribution} or {@link InstallationNodeContribution}.
00051 * Methods for adding, removing, retrieving and changing values in a dictionary
00052 * are provided.
00053 *
00054 * Setting a new value for a key already in use, will overwrite the current
00055 * value with the new value. This happens regardless of the type of value (e.g.
00056 * storing the value <code>true</code> under the key <i>myBool</i>
00057 * and afterwards storing an <code>Angle</code> object under the same key will
00058 * overwrite the value <code>true</code> with the provided <code>Angle</code>
00059 * object).
00060 *
00061 * A auboCap installation screen has an underlying <code>DataModel</code>
00062 * object. That object is saved and loaded along with each AuboScope
00063 * installation.
00064 *
00065 * Similarly, each contributed program node instance has an underlying
00066 * <code>DataModel</code> object. That object is saved and loaded along with the
00067 * program where the node occurs. Undo/redo actions are supported for all
00068 * modifications to the <code>DataModel</code> object in HTML-based program node
00069 * contributions. Qt-based aubo_studio plugins must use the {@link
00070 * UndoRedoManager} to record the changes on the undo/redo stack.
00071 *
00072 * When retrieving an object, both key and object type must match what was
00073 * previously stored. This means that if a
00074 * <code>TCP</code> object was stored using the key <i>myAngle</i>, then
00075 * attempting to retrieve it using
00076 * {@link #get(std::string key, Angle default_value)} with the key
00077 * <i>myAngle</i> will not return the stored value, since the types do not
00078 * match. Instead the provided <code>default_value</code> will be returned.
00079 * \endenglish
00080 *
00081 */
00082 class ARCS_ABI_EXPORT DataModel
00083 {
00084 public:
00085     DataModel(DataModel &f);
00086     DataModel(DataModel &&f);
00087     ~DataModel();
00088
00089     /**
00090     * \chinese
00091     * 将 <code>bool</code> 值赋给指定键。
00092     *
00093     * @param key 数据模型中的键 (不可为 <code>null</code> 或空 <code>std::string</code>)。
00094     * @param value 赋给键的值。
00095     *
00096     * @throws IllegalArgumentException 如果键为 <code>null</code> 或空 <code>std::string</code>。
00097     * @throws IllegalStateException 如果在 Qt-based AuboCap 程序节点中调用时不在
00098     * {@link UndoableChanges} 范围内 (另见 {@link UndoRedoManager})。
00099     * \endchinese
00100     * \english
00101     * Assign a <code>bool</code> value to the specified key.
00102     *
00103     * @param key key in the data model (not <code>null</code> and not an empty
00104     * <code>std::string</code>).
00105     * @param value value assigned to key.
00106     *
00107     * @throws IllegalArgumentException if the key is <code>null</code> or an
00108     * empty <code>std::string</code>.
00109     * @throws IllegalStateException if called from a Qt-based AuboCap
00110     * program node outside of an {@link UndoableChanges} scope (see also {@link
00111     * UndoRedoManager}).
00112     * \endenglish
00113     */
00114     void set(const std::string &key, bool value);
00115
00116     /**
00117     * \chinese
00118     * 获取赋给指定键的 <code>bool</code> 值。
00119     *
00120     * @param key 数据模型中的键。
00121     * @param default_value 键不存在时返回的值。
00122     * @return 赋给键的值。如果不存在，返回 default_value。
00123     * \endchinese

```

```

00124     * \english
00125     * Get the <code>bool</code> value assigned to the specified key.
00126     *
00127     * @param key key in the data model.
00128     * @param default_value value to be returned, if key does not exist.
00129     * @return the value assigned to the key. If not exist, default_value is
00130     * returned.
00131     * \endenglish
00132     */
00133     bool get(const std::string &key, bool default_value) const;
00134
00135     /**
00136     * \chinese
00137     * 将 <code>int</code> 值赋给指定键。
00138     *
00139     * @param key 数据模型中的键。
00140     * @param value 赋给键的值。
00141     *
00142     * @throws IllegalArgumentException 如果键为 <code>null</code> 或空字符串。
00143     * @throws IllegalStateException 如果不在 {@link UndoableChanges} 范围内调用。
00144     * \endchinese
00145     * \english
00146     * Assign an <code>int</code> value to the specified key.
00147     *
00148     * @param key key in the data model (not <code>null</code> and not an empty
00149     * <code>std::string</code>).
00150     * @param value value assigned to key.
00151     *
00152     * @throws IllegalArgumentException if the key is <code>null</code> or an
00153     * empty <code>std::string</code>.
00154     * @throws IllegalStateException if called from a Qt-based AudoCap
00155     * program node outside of an {@link UndoableChanges} scope (see also {@link
00156     * UndoRedoManager}).
00157     * \endenglish
00158     */
00159     void set(const std::string &key, int value);
00160
00161     /**
00162     * \chinese
00163     * 获取赋给指定键的 <code>int</code> 值。
00164     *
00165     * @param key 数据模型中的键。
00166     * @param default_value 键不存在时返回的值。
00167     * @return 赋给键的值。
00168     * \endchinese
00169     * \english
00170     * Get the <code>int</code> value assigned to the specified key.
00171     *
00172     * @param key key in the data model.
00173     * @param default_value value to be returned, if key does not exist.
00174     * @return the value assigned to the key. If not exist, default_value is
00175     * returned.
00176     * \endenglish
00177     */
00178     int get(const std::string &key, int default_value) const;
00179
00180     /**
00181     * \chinese
00182     * 将 <code>long</code> 值赋给指定键。
00183     * \endchinese
00184     * \english
00185     * Assign a <code>long</code> value to the specified key.
00186     *
00187     * @param key key in the data model (not <code>null</code> and not an empty
00188     * <code>std::string</code>).
00189     * @param value value assigned to key.
00190     *
00191     * @throws IllegalArgumentException if the key is <code>null</code> or an
00192     * empty <code>std::string</code>.
00193     * @throws IllegalStateException if called from a Qt-based AudoCap
00194     * program node outside of an {@link UndoableChanges} scope (see also {@link
00195     * UndoRedoManager}).
00196     * \endenglish
00197     */
00198     void set(const std::string &key, long value);
00199     void set(const std::string &key, uint64_t value);
00200
00201     /**
00202     * \chinese
00203     * 获取赋给指定键的 <code>long</code> 值。
00204     * \endchinese
00205     * \english
00206     * Get the <code>long</code> value assigned to the specified key.
00207     *
00208     * @param key key in the data model.
00209     * @param default_value value to be returned, if key does not exist.
00210     * @return the value assigned to the key. If not exist, default_value is

```



```

00211     * returned.
00212     * \endenglish
00213     */
00214 long get(const std::string &key, long default_value) const;
00215 uint64_t get(const std::string &key, uint64_t default_value) const;
00216
00217 /**
00218  * \chinese
00219  * 将 <code>float</code> 值赋给指定键。
00220  * \endchinese
00221  * \english
00222  * Assign a <code>float</code> value to the specified key.
00223  *
00224  * @param key key in the data model (not <code>null</code> and not an empty
00225  * <code>std::string</code>).
00226  * @param value value assigned to key.
00227  *
00228  * @throws IllegalArgumentException if the key is <code>null</code> or an
00229  * empty <code>std::string</code>.
00230  * @throws IllegalStateException if called from a Qt-based AudoCap
00231  * program node outside of an {@link UndoableChanges} scope (see also {@link
00232  * UndoRedoManager}).
00233  * \endenglish
00234  */
00235 void set(const std::string &key, float value);
00236
00237 /**
00238  * \chinese
00239  * 获取赋给指定键的 <code>float</code> 值。
00240  * \endchinese
00241  * \english
00242  * Get the <code>float</code> value assigned to the specified key.
00243  *
00244  * @param key key in the data model.
00245  * @param default_value value to be returned, if key does not exist.
00246  * @return the value assigned to the key. If not exist, default_value is
00247  * returned.
00248  * \endenglish
00249  */
00250 float get(const std::string &key, float default_value) const;
00251
00252 /**
00253  * \chinese
00254  * 将 <code>double</code> 值赋给指定键。
00255  * \endchinese
00256  * \english
00257  * Assign a <code>double</code> value to the specified key.
00258  *
00259  * @param key key in the data model (not <code>null</code> and not an empty
00260  * <code>std::string</code>).
00261  * @param value value assigned to key.
00262  *
00263  * @throws IllegalArgumentException if the key is <code>null</code> or an
00264  * empty <code>std::string</code>.
00265  * @throws IllegalStateException if called from a Qt-based AudoCap
00266  * program node outside of an {@link UndoableChanges} scope (see also {@link
00267  * UndoRedoManager}).
00268  * \endenglish
00269  */
00270 void set(const std::string &key, double value);
00271
00272 /**
00273  * \chinese
00274  * 获取赋给指定键的 <code>double</code> 值。
00275  * \endchinese
00276  * \english
00277  * Get the <code>double</code> value assigned to the specified key.
00278  *
00279  * @param key key in the data model.
00280  * @param default_value value to be returned, if key does not exist.
00281  * @return the value assigned to the key. If not exist, default_value is
00282  * returned.
00283  * \endenglish
00284  */
00285 double get(const std::string &key, double default_value) const;
00286
00287 /**
00288  * \chinese
00289  * 将 <code>std::string</code> 值赋给指定键。
00290  * \endchinese
00291  * \english
00292  * Assign a <code>std::string</code> value to the specified key.
00293  *
00294  * @param key key in the data model (not <code>null</code> and not an empty
00295  * <code>std::string</code>).
00296  * @param value value assigned to key.
00297  *

```

```

00298     * @throws IllegalArgumentException if the key is <code>null</code> or an
00299     * empty <code>std::string</code>.
00300     * @throws IllegalStateException if called from a Qt-based AudoCap
00301     * program node outside of an {@link UndoableChanges} scope (see also {@link
00302     * UndoRedoManager}).
00303     * \endenglish
00304     */
00305 void set(const std::string &key, const std::string &value);
00306
00307 /**
00308     * \chinese
00309     * 获取赋给指定键的 <code>std::string</code> 值。
00310     * \endchinese
00311     * \english
00312     * Get the <code>std::string</code> value assigned to the specified key.
00313     *
00314     * @param key key in the data model.
00315     * @param default_value value to be returned, if key does not exist.
00316     * @return the value assigned to the key. If not exist, default_value is
00317     * returned.
00318     * \endenglish
00319     */
00320 std::string get(const std::string &key,
00321                const std::string &default_value) const;
00322
00323 /**
00324     * \chinese
00325     * 将 <code>Variable</code> 值赋给指定键。
00326     * \endchinese
00327     * \english
00328     * Assign a <code>Variable</code> value to the specified key.
00329     *
00330     * @param key key in the data model (not <code>null</code> and not an empty
00331     * <code>std::string</code>).
00332     * @param value value assigned to key.
00333     *
00334     * @throws IllegalArgumentException if the key is <code>null</code> or an
00335     * empty <code>std::string</code>.
00336     * @throws IllegalStateException if called from a Qt-based AudoCap
00337     * program node outside of an {@link UndoableChanges} scope (see also {@link
00338     * UndoRedoManager}).
00339     * \endenglish
00340     */
00341 void set(const std::string &key, VariablePtr value);
00342 void set(const std::string &key, ExpressionPtr value);
00343 void set(const std::string &key, PayloadPtr value);
00344 void set(const std::string &key, IoPtr value);
00345 void set(const std::string &key, WaypointPtr value);
00346
00347 /**
00348     * \chinese
00349     * 获取赋给指定键的 <code>Variable</code> 值。
00350     * @return 赋给键的值。
00351     * \endchinese
00352     * \english
00353     * Get the <code>Variable</code> value assigned to the specified key.
00354     *
00355     * @param key key in the data model.
00356     * @param default_value value to be returned, if key does not exist.
00357     * @return the value assigned to the key. If not exist, default_value is
00358     * returned.
00359     * \endenglish
00360     */
00361 VariablePtr get(const std::string &key, VariablePtr default_value) const;
00362 ExpressionPtr get(const std::string &key,
00363                  ExpressionPtr default_value) const;
00364 PayloadPtr get(const std::string &key, PayloadPtr default_value) const;
00365 IoPtr get(const std::string &key, IoPtr default_value) const;
00366 WaypointPtr get(const std::string &key, WaypointPtr default_value) const;
00367
00368 /**
00369     * \chinese
00370     * 将 <code>bool[]</code> 数组赋给指定键。
00371     * \endchinese
00372     * \english
00373     * Assign a <code>bool[]</code> as value to the specified key.
00374     * @param key key in the data model (not <code>null</code>) and not an empty
00375     * <code>std::string</code>.
00376     * @param value value assigned to key.
00377     *
00378     * @throws IllegalArgumentException if the key is <code>null</code> or an
00379     * empty <code>std::string</code>.
00380     * @throws IllegalStateException if called from a Qt-based AudoCap
00381     * program node outside of an {@link UndoableChanges} scope (see also {@link
00382     * UndoRedoManager}).
00383     * \endenglish
00384     */
00385 void set(const std::string &key, const std::vector<bool> &value);

```

```

00385
00386 /**
00387  * \chinese
00388  * 获取赋给指定键的 <code>bool[]</code> 数组。
00389  * \endchinese
00390  * \english
00391  * Get the <code>bool[]</code> as value assigned to the specified key.
00392  *
00393  * @param key key in the data model.
00394  * @param default_value value to be returned, if key does not exist.
00395  * @return the value assigned to the key. If not exist, default_value is
00396  * returned.
00397  * \endenglish
00398  */
00399 std::vector<bool> get(const std::string &key,
00400                     const std::vector<bool> &default_value) const;
00401
00402 /**
00403  * \chinese
00404  * 将 <code>int[]</code> 数组赋给指定键。
00405  * \endchinese
00406  * \english
00407  * Assign a <code>int[]</code> as value to the specified key.
00408  *
00409  * @param key key in the data model (not <code>null</code> and not an empty
00410  * <code>std::string</code>).
00411  * @param value value assigned to key.
00412  *
00413  * @throws IllegalArgumentException if the key is <code>null</code> or an
00414  * empty <code>std::string</code>.
00415  * @throws IllegalStateException if called from a Qt-based AudoCap
00416  * program node outside of an {@link UndoableChanges} scope (see also {@link
00417  * UndoRedoManager}).
00418  * \endenglish
00419  */
00420 void set(const std::string &key, const std::vector<int> &value);
00421
00422 /**
00423  * \chinese
00424  * 获取赋给指定键的 <code>int[]</code> 数组。
00425  * \endchinese
00426  * \english
00427  * Get the <code>int[]</code> as value assigned to the specified key.
00428  *
00429  * @param key key in the data model.
00430  * @param default_value value to be returned, if key does not exist.
00431  * @return the value assigned to the key. If not exist, default_value is
00432  * returned.
00433  * \endenglish
00434  */
00435 std::vector<int> get(const std::string &key,
00436                    const std::vector<int> &default_value) const;
00437
00438 /**
00439  * \chinese
00440  * 将 <code>long[]</code> 数组赋给指定键。
00441  * \endchinese
00442  * \english
00443  * Assign a <code>long[]</code> as value to the specified key.
00444  *
00445  * @param key key in the data model (not <code>null</code> and not an empty
00446  * <code>std::string</code>).
00447  * @param value value assigned to key.
00448  *
00449  * @throws IllegalArgumentException if the key is <code>null</code> or an empty
00450  * <code>std::string</code>.
00451  * @throws IllegalStateException if called from a Qt-based AudoCap
00452  * program node outside of an {@link UndoableChanges} scope (see also {@link
00453  * UndoRedoManager}).
00454  * \endenglish
00455  */
00456 void set(const std::string &key, const std::vector<long> &value);
00457 void set(const std::string &key, const std::vector<uint64_t> &value);
00458
00459 /**
00460  * \chinese
00461  * 获取赋给指定键的 <code>long[]</code> 数组。
00462  * \endchinese
00463  * \english
00464  * Get the <code>long[]</code> as value assigned to the specified key.
00465  *
00466  * @param key key in the data model.
00467  * @param default_value value to be returned, if key does not exist.
00468  * @return the value assigned to the key. If not exist, default_value is
00469  * returned.
00470  * \endenglish
00471  */

```

```

00472 std::vector<long> get(const std::string &key,
00473                     const std::vector<long> &default_value) const;
00474 std::vector<uint64_t> get(const std::string &key,
00475                          const std::vector<uint64_t> &default_value) const;
00476
00477 /**
00478  * \chinese
00479  * 将 <code>float[]</code> 数组赋给指定键。
00480  * \endchinese
00481  * \english
00482  * Assign a <code>float[]</code> as value to the specified key.
00483  *
00484  * @param key key in the data model (not <code>null</code> and not an empty
00485  * <code>std::string</code>).
00486  * @param value value assigned to key.
00487  *
00488  * @throws IllegalArgumentException if the key is <code>null</code> or an
00489  * empty <code>std::string</code>.
00490  * @throws IllegalStateException if called from a Qt-based AudoCap
00491  * program node outside of an {@link UndoableChanges} scope (see also {@link
00492  * UndoRedoManager}).
00493  * \endenglish
00494  */
00495 void set(const std::string &key, const std::vector<float> &value);
00496
00497 /**
00498  * \chinese
00499  * 获取赋给指定键的 <code>float[]</code> 数组。
00500  * \endchinese
00501  * \english
00502  * Get the <code>float[]</code> as value assigned to the specified key.
00503  *
00504  * @param key key in the data model.
00505  * @param default_value value to be returned, if key does not exist.
00506  * @return the value assigned to the key. If not exist, default_value is
00507  * returned.
00508  * \endenglish
00509  */
00510 std::vector<float> get(const std::string &key,
00511                     const std::vector<float> &default_value) const;
00512
00513 /**
00514  * \chinese
00515  * 将 <code>double[]</code> 数组赋给指定键。
00516  * \endchinese
00517  * \english
00518  * Assign a <code>double[]</code> as value to the specified key.
00519  *
00520  * @param key key in the data model (not <code>null</code> and not an empty
00521  * <code>std::string</code>).
00522  * @param value value assigned to key.
00523  *
00524  * @throws IllegalArgumentException if the key is <code>null</code> or an
00525  * empty <code>std::string</code>.
00526  * @throws IllegalStateException if called from a Qt-based AudoCap
00527  * program node outside of an {@link UndoableChanges} scope (see also {@link
00528  * UndoRedoManager}).
00529  * \endenglish
00530  */
00531 void set(const std::string &key, const std::vector<double> &value);
00532
00533 /**
00534  * \chinese
00535  * 获取赋给指定键的 <code>double[]</code> 数组。
00536  * \endchinese
00537  * \english
00538  * Get the <code>double[]</code> as value assigned to the specified key.
00539  *
00540  * @param key key in the data model.
00541  * @param default_value value to be returned, if key does not exist.
00542  * @return the value assigned to the key. If not exist, default_value is
00543  * returned.
00544  * \endenglish
00545  */
00546 std::vector<double> get(const std::string &key,
00547                      const std::vector<double> &default_value) const;
00548
00549 /**
00550  * \chinese
00551  * 将 <code>std::string[]</code> 数组赋给指定键。
00552  * \endchinese
00553  * \english
00554  * Assign a <code>std::string[]</code> as value to the specified key.
00555  *
00556  * @param key key in the data model (not <code>null</code> and not an empty
00557  * <code>std::string</code>).
00558  * @param value value assigned to key.

```

```

00559      *
00560      * @throws IllegalArgumentException if the key is <code>null</code> or an
00561      * empty <code>std::string</code>.
00562      * @throws IllegalStateException if called from a Qt-based AudoCap
00563      * program node outside of an {@link UndoableChanges} scope (see also {@link
00564      * UndoRedoManager}).
00565      * \endenglish
00566      */
00567      void set(const std::string &key, const std::vector<std::string> &value);
00568
00569      /**
00570      * \chinese
00571      * 获取赋给指定键的 <code>std::string[]</code> 数组。
00572      * \endchinese
00573      * \english
00574      * Get the <code>std::string[]</code> as value assigned to the specified
00575      * key.
00576      *
00577      * @param key key in the data model.
00578      * @param default_value value to be returned, if key does not exist.
00579      * @return the value assigned to the key. If not exist, default_value is
00580      * returned.
00581      * \endenglish
00582      */
00583      std::vector<std::string> get(
00584          const std::string &key,
00585          const std::vector<std::string> &default_value) const;
00586
00587      /**
00588      * \chinese
00589      * 将 <code>TCP</code> 值赋给指定键。
00590      * \endchinese
00591      * \english
00592      * Assign a <code>TCP</code> value to the specified key.
00593      *
00594      * @param key key in the data model (not <code>null</code> and not an empty
00595      * <code>std::string</code>).
00596      * @param value value assigned to key.
00597      *
00598      * @throws IllegalArgumentException if the key is <code>null</code> or an
00599      * empty <code>std::string</code>.
00600      * @throws IllegalStateException if called from a Qt-based AudoCap
00601      * program node outside of an {@link UndoableChanges} scope (see also {@link
00602      * UndoRedoManager}).
00603      * \endenglish
00604      */
00605      void set(const std::string &key, TCPPtr value);
00606
00607      /**
00608      * \chinese
00609      * 获取赋给指定键的 <code>TCP</code> 值。
00610      * \endchinese
00611      * \english
00612      * Get the <code>TCP</code> value assigned to the specified key.
00613      *
00614      * @param key key in the data model.
00615      * @param default_value value to be returned, if key does not exist.
00616      * @return the value assigned to the key. If not exist, default_value is
00617      * returned.
00618      * \endenglish
00619      */
00620      TCPPtr get(const std::string &key, TCPPtr default_value) const;
00621
00622      /**
00623      * \chinese
00624      * 将 <code>Feature</code> 值赋给指定键。
00625      * \endchinese
00626      * \english
00627      * Assign a <code>Feature</code> value to the specified key.
00628      *
00629      * @param key key in the data model (not <code>null</code> and not an empty
00630      * <code>std::string</code>).
00631      * @param value value assigned to key.
00632      *
00633      * @throws IllegalArgumentException if the key is <code>null</code> or an
00634      * empty <code>std::string</code>.
00635      * @throws IllegalStateException if called from a Qt-based AudoCap
00636      * program node outside of an {@link UndoableChanges} scope (see also {@link
00637      * UndoRedoManager}).
00638      * \endenglish
00639      */
00640      void set(const std::string &key, FeaturePtr value);
00641
00642      /**
00643      * \chinese
00644      * 获取赋给指定键的 <code>Feature</code> 值。
00645      * \endchinese

```

```

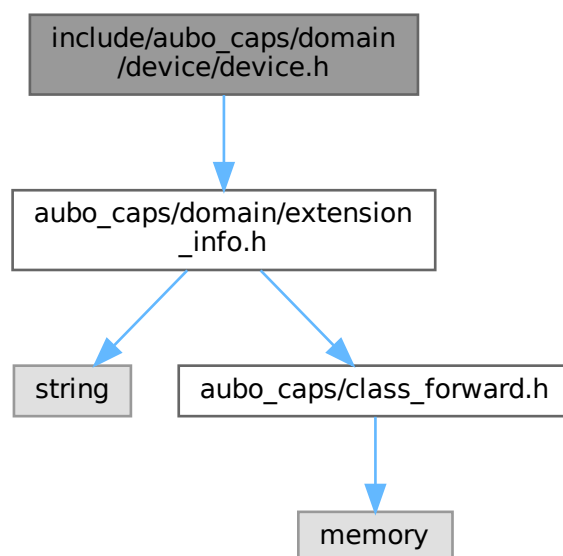
00646     * \english
00647     * Get the <code>Feature</code> value assigned to the specified key.
00648     *
00649     * @param key key in the data model.
00650     * @param default_value value to be returned, if key does not exist.
00651     * @return the value assigned to the key. If not exist, default_value is
00652     * returned.
00653     * \endenglish
00654     */
00655     FeaturePtr get(const std::string &key, FeaturePtr default_value) const;
00656
00657     /**
00658     * \chinese
00659     * 获取数据模型中所有键的集合。
00660     * @return 键的集合。
00661     * \endchinese
00662     * \english
00663     * Get a set of all the keys in the data model.
00664     *
00665     * @return A Set of keys.
00666     * \endenglish
00667     */
00668     std::set<std::string> getKeys() const;
00669
00670     /**
00671     * \chinese
00672     * 检查数据模型中是否存在指定键。
00673     * @param key 数据模型中的键。
00674     * @return 如果键存在返回 <code>true</code>, 否则返回 <code>false</code>。
00675     * \endchinese
00676     * \english
00677     * Check if a key is present in the data model.
00678     *
00679     * @param key key in the data model (not <code>null</code> and not an empty
00680     * <code>std::string</code>).
00681     * @return <code>true</code>, if key exist, otherwise <code>false</code>.
00682     * \endenglish
00683     */
00684     bool isSet(const std::string &key) const;
00685
00686     /**
00687     * \chinese
00688     * 从数据模型中移除一个键值对。
00689     * @param key 数据模型中的键。
00690     * @return 成功返回 <code>true</code>, 否则返回 <code>false</code>。
00691     * @throws IllegalStateException 如果在 Qt-based AudoCap 程序节点中调用时不在
00692     * {@link UndoableChanges} 范围内。
00693     * \endchinese
00694     * \english
00695     * Remove a key-value pair from the data model.
00696     *
00697     * @param key key in the data model (not <code>null</code> and not an empty
00698     * <code>std::string</code>).
00699     * @return <code>true</code>, if succeed, otherwise <code>false</code>.
00700     * @throws IllegalStateException if called from a Qt-based AudoCap
00701     * program node outside of an {@link UndoableChanges} scope (see also {@link
00702     * UndoRedoManager}).
00703     * \endenglish
00704     */
00705     bool remove(const std::string &key);
00706
00707 private:
00708     friend class DataSwitch;
00709     DataModel();
00710     void *d_{ nullptr };
00711 };
00712
00713 } // namespace aubo_scope
00714 } // namespace arcs
00715
00716 #endif

```

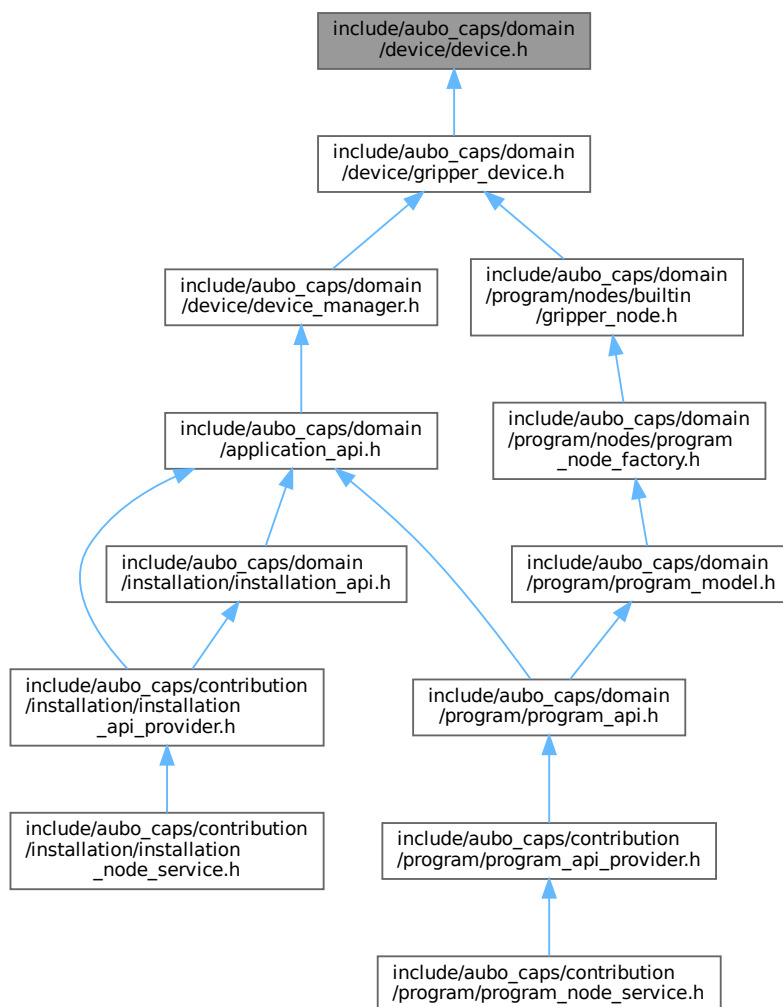
12.64 include/aubo_caps/domain/device/device.h 文件参考

```
#include <aubo_caps/domain/extension_info.h>
```

device.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class `arcs::aubo_scope::Device`
设备接口此基础接口表示 *AuboScope* 支持的设备，例如夹爪设备。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (Device)`

12.65 device.h

[浏览该文件的文档.](#)

```

00001 #include <aubo_caps/domain/extension_info.h>
00002

```



```

00003 namespace arcs {
00004 namespace aubo_scope {
00005 ARCS_CLASS_FORWARD(Device);
00006
00007 /**
00008  * \chinese
00009  * 设备接口
00010  * 此基础接口表示 AuboScope 支持的设备，例如夹爪设备。
00011  * \endchinese
00012  * \english
00013  * Device
00014  * This base interface represents a device supported by AuboScope, e.g. a
00015  * gripping device.
00016  * \endenglish
00017 */
00018 class ARCS_ABI_EXPORT Device
00019 {
00020 public:
00021     virtual ~Device() = 0;
00022     /**
00023      * \chinese
00024      * 获取将此夹爪设备添加/贡献到 AuboScope 的 AuboCap 信息。
00025      * @return 此夹爪设备的 AuboCap 信息。
00026      * \endchinese
00027      * \english
00028      * Get information about the AuboCap that added/contributed this gripper
00029      * device to AuboScope.
00030      *
00031      * @return AuboCap information for this gripper device.
00032      * \endenglish
00033      */
00034     virtual AuboCapInfoPtr getContributorInfo() = 0;
00035
00036     /**
00037      * \chinese
00038      * <p>
00039      * 获取此夹爪的唯一标识符 (id)。可用于在特定多夹爪设备中区分不同的夹爪。
00040      * </p>
00041      * 注意此 id 并非"全局"唯一，仅对属于特定多夹爪设备的各个夹爪唯一。
00042      * @return 此夹爪的唯一标识符。
00043      * \endchinese
00044      * \english
00045      * <p>
00046      * Get the unique identifier (id) for this gripper. This can be used to
00047      * identify this specific gripper among the set of individual grippers
00048      * available in a specific multi-gripper device.
00049      * </p>
00050      *
00051      * Note that this id is not "globally" unique, but only unique for the
00052      * individual grippers belonging to a specific multi-gripper device.
00053      *
00054      * @return The unique identifier of this gripper.
00055      * \endenglish
00056      */
00057     virtual std::string getId() = 0;
00058
00059     /**
00060      * \chinese
00061      * <p>
00062      * AuboScope UI 中显示的夹爪名称，例如在夹爪工具栏和程序节点中。
00063      * </p>
00064      * <b>注意：</b> 此名称不应用于唯一标识特定的单个夹爪，因为名称可能会被翻译
00065      * （取决于 AuboScope 中选择的语言）或在添加夹爪的 AuboCap 版本之间发生更改。
00066      * @return 此夹爪在 AuboScope 中显示的名称。
00067      * \endchinese
00068      * \english
00069      * <p>
00070      * The name of the gripper displayed in the AuboScope UI, e.g. in the
00071      * Gripper toolbar and program node.
00072      * </p>
00073      *
00074      * <b>Note:</b> This name must not be used to uniquely identify a specific
00075      * individual gripper, since the name might be translated (depending on the
00076      * selected language in AuboScope) or be changed between versions of the
00077      * AuboCap that added the gripper.
00078      *
00079      * @return The name of this gripper displayed in AuboScope.
00080      * \endenglish
00081      */
00082     virtual std::string getDisplayName() = 0;
00083
00084     /**
00085      * \chinese
00086      * 设备无法保证始终存在于 AuboScope 中。如果添加设备的 AuboCap 未安装，设备将不可用。
00087      * 此方法用于判断设备是否可用。
00088      * @return 如果此夹爪设备存在于 AuboScope 中返回 <code>true</code>，否则返回
00089      * <code>false</code>。

```

```

00090      * \endchinese
00091      * \english
00092      * A device cannot be guaranteed to be present in AboScope. A device will
00093      * not be present if the AboCap that added the device is not installed.
00094      * This method can be used to determine, if the device is present.
00095      *
00096      * @return <code>true</code> if this gripper device is present in AboScope,
00097      * otherwise <code>false</code>.
00098      * \endenglish
00099      */
00100     virtual bool isResolvable() = 0;
00101 };
00102 } // namespace aubo_scope
00103 } // namespace arcs

```

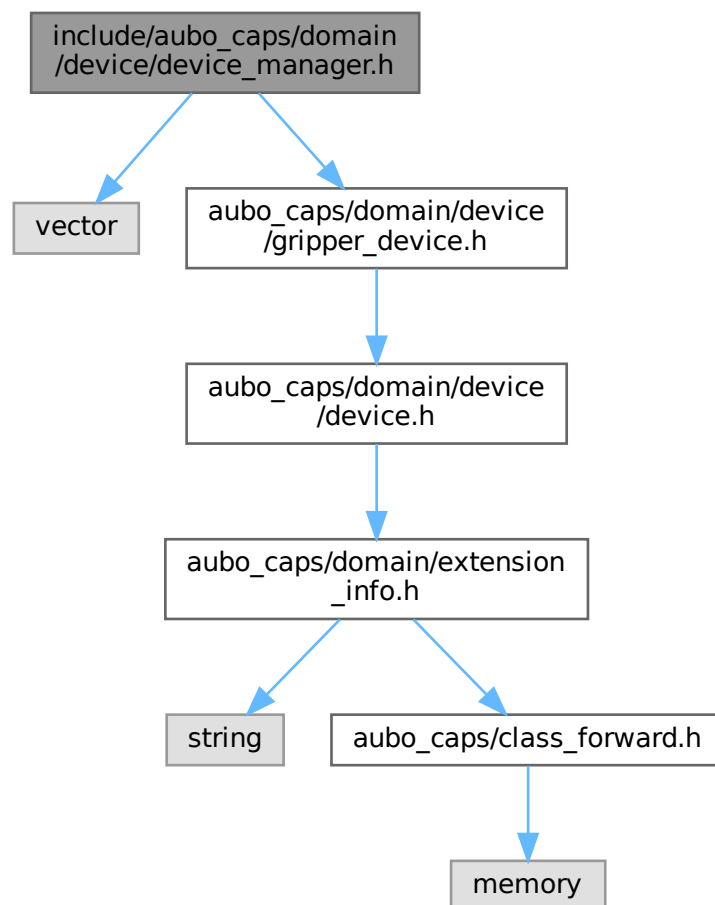
12.66 include/auo_caps/domain/device/device_manager.h 文件参考

```

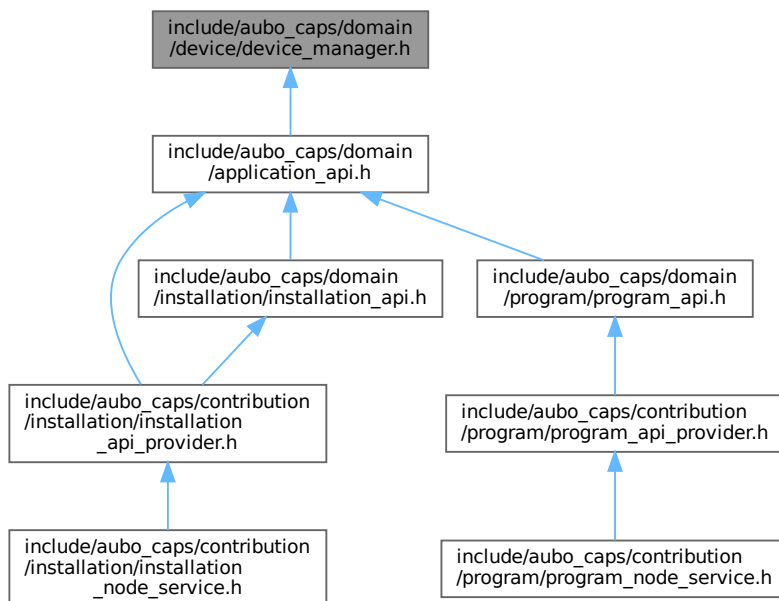
#include <vector>
#include <auo_caps/domain/device/gripper_device.h>

```

device_manager.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class [arcs::aubo_scope::DeviceManager](#)
设备管理器设备管理器的基础接口。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(DeviceManager\)](#)

12.67 device_manager.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_DEVICE_MANAGER_H
00002 #define AUBO_SCOPE_DEVICE_MANAGER_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/device/gripper_device.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(DeviceManager);
00010
00011 /**
00012  * @ingroup ApplicationApi
00013  * \chinese
00014  * 设备管理器
00015  * 设备管理器的基础接口。
00016  *
00017  * 设备管理器为 AuboScope 支持的特定类型设备提供功能。例如，设备管理器可以提供访问
00018  * 所有已注册设备（该类型）以及在程序中使用特定设备的功能。
00019  * \endchinese
00020  * \english
00021  * DeviceManager
  
```


函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(DeviceRegistrationManager\)](#)

12.69 device_registration_manager.h

[浏览该文件的文档.](#)

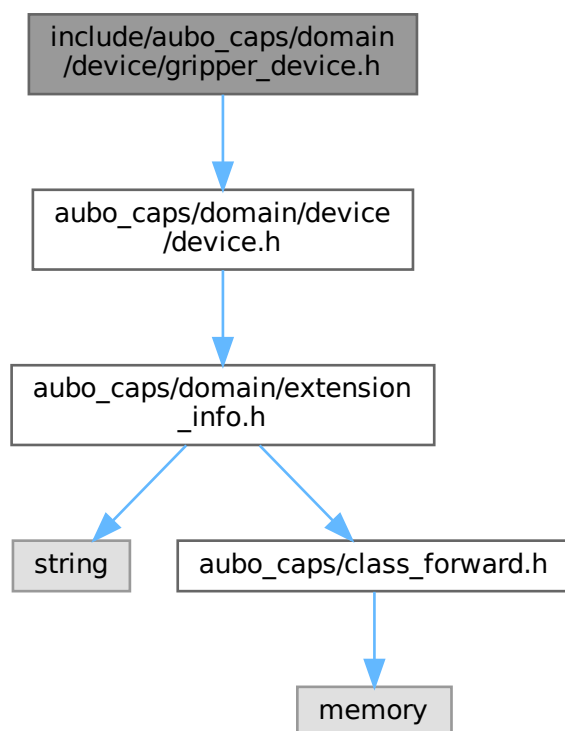
```

00001 #ifndef AUBO_SCOPE_DEVICE_REGISTRATION_MANAGER_H
00002 #define AUBO_SCOPE_DEVICE_REGISTRATION_MANAGER_H
00003
00004 #include <aubo_caps/contribution/driver/gripper/gripper_contribution.h>
00005 #include <aubo_caps/contribution/driver/screw_driver/screw_driver_contribution.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(DeviceRegistrationManager);
00010
00011 /**
00012  * \chinese
00013  * 设备注册管理器
00014  * 设备注册管理器的基础接口。
00015  *
00016  * 设备注册管理器用于将程序/安装节点贡献或服务注册为符合 AuboScope 支持的特定设备类型。
00017  * \endchinese
00018  * \english
00019  * DeviceRegistrationManager
00020  * Base interface for device registration managers.
00021  *
00022  * A device registration manager is used for registering a program/installation
00023  * node contribution/service as conformant with a specific device type supported
00024  * by AuboScope.
00025  * \endenglish
00026  */
00027 class ARCS_ABI_EXPORT DeviceRegistrationManager
00028 {
00029 public:
00030     virtual ~DeviceRegistrationManager() = default;
00031
00032     /**
00033      * \chinese
00034      * 注册夹爪贡献
00035      * @param contribution 夹爪贡献指针。
00036      * \endchinese
00037      * \english
00038      * Register a gripper contribution.
00039      * @param contribution the gripper contribution to register.
00040      * \endenglish
00041      */
00042     virtual void registerGripperContribution(
00043         GripperContributionPtr contribution) = 0;
00044
00045     /**
00046      * \chinese
00047      * 注册螺丝刀贡献
00048      * @param contribution 螺丝刀贡献指针。
00049      * \endchinese
00050      * \english
00051      * Register a screwdriver contribution.
00052      * @param contribution the screwdriver contribution to register.
00053      * \endenglish
00054      */
00055     virtual void registerScrewDriverContribution(
00056         ScrewdriverContributionPtr contribution) = 0;
00057 };
00058 } // namespace aubo_scope
00059 } // namespace arcs
00060
00061 #endif

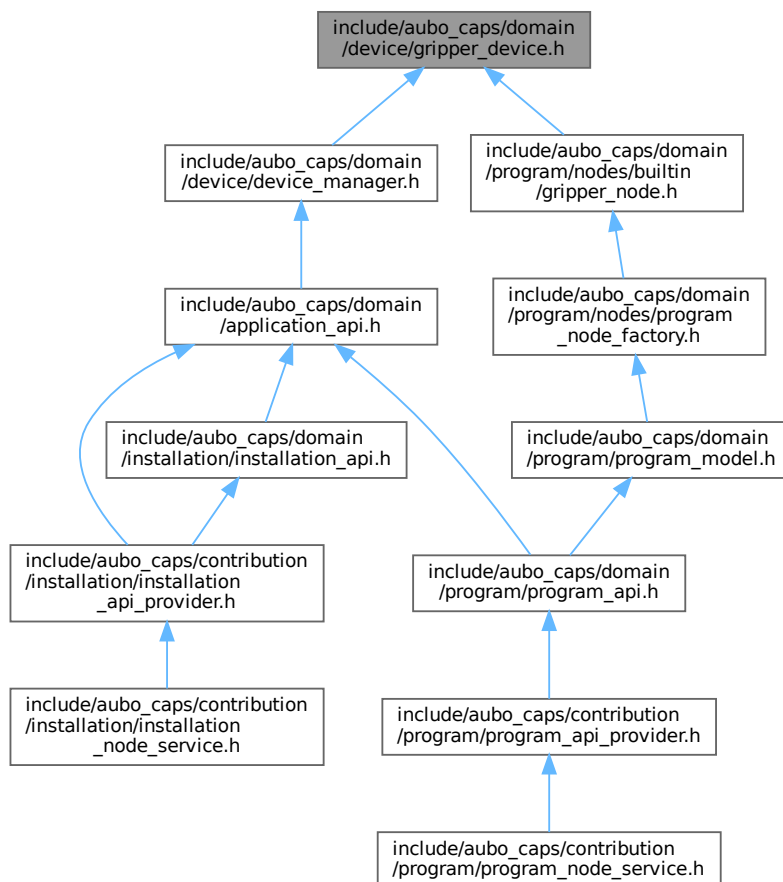
```

12.70 include/aubo_caps/domain/device/gripper_device.h 文件参考

`#include <aubo_caps/domain/device/device.h>`
gripper_device.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class `arcs::aubo_scope::GripperDevice`
夹爪设备此接口表示 *AuboScope* 中的夹爪设备。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (GripperDevice)`

12.71 gripper_device.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_GRIPPER_DEVICE_H
00002 #define AUBO_SCOPE_GRIPPER_DEVICE_H
00003
00004 #include <aubo_caps/domain/device/device.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(GripperDevice);

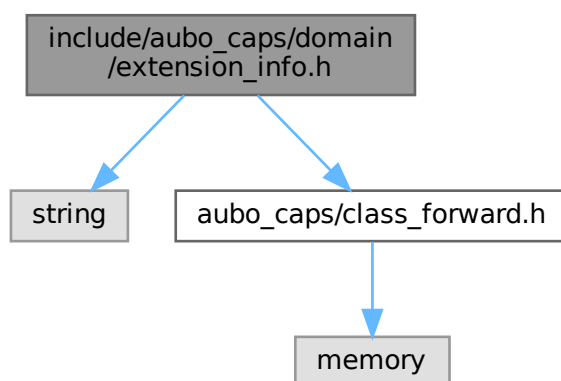
```

```
00009
00010 /**
00011  * \chinese
00012  * 夹爪设备
00013  * 此接口表示 AuboScope 中的夹爪设备。
00014  * \endchinese
00015  * \english
00016  * GripperDevice
00017  * This interface represents a gripper device in AuboScope.
00018  * \endenglish
00019 */
00020 class ARCS_ABI_EXPORT GripperDevice : public Device
00021 {
00022 public:
00023     virtual ~GripperDevice() = default;
00024 };
00025 } // namespace aubo_scope
00026 } // namespace arcs
00027
00028 #endif
```

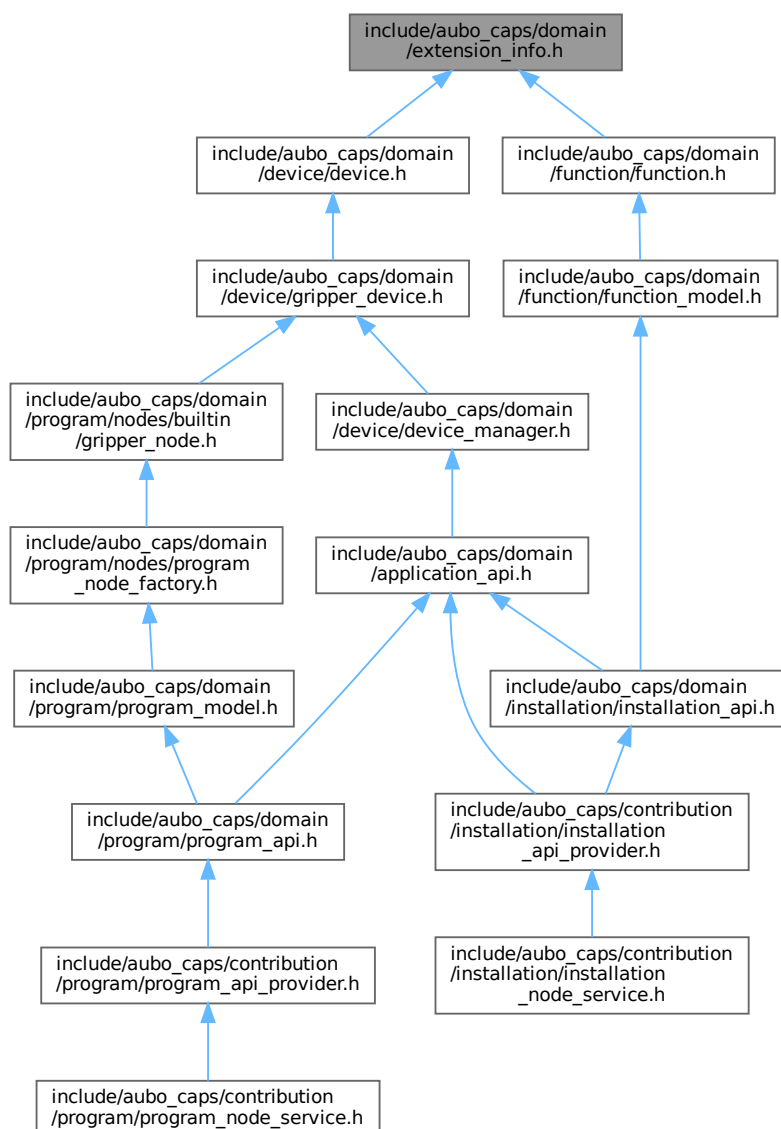
12.72 include/aubo_caps/domain/extension_info.h 文件参考

```
#include <string>
#include <aubo_caps/class_forward.h>
```

extension_info.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::AuboCapInfo](#)

This interface provides various metadata information about an installed AuboCap, such as the name of the AuboCap.

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(AuboCapInfo\)](#)

12.73 extension_info.h

[浏览该文件的文档.](#)

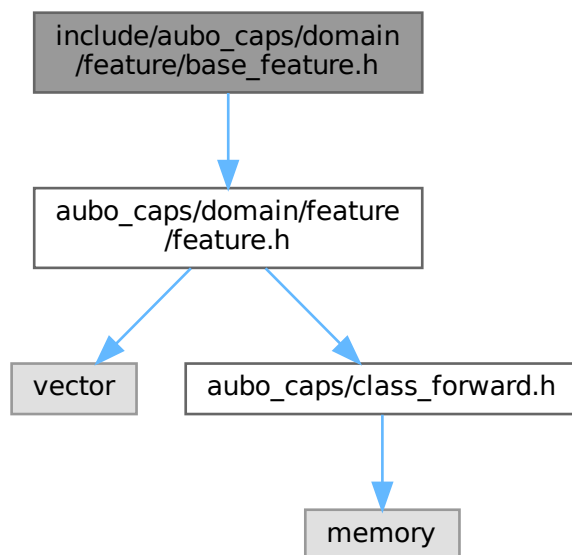
```

00001 #ifndef AUBO_SCOPE_EXTENSION_INFO_H
00002 #define AUBO_SCOPE_EXTENSION_INFO_H
00003
00004 #include <string>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(AuboCapInfo);
00010
00011 /**
00012  * This interface provides various metadata information about an installed
00013  * AuboCap, such as the name of the AuboCap.
00014  *
00015  * @since 1.1.0
00016  */
00017 class ARCS_ABI_EXPORT AuboCapInfo
00018 {
00019 public:
00020     virtual ~AuboCapInfo() = default;
00021
00022     /**
00023      * @retAubon The symbolic name (id) of the AuboCap.
00024      */
00025     virtual std::string getSymbolicName() = 0;
00026
00027     /**
00028      * @retAubon The name of the AuboCap.
00029      */
00030     virtual std::string getName() = 0;
00031
00032     /**
00033      * @retAubon The vendor of the AuboCap.
00034      */
00035     virtual std::string getVendor() = 0;
00036 };
00037 } // namespace aubo_scope
00038 } // namespace arcs
00039
00040 #endif

```

12.74 include/aubo_caps/domain/feature/base_feature.h 文件参考

`#include <aubo_caps/domain/feature/feature.h>`
base_feature.h 的引用 (Include) 关系图:



类

- class [arcs::aubo_scope::BaseFeature](#)
This interface represents the Base feature in AuboScope.

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(BaseFeature\)](#)

12.75 base_feature.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_BASE_FEATURE_H
00002 #define AUBO_SCOPE_BASE_FEATURE_H
00003
00004 #include <aubo_caps/domain/feature/feature.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(BaseFeature);
00009
00010         /**
00011          * This interface represents the Base feature in AuboScope.
00012          */
00013         class ARCS_ABI_EXPORT BaseFeature : public Feature
  
```

```

00014 {
00015 public:
00016     BaseFeature(BaseFeature &f);
00017     BaseFeature(BaseFeature &&f);
00018     virtual ~BaseFeature();
00019
00020 private:
00021     friend class DataSwitch;
00022     BaseFeature();
00023     void *d_{ nullptr };
00024 };
00025
00026 } // namespace aubo_scope
00027 } // namespace arcs
00028
00029 #endif // AUBO_SCOPE_BASE_FEATURE_H

```

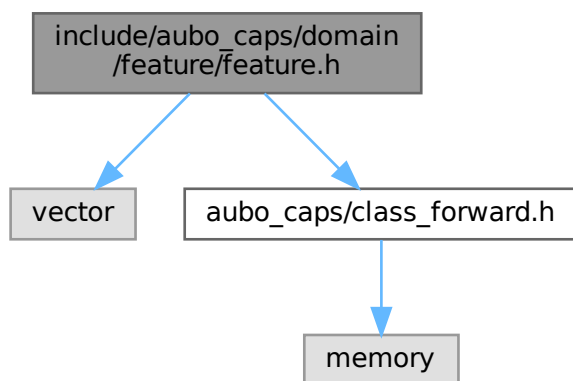
12.76 include/aubo_caps/domain/feature/feature.h 文件参考

```

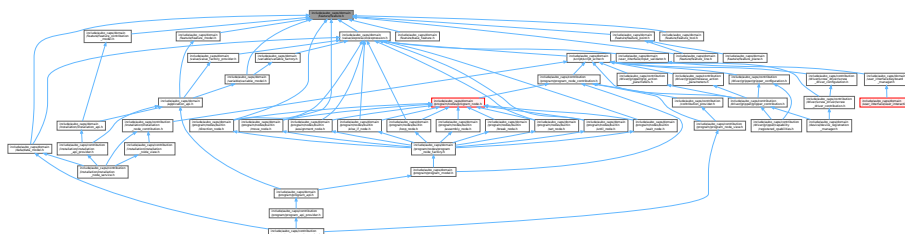
#include <vector>
#include <aubo_caps/class_forward.h>

```

feature.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::Feature`
特征接口此基础接口表示 *AuboScope* 中所有可用的特征类型。
- struct `arcs::aubo_scope::FeatureDefinedFilter`
- struct `arcs::aubo_scope::FeatureJoggableFilter`

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([Feature](#))

12.77 feature.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_FEATURE_H
00002 #define AUBO_SCOPE_FEATURE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009
00010 ARCS_CLASS_FORWARD(Feature);
00011
00012 /**
00013  * \chinese
00014  * 特征接口
00015  * 此基础接口表示 AuboScope 中所有可用的特征类型。
00016  * \endchinese
00017  * \english
00018  * Feature
00019  * This base interface represents all types of features available in AuboScope.
00020  * \endenglish
00021  */
00022 class ARCS_ABI_EXPORT Feature
00023 {
00024 public:
00025     Feature(Feature &f);
00026     Feature(Feature &&f);
00027     virtual ~Feature();
00028
00029     /**
00030      * \chinese
00031      * 获取特征的显示名称。注意：特征可随时被重命名。
00032      * @return 特征名称。
00033      * \endchinese
00034      * \english
00035      * Note: The feature can be renamed at any time.
00036      * @return The name of the feature.
00037      * \endenglish
00038      */
00039     std::string getDisplayName();
00040
00041     /**
00042      * \chinese
00043      * 返回特征的位置是否已完全定义/指定。
00044      * @return 如果特征已定义返回 <code>true</code>, 否则返回 <code>false</code>。
00045      * \endchinese
00046      * \english
00047      * Returns whether the position of the feature has been fully
00048      * defined/specified.
00049      *
00050      * @return <code>true</code> if the feature is defined, <code>false</code>
00051      * otherwise.
00052      * \endenglish
00053      */
00054     bool isDefined();
00055
00056     /**
00057      * \chinese
00058      * 返回特征是否可作为变量在程序中使用。
00059      * @return 如果特征可作为变量使用返回 <code>true</code>, 否则返回 <code>false</code>。
00060      * \endchinese
00061      * \english
00062      * Returns whether the feature can be used as a variable in the program.
00063      *
00064      * @return <code>true</code> if the feature is variable, <code>false</code>
00065      * otherwise.
00066      * \endenglish
00067      */
00068     bool isVariable();
00069     bool isJoggable();

```

```

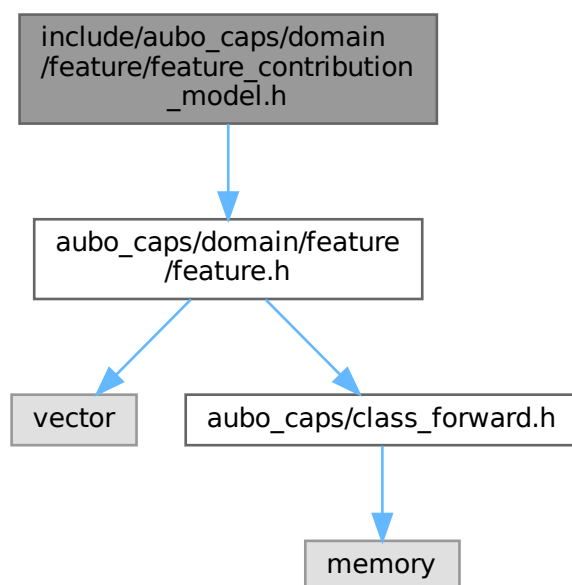
00070
00071 /**
00072  * \chinese
00073  * 特征无法保证始终存在于 AuboScope 中。终端用户可以加载不包含该特征的不同安装，
00074  * 或者终端用户或添加该特征的 AuboCap 可以将其移除。此方法用于判断特征是否存在。
00075  * @return 如果此特征存在于 AuboScope 中返回 <code>true</code>，否则返回
00076  * <code>false</code>。
00077  * \endchinese
00078  * \english
00079  * A feature cannot be guaranteed to be present in AuboScope. The end user
00080  * can load a different installation (not containing the feature) or the
00081  * feature can be removed by the end user or the AuboCap that added the
00082  * feature. This method can be used to determine, if the feature is present.
00083  *
00084  * @return <code>true</code> if this feature is present in AuboScope,
00085  * otherwise <code>false</code>.
00086  * \endenglish
00087  */
00088 bool isResolvable();
00089
00090 /**
00091  * \chinese
00092  * 返回特征的笛卡尔位置。
00093  * @return 表示特征笛卡尔位置的位姿。
00094  * \endchinese
00095  * \english
00096  * Returns the Cartesian location of the feature.
00097  * @return A pose representing the Cartesian location of the feature.
00098  * \endenglish
00099  */
00100 std::vector<double> getPose();
00101
00102 size_t hashCode();
00103
00104 protected:
00105     Feature();
00106
00107 private:
00108     friend class DataSwitch;
00109     void *d_{ nullptr };
00110 };
00111
00112 struct FeatureDefinedFilter
00113 {
00114     bool operator()(FeaturePtr feature)
00115     {
00116         return feature && feature->isDefined();
00117     }
00118 };
00119
00120 struct FeatureJoggableFilter
00121 {
00122     bool operator()(FeaturePtr feature)
00123     {
00124         return feature && feature->isJoggable() && feature->isDefined();
00125     }
00126 };
00127
00128 } // namespace aubo_scope
00129 } // namespace arcs
00130 #endif

```

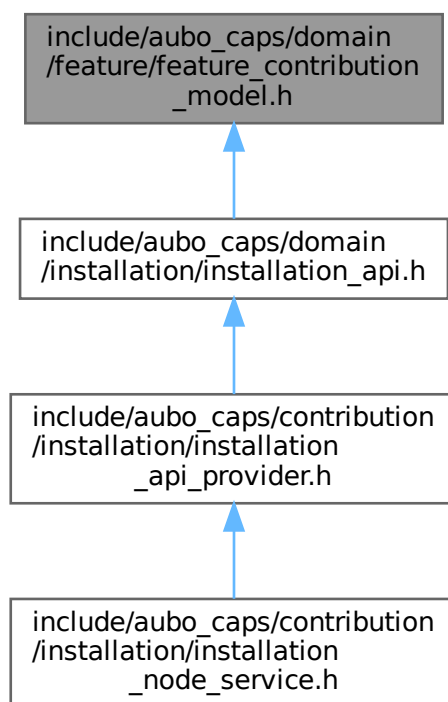
12.78 include/aubo_caps/domain/feature/feature_ contribution_model.h 文件参考

```
#include <aubo_caps/domain/feature/feature.h>
```

feature_contribution_model.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::FeatureContributionModel](#)
特征贡献模型此接口提供在 *AuboScope* 中添加、更新和删除特征的功能。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([FeatureContributionModel](#))

12.79 feature_contribution_model.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_FEATURE_CONTRIBUTION_MODEL_H
00002 #define AUBO_SCOPE_FEATURE_CONTRIBUTION_MODEL_H
00003
00004 #include <aubo_caps/domain/feature/feature.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008
00009 ARCS_CLASS_FORWARD(FeatureContributionModel);
00010
00011 /**
00012  * @ingroup InstallationApi
00013  * \chinese
  
```



```

00014 * 特征贡献模型
00015 * 此接口提供在 AuboScope 中添加、更新和删除特征的功能。
00016 *
00017 * <b> 注意: </b> 此接口只能在安装贡献中使用。
00018 * \endchinese
00019 * \english
00020 * FeatureContributionModel
00021 * This interface provides functionality for adding, updating and removing
00022 * features in AuboScope.
00023 *
00024 * <b>Note:</b> This interface must only be used in an installation
00025 * contribution.
00026 * \endenglish
00027 */
00028 class ARCS_ABI_EXPORT FeatureContributionModel
00029 {
00030 public:
00031     FeatureContributionModel(FeatureContributionModel &f);
00032     FeatureContributionModel(FeatureContributionModel &&f);
00033     ~FeatureContributionModel();
00034
00035     /**
00036     * \chinese
00037     * 向当前 AuboScope 安装中添加一个特征。添加后终端用户可选择该特征，但不可修改。
00038     *
00039     * @param idKey 用于标识此特征的键。该键仅对此 AuboCap 有效。
00040     * @param suggestedName 建议的特征名称。有效名称必须匹配正则表达式
00041     * [a-zA-Z][a-zA-Z0-9_]{0,14}。
00042     * @param pose 特征相对于机器人基座的位姿。
00043     * @return 在 AuboScope 中创建并注册的特征。
00044     * @throws FeatureAlreadyAddedException 如果之前已用相同 idKey 添加过特征。
00045     * @throws IllegalFeatureNameException 如果建议的名称不匹配所需正则表达式。
00046     * \endchinese
00047     * \english
00048     * Add a feature to the current AuboScope installation. This makes it
00049     * selectable by the end user. The feature is not modifiable by the end
00050     * user.
00051     *
00052     * @param idKey The key to identify this feature by.
00053     * @param suggestedName Suggested name for the feature.
00054     * @param pose The pose of the feature with respect to the robot base.
00055     * @return The feature created and registered in AuboScope.
00056     * @throws FeatureAlreadyAddedException If a feature has previously been
00057     * added the same <code>idKey</code> identifier.
00058     * @throws IllegalFeatureNameException If the suggested name does not match
00059     * required regex.
00060     * \endenglish
00061     */
00062     FeaturePtr addFeature(const std::string &idKey,
00063                          const std::string &suggestedName,
00064                          const std::vector<double> &pose);
00065
00066     /**
00067     * \chinese
00068     * 返回此前由此 AuboCap 使用相同 idKey 标识符添加的特征。用于验证特征是否存在。
00069     * @param idKey 用于标识特征的键。
00070     * @return 此前添加的特征，如果不存在则返回 <code>null</code>。
00071     * \endchinese
00072     * \english
00073     * Returns the feature previously added by this AuboCap using the same
00074     * <code>idKey</code> identifier. Use this to verify if the feature is
00075     * present in the current installation.
00076     *
00077     * @param idKey The key to identify this feature by.
00078     * @return The feature previously added by this AuboCap.
00079     * Returns <code>null</code> if no feature exists in the current
00080     * installation.
00081     * \endenglish
00082     */
00082     FeaturePtr getFeature(const std::string &idKey);
00083
00084     /**
00085     * \chinese
00086     * 更新此 AuboCap 已添加特征的位姿。
00087     * @param idKey 用于标识特征的键。
00088     * @param newPose 特征的新位姿。
00089     * @throws FeatureNotFoundException 如果不存在对应 idKey 的特征。
00090     * \endchinese
00091     * \english
00092     * Update the pose of an existing feature added by this AuboCap.
00093     *
00094     * @param idKey The key to identify the feature.
00095     * @param newPose The new pose to set for the feature.
00096     * @throws FeatureNotFoundException If no feature exists with the provided
00097     * <code>idKey</code>.
00098     * \endenglish
00099     */
00100     void updateFeature(const std::string &idKey,

```

```

00101         const std::vector<double> &newPose);
00102 void renameFeature(const std::string &idKey, const std::string &newName);
00103
00104 /**
00105  * \chinese
00106  * 从此 AuboCap 中移除已添加的特征。使用该特征的程序节点将变为未定义。
00107  * @param idKey 添加特征时使用的键。
00108  * @throws FeatureNotFoundException 如果不存在对应 idKey 的特征。
00109  * \endchinese
00110  * \english
00111  * Remove a feature added by this AuboCap from AuboScope. Program nodes
00112  * using the feature will be become undefined because the feature is no
00113  * longer resolvable.
00114  *
00115  * @param idKey The key used to add the feature with.
00116  * @throws FeatureNotFoundException If no feature exists with the provided
00117  * <code>idKey</code>.
00118  * \endenglish
00119  */
00120 void removeFeature(const std::string &idKey);
00121
00122 private:
00123     friend class DataSwitch;
00124     FeatureContributionModel();
00125     void *d_{ nullptr };
00126 };
00127
00128 } // namespace aubo_scope
00129 } // namespace arcs
00130
00131 #endif // AUBO_SCOPE_FEATURE_CONTRIBUTION_MODEL_H

```

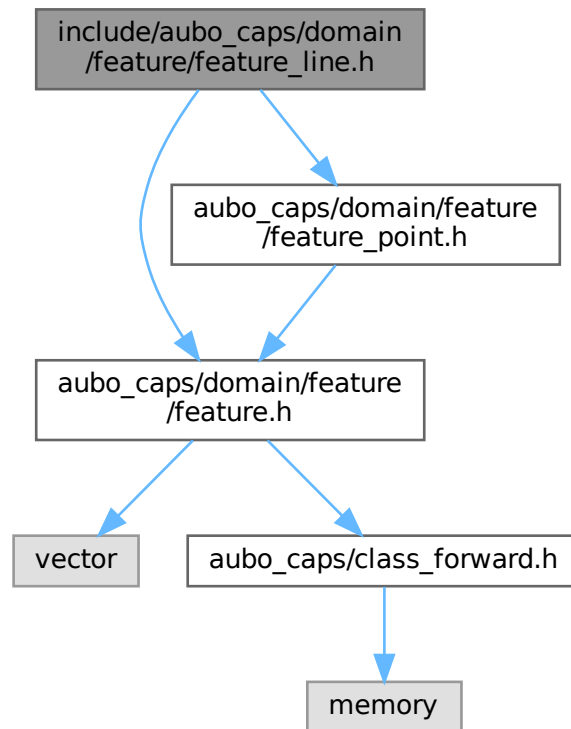
12.80 include/aubo_caps/domain/feature/feature_line.h 文件参考

```

#include <aubo_caps/domain/feature/feature.h>
#include <aubo_caps/domain/feature/feature_point.h>

```

feature_line.h 的引用 (Include) 关系图:



类

- `class arcs::aubo_scope::FeatureLine`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (FeatureLine)`

12.81 feature_line.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_FEATURE_LINE_H
00002 #define AUBO_SCOPE_FEATURE_LINE_H
00003
00004 #include <aubo_caps/domain/feature/feature.h>
00005 #include <aubo_caps/domain/feature/feature_point.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009
00010 ARCS_CLASS_FORWARD (FeatureLine);
00011
00012 class ARCS_ABI_EXPORT FeatureLine : public Feature
00013 {

```

```

00014 public:
00015     FeatureLine(FeatureLine &f);
00016     FeatureLine(FeatureLine &&f);
00017     virtual ~FeatureLine();
00018
00019 private:
00020     friend class DataSwitch;
00021     FeatureLine();
00022     void *d_{ nullptr };
00023 };
00024
00025 } // namespace aubo_scope
00026 } // namespace arcs
00027
00028 #endif // AUBO_SCOPE_FEATURE_LINE_H

```

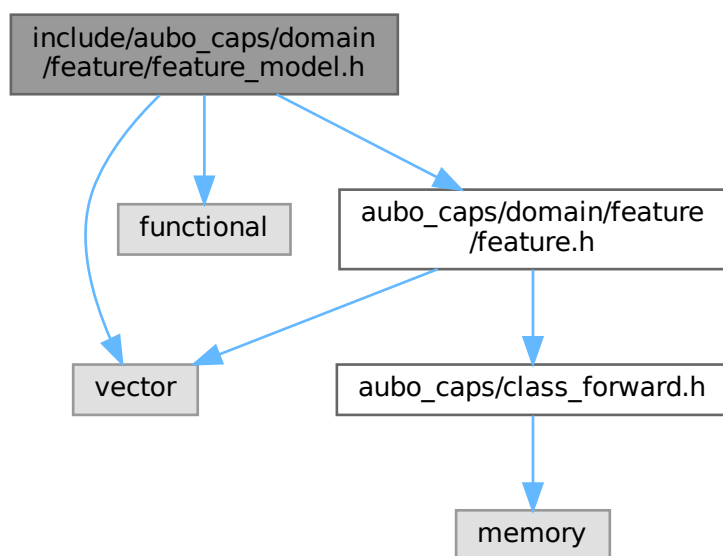
12.82 include/aubo_caps/domain/feature/feature_model.h 文件参考

```

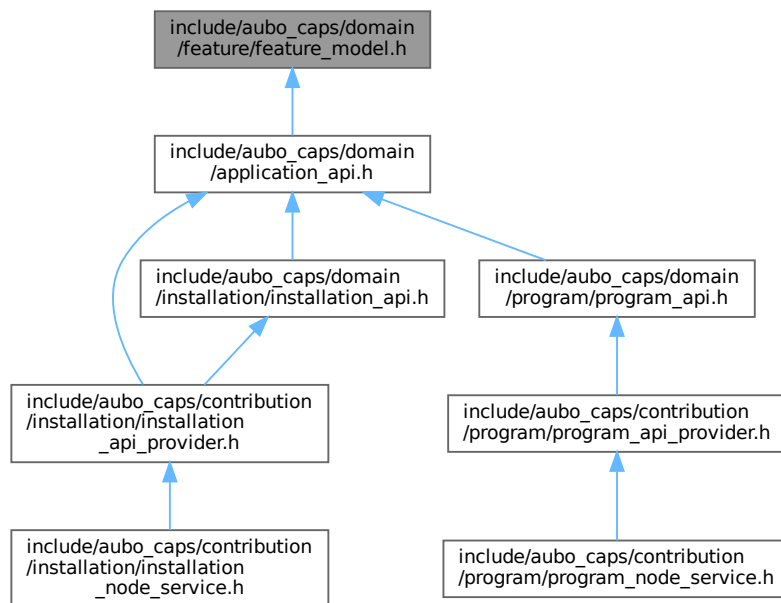
#include <vector>
#include <functional>
#include <aubo_caps/domain/feature/feature.h>

```

feature_model.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class `arcs::aubo_scope::FeatureModel`
特征模型提供从当前机器人安装中获取特征的方法。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (FeatureModel)`

12.83 feature_model.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_FEATURE_MODEL_H
00002 #define AUBO_SCOPE_FEATURE_MODEL_H
00003
00004 #include <vector>
00005 #include <functional>
00006 #include <aubo_caps/domain/feature/feature.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010
00011 ARCS_CLASS_FORWARD (FeatureModel);
00012
00013 /**
00014  * @ingroup ApplicationApi
00015  * \chinese
00016  * 特征模型
00017  * 提供从当前机器人安装中获取特征的方法。
00018  * \endchinese
00019  * \english
00020  * FeatureModel
00021  * Provides methods that returns features from the current robot installation.
  
```

```

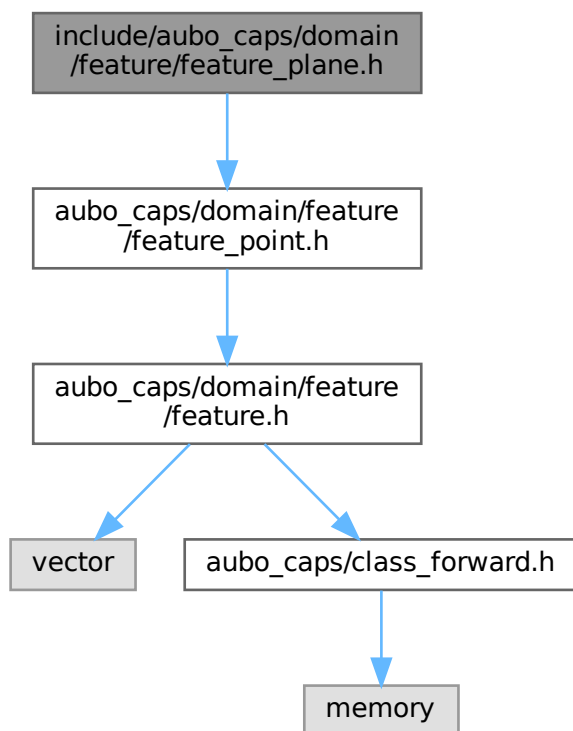
00022 * \endenglish
00023 */
00024 class ARCS_ABI_EXPORT FeatureModel
00025 {
00026 public:
00027     FeatureModel(FeatureModel &f);
00028     FeatureModel(FeatureModel &&f);
00029     ~FeatureModel();
00030
00031     std::vector<FeaturePtr> getAll();
00032
00033     /**
00034      * @ingroup FeatureModel
00035      * \chinese
00036      * 获取当前安装中的几何特征
00037      * @return 几何特征集合。
00038      * \endchinese
00039      * \english
00040      * Returns the geometric features of the current installation.
00041      * @return the geometric features of the current installation.
00042      * \endenglish
00043      */
00044     template <typename T>
00045     std::vector<FeaturePtr> getGeomFeture();
00046
00047     /**
00048      * @ingroup FeatureModel
00049      * \chinese
00050      * 根据名称获取特征
00051      * @param name 特征名称。
00052      * @return 对应名称的特征。
00053      * \endchinese
00054      * \english
00055      * Returns a feature by name.
00056      * @param name the name of the feature.
00057      * @return the feature with the given name.
00058      * \endenglish
00059      */
00060     FeaturePtr getByName(const std::string &name);
00061
00062     /**
00063      * @ingroup FeatureModel
00064      * \chinese
00065      * 获取当前安装的基础特征
00066      * @return 基础特征。
00067      * \endchinese
00068      * \english
00069      * Returns the base feature of the current installation.
00070      * @return the base feature of the current installation.
00071      * \endenglish
00072      */
00073     FeaturePtr getBaseFeature();
00074
00075     /**
00076      * @ingroup FeatureModel
00077      * \chinese
00078      * 获取当前安装的工具特征
00079      * @return 工具特征。
00080      * \endchinese
00081      * \english
00082      * Returns the tool feature of the current installation.
00083      * @return the tool feature of the current installation.
00084      * \endenglish
00085      */
00086     FeaturePtr getToolFeature();
00087
00088     /**
00089      * @ingroup FeatureModel
00090      * \chinese
00091      * 根据过滤器获取特征列表
00092      * @param filter 过滤条件。
00093      * @return 满足过滤条件的特征集合。
00094      * \endchinese
00095      * \english
00096      * Returns features matching the given filter.
00097      * @param filter the filter to apply.
00098      * @return a collection of features that match the filter.
00099      * \endenglish
00100      */
00101     std::vector<FeaturePtr> getFeatures(std::function<bool(FeaturePtr)> filter);
00102
00103 private:
00104     friend class DataSwitch;
00105     FeatureModel();
00106     void *d_{ nullptr };
00107 };
00108

```

```
00109 } // namespace aubo_scope
00110 } // namespace arcs
00111
00112 #endif // AUBO_SCOPE_FEATURE_MODEL_H
```

12.84 include/aubo_caps/domain/feature/feature_plane.h 文件参考

#include <aubo_caps/domain/feature/feature_point.h>
feature_plane.h 的引用 (Include) 关系图:



类

- `class arcs::aubo_scope::FeaturePlane`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (FeaturePlane)`

12.85 feature_plane.h

[浏览该文件的文档.](#)

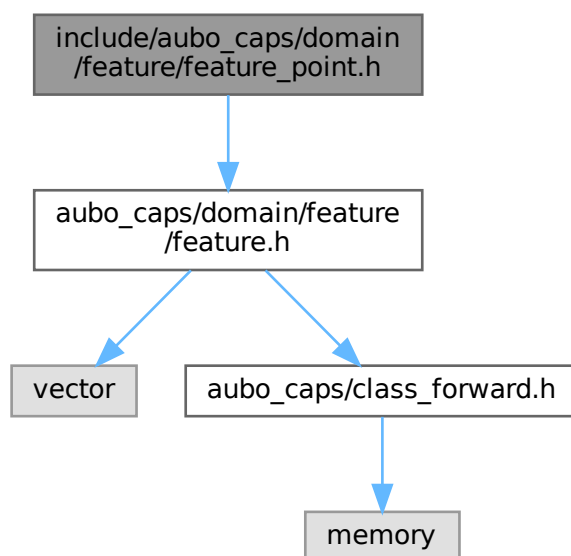
```

00001 #ifndef AUBO_SCOPE_FEATURE_PLANE_H
00002 #define AUBO_SCOPE_FEATURE_PLANE_H
00003
00004 #include <aubo_caps/domain/feature/feature_point.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(FeaturePlane);
00009
00010         class ARCS_ABI_EXPORT FeaturePlane : public Feature
00011         {
00012         public:
00013             FeaturePlane(FeaturePlane &f);
00014             FeaturePlane(FeaturePlane &&f);
00015             virtual ~FeaturePlane();
00016
00017         private:
00018             friend class DataSwitch;
00019             FeaturePlane();
00020             void *d_{ nullptr };
00021         };
00022
00023     } // namespace aubo_scope
00024 } // namespace arcs
00025
00026 #endif // AUBO_SCOPE_FEATURE_PLANE_H

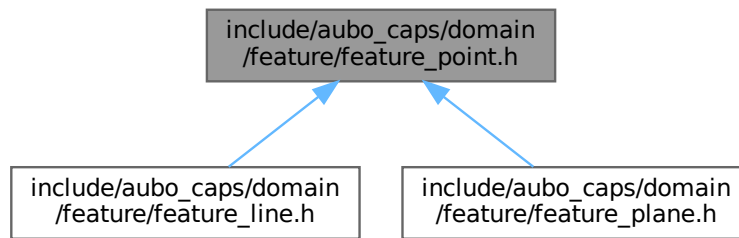
```

12.86 include/aubo_caps/domain/feature/feature_point.h 文件参考

#include <aubo_caps/domain/feature/feature.h>
 feature_point.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- `class arcs::aubo_scope::FeaturePoint`

This interface represents a Point feature in AboScope

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (FeaturePoint)`

12.87 feature_point.h

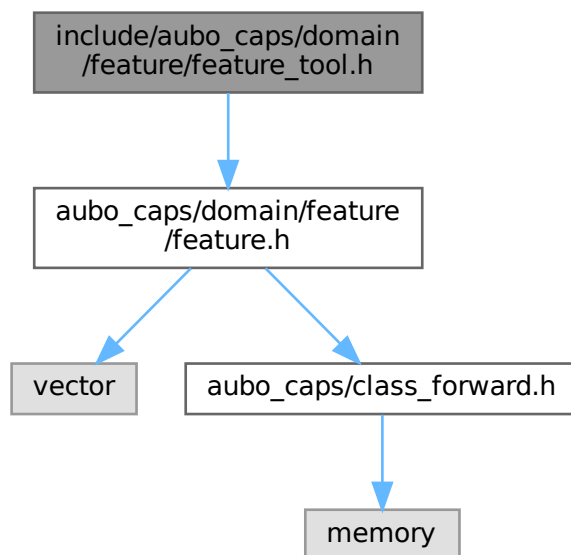
[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_FEATURE_POINT_H
00002 #define AUBO_SCOPE_FEATURE_POINT_H
00003
00004 #include <aubo_caps/domain/feature/feature.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD (FeaturePoint);
00009
00010         /**
00011          * This interface represents a Point feature in AboScope
00012          */
00013         class ARCS_ABI_EXPORT FeaturePoint : public Feature
00014         {
00015         public:
00016             FeaturePoint (FeaturePoint &f);
00017             FeaturePoint (FeaturePoint &&f);
00018             virtual ~FeaturePoint();
00019
00020         private:
00021             friend class DataSwitch;
00022             FeaturePoint();
00023             void *d_{ nullptr };
00024         };
00025
00026     } // namespace aubo_scope
00027 } // namespace arcs
00028
00029 #endif // AUBO_SCOPE_FEATURE_POINT_H
  
```

12.88 include/aubo_caps/domain/feature/feature_tool.h 文件参考

`#include <aubo_caps/domain/feature/feature.h>`
 feature_tool.h 的引用 (Include) 关系图:



类

- class `arcs::aubo_scope::FeatureTool`
This interface represents the Tool feature in AuboScope.

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (FeatureTool)`

12.89 feature_tool.h

[浏览该文件的文档.](#)

```

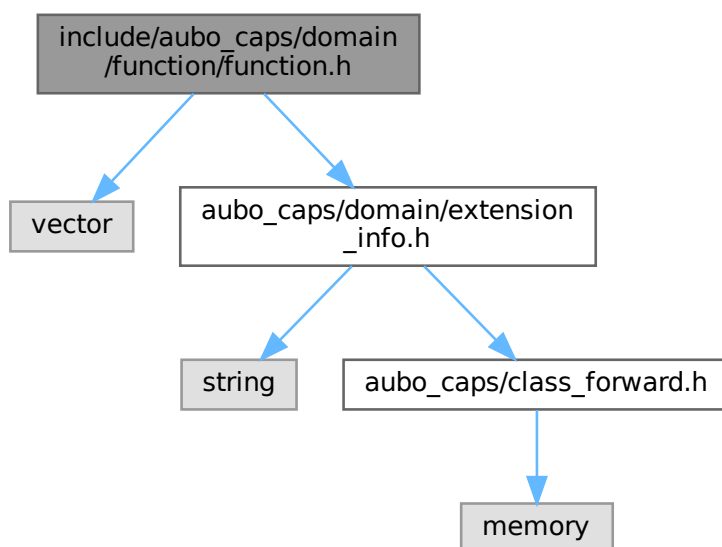
00001 #ifndef AUBO_SCOPE_FEATURE_TOOL_H
00002 #define AUBO_SCOPE_FEATURE_TOOL_H
00003
00004 #include <aubo_caps/domain/feature/feature.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD (FeatureTool);
00009
00010         /**
00011          * This interface represents the Tool feature in AuboScope.
00012          */
00013         class ARCS_ABI_EXPORT FeatureTool : public Feature
  
```

```
00014 {
00015 public:
00016     FeatureTool(FeatureTool &f);
00017     FeatureTool(FeatureTool &&f);
00018     virtual ~FeatureTool();
00019
00020 private:
00021     friend class DataSwitch;
00022     FeatureTool();
00023     void *d_{ nullptr };
00024 };
00025
00026 } // namespace aubo_scope
00027 } // namespace arcs
00028
00029 #endif // AUBO_SCOPE_FEATURE_TOOL_H
```

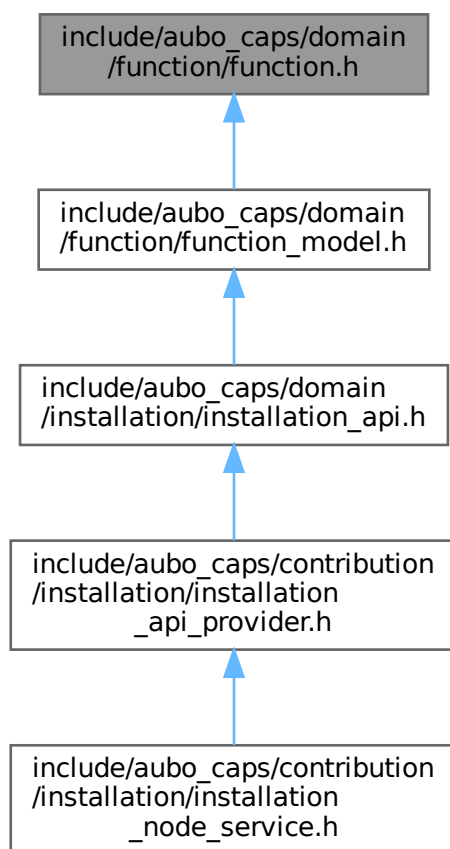
12.90 include/aubo_caps/domain/function/function.h 文件参考

```
#include <vector>
#include <aubo_caps/domain/extension_info.h>
```

function.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::Function](#)
函数接口表示在表达式编辑器中显示的函数。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(Function\)](#)

12.91 function.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_FUNCTION_H
00002 #define AUBO_SCOPE_FUNCTION_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/extension_info.h>
00006

```

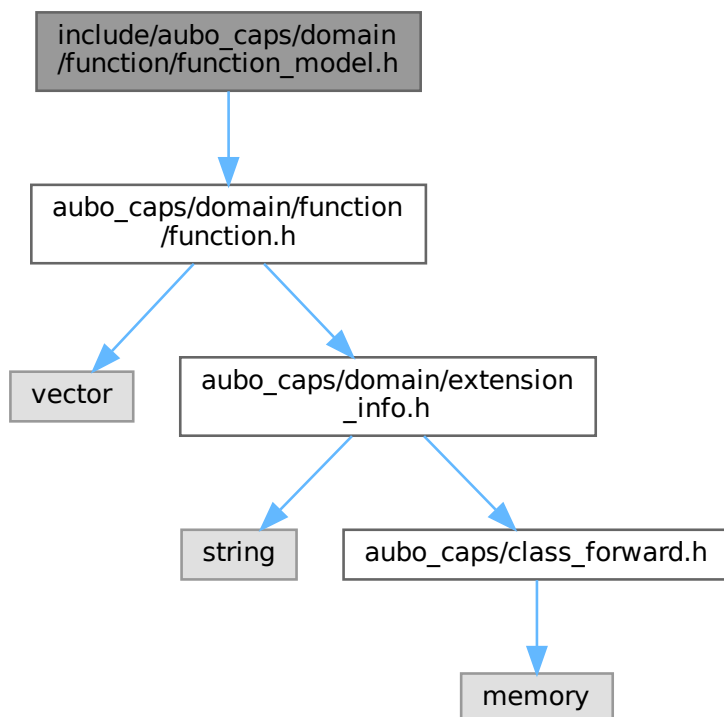
```

00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(Function);
00010
00011 /**
00012  * \chinese
00013  * 函数接口
00014  * 表示在表达式编辑器中显示的函数。
00015  *
00016  * 另请参见 {@link FunctionModel}。
00017  * \endchinese
00018  * \english
00019  * Function
00020  * Represents a function shown in the Expression Editor.
00021  *
00022  * See also {@link FunctionModel}.
00023  * \endenglish
00024  */
00025 class ARCS_ABI_EXPORT Function
00026 {
00027 public:
00028     Function(Function &f);
00029     Function(Function &&f);
00030     ~Function();
00031
00032     /**
00033      * \chinese
00034      * 获取函数名称
00035      * @return 函数名称。
00036      * \endchinese
00037      * \english
00038      * Returns the name of the function.
00039      * @return The name of the function.
00040      * \endenglish
00041      */
00042     std::string getName();
00043
00044     /**
00045      * \chinese
00046      * 获取函数参数列表
00047      * @return 函数参数列表。
00048      * \endchinese
00049      * \english
00050      * Returns a list of the arguments of the function.
00051      * @return A list of the arguments of the function.
00052      * \endenglish
00053      */
00054     std::vector<std::string> getArguments();
00055
00056     /**
00057      * \chinese
00058      * 获取将此函数添加/贡献到 AuboScope 的 AuboCap 信息。
00059      * @return 此函数的 AuboCap 信息。
00060      * \endchinese
00061      * \english
00062      * Get information about the AuboCap that added/contributed this function
00063      * to AuboScope.
00064      *
00065      * @return AuboCap information for this function.
00066      * \endenglish
00067      */
00068     AuboCapInfoPtr getContributorInfo();
00069
00070 private:
00071     friend class DataSwitch;
00072     Function();
00073     void *d_{ nullptr };
00074 };
00075
00076 } // namespace aubo_scope
00077 } // namespace arcs
00078
00079 #endif // AUBO_SCOPE_FUNCTION_H

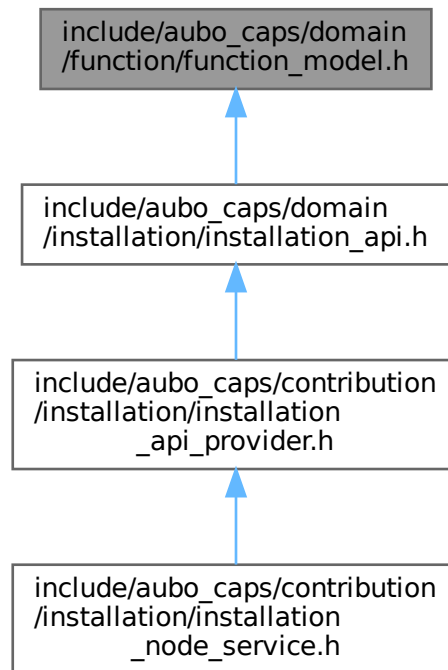
```

12.92 include/aubo_caps/domain/function/function_model.h 文件参考

`#include <aubo_caps/domain/function/function.h>`
function_model.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::FunctionModel](#)

函数模型此接口提供在表达式编辑器中添加和删除脚本函数的方法, 该编辑器可从赋值和 *If* 程序节点等位置访问。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([FunctionModel](#))

12.93 function__model.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_FUNCTION_MODEL_H
00002 #define AUBO_SCOPE_FUNCTION_MODEL_H
00003
00004 #include <aubo_caps/domain/function/function.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(FunctionModel);
00009     }
00010 }
00011 /**
00012  * @ingroup InstallationApi
00013  * \chinese
00014  * 函数模型
  
```

```

00014 * 此接口提供在表达式编辑器中添加和删除脚本函数的方法，该编辑器可从赋值和 If
00015 * 程序节点等位置访问。
00016 *
00017 * 添加的函数仅为终端用户提供了一种便捷的方式来使用贡献的函数。特定脚本函数的实现
00018 * 必须通过 {@link InstallationNodeContribution#generateScript(ScriptWriter)}
00019 * 添加到程序的前导部分。
00020 *
00021 * 所有可贡献的函数都必须始终添加到前导部分，无论它们是否被添加到表达式编辑器中。
00022 * 否则，依赖脚本函数贡献的旧程序可能会停止工作。
00023 *
00024 * 另请参见 {@link Function}。
00025 * \endchinese
00026 * \english
00027 * FunctionModel
00028 * This interface provides methods to add and remove script functions in the
00029 * Expression Editor accessible from Assignment and If program nodes among
00030 * others.
00031 *
00032 * Added functions merely serve as an easy way for the end customer to use the
00033 * contributed functions. The implementation of a given script function must be
00034 * added in the preamble of program through the use of
00035 * {@link InstallationNodeContribution#generateScript(ScriptWriter)}.
00036 *
00037 * The full set of functions which can be contributed must be added in the
00038 * preamble at all times, regardless of whether they are added to the Expression
00039 * Editor or not. Otherwise old programs that rely on a script function
00040 * contribution might stop working.
00041 *
00042 * See also {@link Function}.
00043 * \endenglish
00044 */
00045 class ARCS_ABI_EXPORT FunctionModel
00046 {
00047 public:
00048     FunctionModel(FunctionModel &f);
00049     FunctionModel(FunctionModel &&f);
00050     ~FunctionModel();
00051
00052     /**
00053      * \chinese
00054      * 添加脚本函数到表达式编辑器。有效名称必须匹配正则表达式 [a-zA-Z][a-zA-Z0-9_]{0,24}，
00055      * 共 25 个字符。部分 AUBO Robot 保留名称不可使用。
00056      * @param argumentNames 函数参数提示列表，在表达式编辑器中以逗号分隔显示在尖括号
00057      * 内。最多指定五个参数。
00058      * @return 成功添加返回表示函数的对象，否则返回 null。
00059      * @throws FunctionException 如果名称非法或指定了超过五个参数。
00060      * \endchinese
00061      * \english
00062      * Add a script function to the Expression Editor.
00063      * @param name Name of the function. Safe names must match regex
00064      * [a-zA-Z][a-zA-Z0-9_]{0,24} for a total of 25 characters.
00065      * @param argumentNames Is a list of hints for the function arguments.
00066      * @return If the script function was successfully added, an object
00067      * representing the Function will be returned. Null otherwise.
00068      * @throws FunctionException if the name is illegal or more than five
00069      * arguments are specified.
00070      * \endenglish
00071      */
00072     FunctionPtr addFunction(const std::string &name,
00073                             const std::vector<std::string> &argumentNames);
00074
00075     /**
00076      * \chinese
00077      * 移除函数。只能移除此 AuboCap 添加的函数。
00078      * @param function 要移除的函数。
00079      * @return 成功移除返回 <code>true</code>，否则返回 <code>false</code>。
00080      * \endchinese
00081      * \english
00082      * Remove a function. Only functions added by this AuboCap can be removed.
00083      * @param function Function to be removed.
00084      * @return <code>true</code>, if the function was successfully removed.
00085      * \endenglish
00086      */
00087     bool removeFunction(FunctionPtr func);
00088
00089     /**
00090      * \chinese
00091      * 根据名称获取函数
00092      * @param name 函数名称。
00093      * @return 找到返回函数，否则返回 null。
00094      * \endchinese
00095      * \english
00096      * Get a function by name.
00097      * @param name The name of the function.
00098      * @return The function if found, otherwise null.
00099      * \endenglish
00100

```



```

00101     */
00102     FunctionPtr getFunction(const std::string &name);
00103
00104     /**
00105     * \chinese
00106     * 获取所有已添加的函数
00107     * @return 已添加函数的集合。
00108     * \endchinese
00109     * \english
00110     * Get all added functions.
00111     * @return A collection of added functions.
00112     * \endenglish
00113     */
00114     std::vector<FunctionPtr> getFunctions();
00115
00116 private:
00117     friend class DataSwitch;
00118     FunctionModel();
00119     void *d_{ nullptr };
00120 };
00121
00122 } // namespace aubo_scope
00123 } // namespace arcs
00124
00125 #endif // AUBO_SCOPE_FUNCTION_MODEL_H

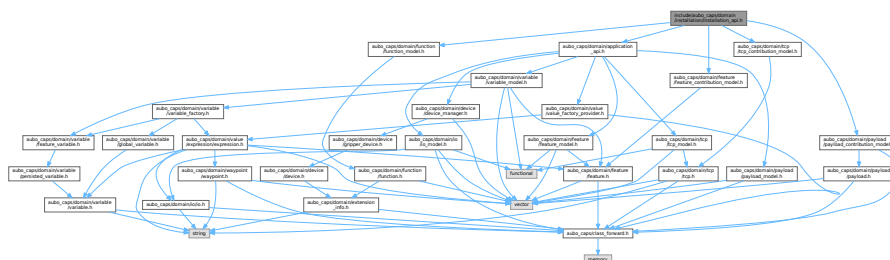
```

12.94 include/aubo_caps/domain/installation/installation_api.h 文件参考

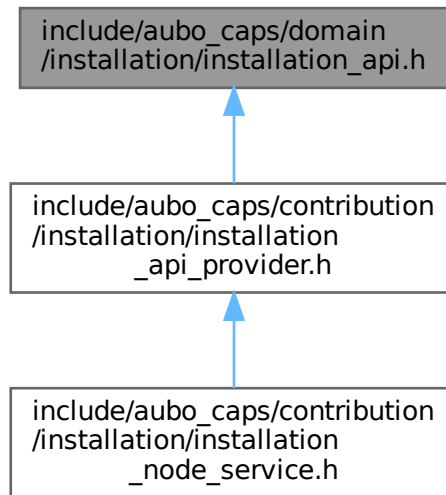
```

#include <aubo_caps/domain/application_api.h>
#include <aubo_caps/domain/function/function_model.h>
#include <aubo_caps/domain/tcp/tcp_contribution_model.h>
#include <aubo_caps/domain/feature/feature_contribution_model.h>
#include <aubo_caps/domain/payload/payload_contribution_model.h>
installation_api.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::InstallationApi](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(InstallationApi\)](#)

12.95 installation_api.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_INSTALLATION_API_H
00002 #define AUBO_SCOPE_INSTALLATION_API_H
00003
00004 #include <aubo_caps/domain/application_api.h>
00005 #include <aubo_caps/domain/function/function_model.h>
00006 #include <aubo_caps/domain/tcp/tcp_contribution_model.h>
00007 #include <aubo_caps/domain/feature/feature_contribution_model.h>
00008 #include <aubo_caps/domain/payload/payload_contribution_model.h>
00009
00010 namespace arcs {
00011 namespace aubo_scope {
00012 ARCS_CLASS_FORWARD (InstallationApi);
00013
00014 /**
00015  * \chinese
00016  * @defgroup InstallationApi InstallationApi(安装)
00017  * @ingroup InstallationApi
00018  * 安装接口
00019  * 提供与安装相关的功能和服务访问。这包括 AuboScope 内部已有的功能，以及在
00020  * AuboScope 中创建要使用的额外模型元素。
00021  * \endchinese
00022  * \english
00023  * @defgroup InstallationApi InstallationApi
00024  * @ingroup InstallationApi
00025  * InstallationApi
  
```

```

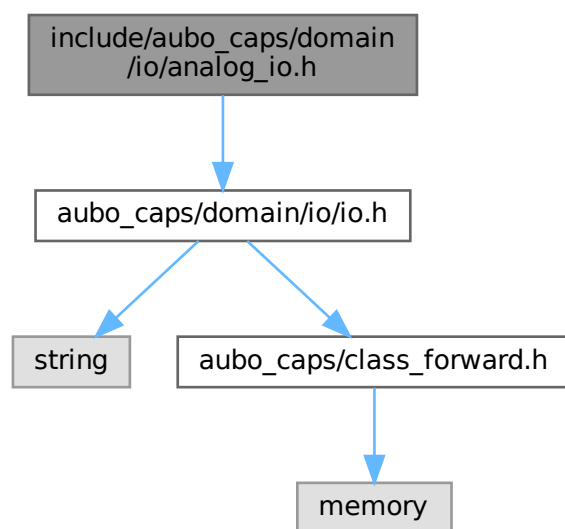
00026 * Provides access to functionality and services related to the installation.
00027 * This includes what is available from within AudoScope, as well as creating
00028 * additional model elements to be used within AudoScope.
00029 * \endenglish
00030 */
00031 class ARCS_ABI_EXPORT InstallationApi : public ApplicationApi
00032 {
00033 public:
00034     InstallationApi(InstallationApi &f);
00035     InstallationApi(InstallationApi &&f);
00036     virtual ~InstallationApi();
00037
00038     /**
00039      * @ingroup InstallationApi
00040      * \chinese
00041      * 获取向 AudoScope 添加脚本功能的接口
00042      * @return 用于添加脚本功能的接口。
00043      * \endchinese
00044      * \english
00045      * Returns an interface for adding script functions to AudoScope.
00046      * @return An interface for adding script functions to AudoScope.
00047      * \endenglish
00048      */
00049     FunctionModelPtr getFunctionModel();
00050
00051     /**
00052      * @ingroup InstallationApi
00053      * \chinese
00054      * 获取向 AudoScope 贡献 TCP 的接口
00055      * @return 用于贡献 TCP 的接口。
00056      * \endchinese
00057      * \english
00058      * Returns an interface for contributing TCPs to AudoScope.
00059      * @return An interface for contributing TCPs to AudoScope.
00060      * \endenglish
00061      */
00062     TcpContributionModelPtr getTcpContributionModel();
00063
00064     /**
00065      * @ingroup InstallationApi
00066      * \chinese
00067      * 获取向 AudoScope 贡献特征的接口
00068      * @return 用于贡献特征的接口。
00069      * \endchinese
00070      * \english
00071      * Returns an interface for contributing features to AudoScope.
00072      * @return An interface for contributing features to AudoScope.
00073      * \endenglish
00074      */
00075     FeatureContributionModelPtr getFeatureContributionModel();
00076
00077     /**
00078      * @ingroup InstallationApi
00079      * \chinese
00080      * <p>
00081      * 获取向 AudoScope 安装贡献负载的接口。
00082      * </p>
00083      *
00084      * @return 用于贡献负载的接口。
00085      * \endchinese
00086      * \english
00087      * <p>
00088      * Gets an interface which provides functionality for contributing payloads
00089      * to the AudoScope installation.
00090      * </p>
00091      *
00092      * @return An interface for contributing payloads to AudoScope.
00093      * \endenglish
00094      */
00095     PayloadContributionModelPtr getPayloadContributionModel();
00096
00097 private:
00098     friend class DataSwitch;
00099     InstallationApi();
00100     void *d_{ nullptr };
00101 };
00102
00103 } // namespace audo_scope
00104 } // namespace arcs
00105
00106 #endif // AUDO_SCOPE_INSTALLATION_API_H

```

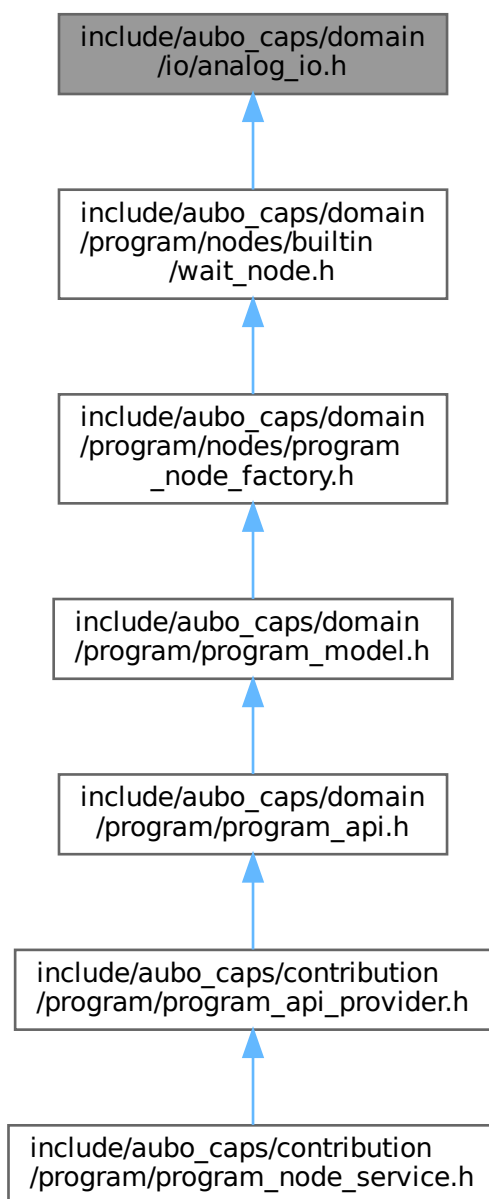
12.96 include/aubo_caps/domain/io/analog_io.h 文件参考

```
#include <aubo_caps/domain/io/io.h>
```

analog_io.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class [arcs::aubo_scope::AnalogIo](#)
模拟 IO 接口此接口提供对模拟 I/O 的支持。
- struct [arcs::aubo_scope::AnalogIoFilter](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (AnalogIo)`

12.97 analog_io.h

浏览该文件的文档.

```

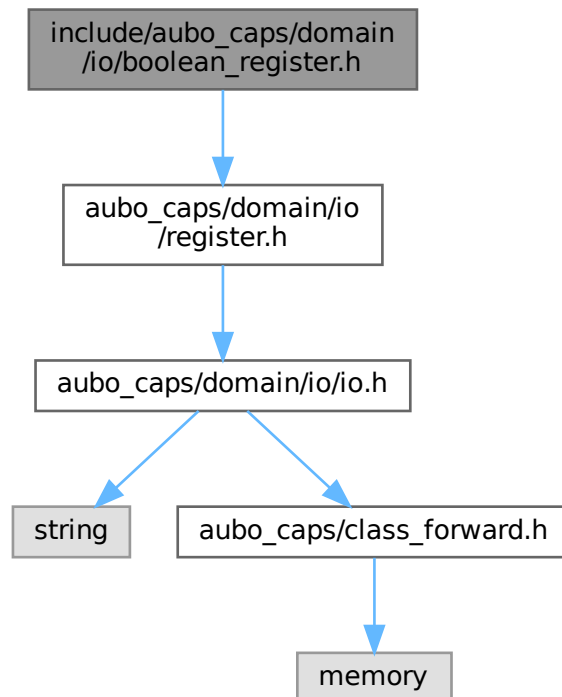
00001 #ifndef AUBO_SCOPE_ANALOG_IO_H
00002 #define AUBO_SCOPE_ANALOG_IO_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(AnalogIo);
00009
00010 /**
00011 * \chinese
00012 * 模拟 IO 接口
00013 * 此接口提供对模拟 I/O 的支持。
00014 * \endchinese
00015 * \english
00016 * AnalogIo
00017 * This interface provides support for analog I/Os.
00018 * \endenglish
00019 */
00020 class ARCS_ABI_EXPORT AnalogIo : public Io
00021 {
00022 public:
00023     AnalogIo(AnalogIo &f);
00024     AnalogIo(AnalogIo &&f);
00025     ~AnalogIo();
00026
00027     double getMinRangeValue();
00028     double getMaxRangeValue();
00029
00030     bool isCurrent();
00031     bool isVoltage();
00032
00033     /**
00034     * \chinese
00035     * 设置模拟 I/O 输出信号。注意该 I/O 必须是输出类型。
00036     * @param value 要设置的模拟值。
00037     * @return 如果模拟信号设置成功返回 <code>true</code>, 否则返回 <code>false</code>。
00038     * \endchinese
00039     * \english
00040     * Set analog I/O output signal. Notice the I/O must be an output I/O.
00041     *
00042     * @param value The analog value to be set.
00043     * @return <code>true</code> if the analog signal was set,
00044     * <code>false</code> if the signal was not set, e.g. because the controller
00045     * was not running.
00046     * \endenglish
00047     */
00048     bool setValue(double value);
00049
00050     /**
00051     * \chinese
00052     * 获取 I/O 的最后一次读数
00053     * @return I/O 的最后一次读数。
00054     * \endchinese
00055     * \english
00056     * Returns the last reading of the I/O.
00057     * @return Returns the last reading of the I/O.
00058     * \endenglish
00059     */
00060     double getValue();
00061
00062 private:
00063     friend class DataSwitch;
00064     AnalogIo();
00065     void *d_{ nullptr };
00066 };
00067
00068 struct AnalogIoFilter
00069 {
00070     AnalogIoFilter(Io::InterfaceType interface, bool is_input)
00071         : interface_(interface), is_input_(is_input)
00072     {
00073     }
00074     bool operator()(IoPtr io)
00075     {
00076         return std::dynamic_pointer_cast<AnalogIo>(io) != nullptr &&
00077             io->getInterfaceType() == interface_ &&

```

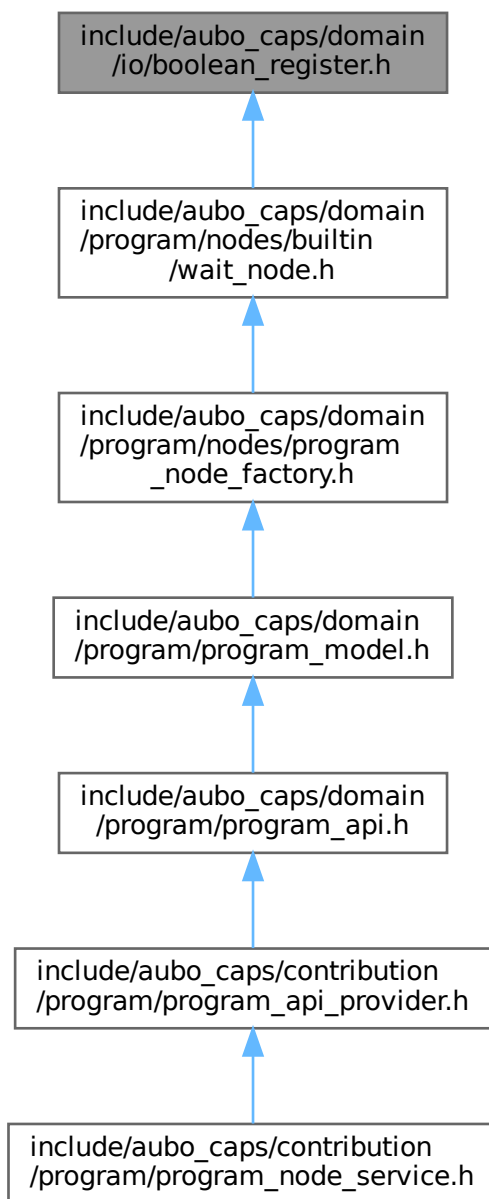
```
00078         io->isInput() == is_input_;
00079     }
00080
00081 private:
00082     Io::InterfaceType interface_;
00083     bool is_input_;
00084 };
00085
00086 } // namespace aubo_scope
00087 } // namespace arcs
00088
00089 #endif // AUBO_SCOPE_ANALOG_IO_H
```

12.98 include/auco_caps/domain/io/boolean_register.h 文件参考

#include <auco_caps/domain/io/register.h>
boolean_register.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class [arcs::aubo_scope::BoolRegister](#)

This interface provides support for bool registers.

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (BoolRegister)`

12.99 boolean__register.h

[浏览该文件的文档.](#)

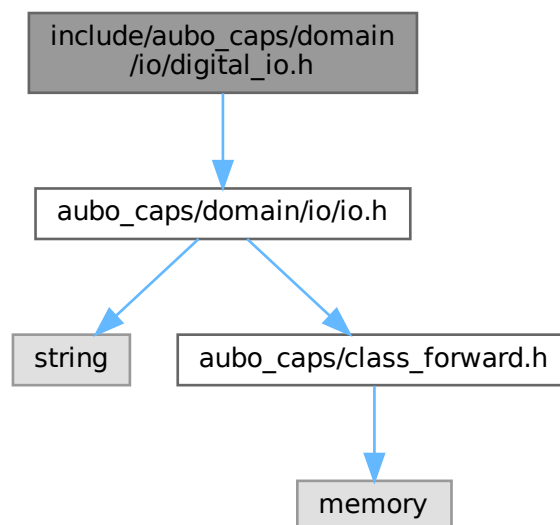
```

00001 #ifndef AUBO_SCOPE_BOOLEAN_REGISTER_H
00002 #define AUBO_SCOPE_BOOLEAN_REGISTER_H
00003
00004 #include <aubo_caps/domain/io/register.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(BoolRegister);
00009
00010 /**
00011  * This interface provides support for bool registers.
00012  *
00013  */
00014 class ARCS_ABI_EXPORT BoolRegister : public Register
00015 {
00016 public:
00017     virtual ~BoolRegister() = default;
00018 };
00019
00020 } // namespace aubo_scope
00021 } // namespace arcs
00022
00023 #endif // AUBO_SCOPE_BOOLEAN_REGISTER_H

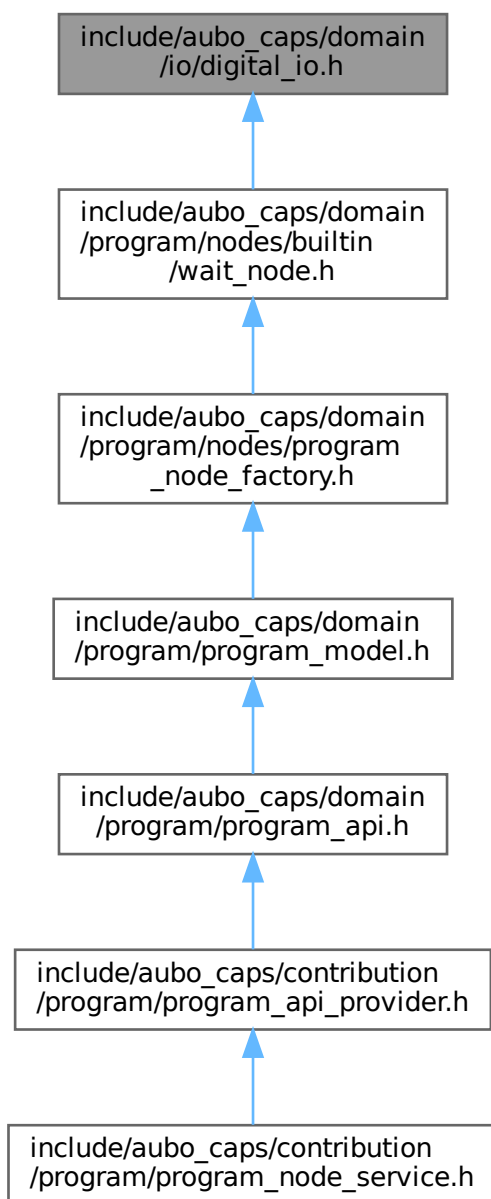
```

12.100 include/aubo_caps/domain/io/digital_io.h 文件参考

`#include <aubo_caps/domain/io/io.h>`
digital_io.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::DigitalIo](#)
数字 IO 接口此接口提供对数字 I/O 的支持。
- struct [arcs::aubo_scope::DigitalIoFilter](#)
- struct [arcs::aubo_scope::InputIoFilter](#)
- struct [arcs::aubo_scope::OutputIoFilter](#)

命名空间

- namespace [arcs](#)

- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`DigitalIo`)

12.101 digital_io.h

浏览该文件的文档.

```
00001 #ifndef AUBO_SCOPE_DIGITAL_IO_H
00002 #define AUBO_SCOPE_DIGITAL_IO_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(DigitalIo);
00009
00010 /**
00011  * \chinese
00012  * 数字 IO 接口
00013  * 此接口提供对数字 I/O 的支持。
00014  * \endchinese
00015  * \english
00016  * DigitalIo
00017  * This interface provides support for digital I/Os.
00018  * \endenglish
00019  */
00020 class ARCS_ABI_EXPORT DigitalIo : public Io
00021 {
00022 public:
00023     DigitalIo(DigitalIo &f);
00024     DigitalIo(DigitalIo &&f);
00025     ~DigitalIo();
00026
00027     /**
00028     * \chinese
00029     * 设置数字 I/O 输出信号。注意该 I/O 必须是输出类型。
00030     * @param high 要设置的数字值。
00031     * @return 如果数字信号设置成功返回 <code>true</code>, 否则返回 <code>false</code>。
00032     * \endchinese
00033     * \english
00034     * Set digital I/O output signal. Note that the I/O must be an output I/O.
00035     *
00036     * @param high the digital value to be set.
00037     * @return <code>true</code> if the digital signal was set,
00038     * <code>false</code> if the signal was not set, e.g. because the controller
00039     * was not running.
00040     * \endenglish
00041     */
00042     bool setValue(bool high);
00043
00044     /**
00045     * \chinese
00046     * 获取 I/O 的最后一次读数
00047     * @return I/O 的最后一次读数。
00048     * \endchinese
00049     * \english
00050     * Returns the last reading of the I/O.
00051     * @return Returns the last reading of the I/O.
00052     * \endenglish
00053     */
00054     bool getValue();
00055
00056 private:
00057     friend class DataSwitch;
00058     DigitalIo();
00059     void *d_{ nullptr };
00060 };
00061
00062 struct DigitalIoFilter
00063 {
00064     DigitalIoFilter(Io::InterfaceType interface, bool is_input)
00065         : interface_(interface), is_input_(is_input)
00066     {
00067     }
00068     bool operator()(IoPtr io)
00069     {
00070         return std::dynamic_pointer_cast<DigitalIo>(io) != nullptr &&
00071             io->getInterfaceType() == interface_ &&
00072             io->isInput() == is_input_;
00073     }
00074 }
```

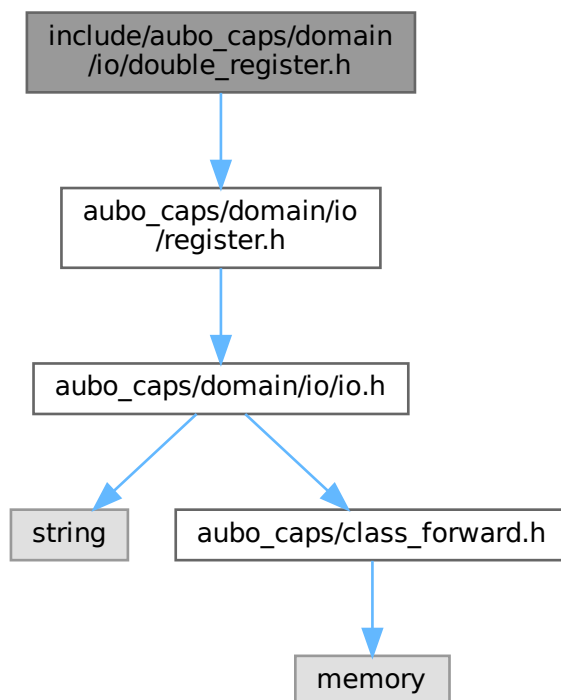
```

00074
00075 private:
00076     Io::InterfaceType interface_;
00077     bool is_input_;
00078 };
00079
00080 struct InputIoFilter
00081 {
00082     bool operator()(IoPtr io) { return io && io->isInput(); }
00083 };
00084
00085 struct OutputIoFilter
00086 {
00087     bool operator()(IoPtr io) { return io && !io->isInput(); }
00088 };
00089
00090 } // namespace aubo_scope
00091 } // namespace arcs
00092
00093 #endif // AUBO_SCOPE_DIGITAL_IO_H

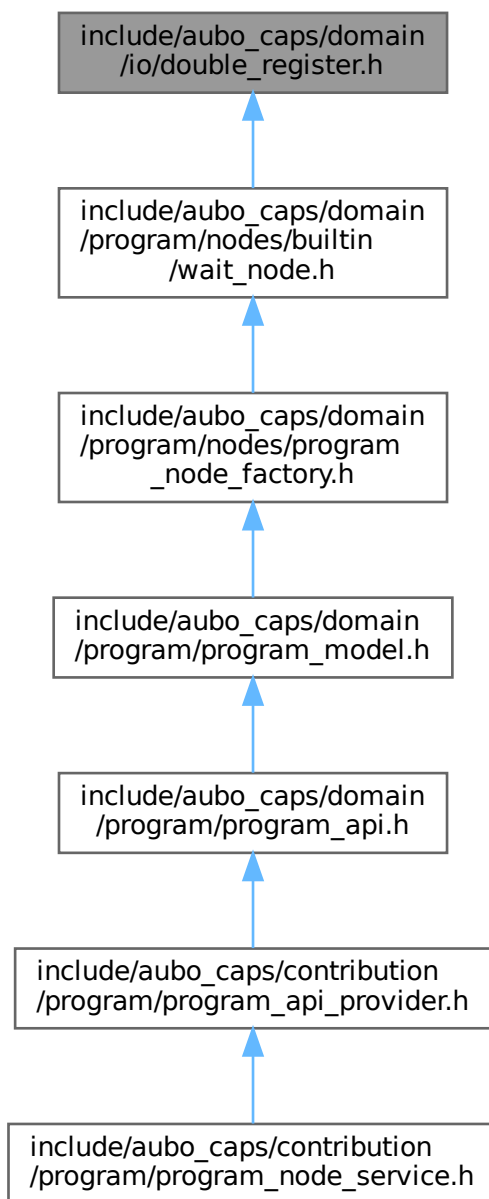
```

12.102 include/auco_caps/domain/io/double_register.h 文件参考

#include <auco_caps/domain/io/register.h>
double_register.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::DoubleRegister](#)

This interface provides support for float registers.

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (DoubleRegister)`

12.103 double__register.h

[浏览该文件的文档.](#)

```

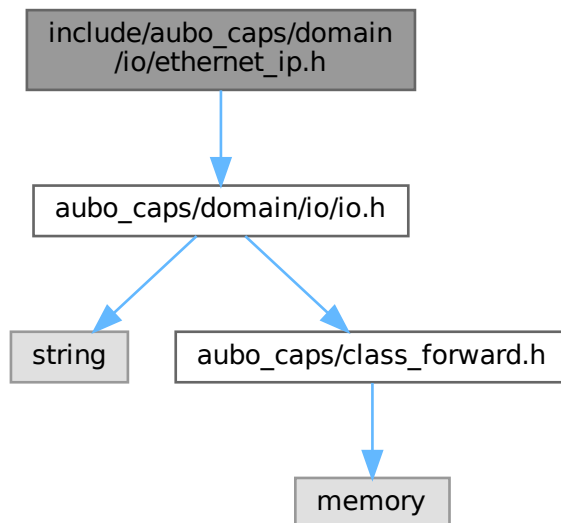
00001 #ifndef AUBO_SCOPE_DOUBLE_REGISTER_H
00002 #define AUBO_SCOPE_DOUBLE_REGISTER_H
00003
00004 #include <aubo_caps/domain/io/register.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(DoubleRegister);
00009
00010 /**
00011  * This interface provides support for float registers.
00012  *
00013  */
00014     class ARCS_ABI_EXPORT DoubleRegister : public Register
00015     {
00016     public:
00017         virtual ~DoubleRegister() = default;
00018     };
00019
00020 } // namespace aubo_scope
00021 } // namespace arcs
00022
00023 #endif // AUBO_SCOPE_DOUBLE_REGISTER_H

```

12.104 include/aubo_caps/domain/io/ethernet_ip.h 文件参考

```
#include <aubo_caps/domain/io/io.h>
```

ethernet_ip.h 的引用 (Include) 关系图:



类

- class `arcs::aubo_scope::EtherNetIP`

EtherNet/IP IO 接口此接口提供对 *EtherNet/IP* I/O 的支持。

- struct `arcs::aubo_scope::EtherNetIPFilter`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`EtherNetIP`)

12.105 ethernet_ip.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_ETHERNET_IP_H
00002 #define AUBO_SCOPE_ETHERNET_IP_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(EtherNetIP);
00009
00010         /**
00011          * \chinese
00012          * EtherNet/IP I/O 接口
00013          * 此接口提供对 EtherNet/IP I/O 的支持。
00014          * \endchinese
00015          * \english
00016          * EtherNetIP
00017          * This interface provides support for EtherNet/IP I/Os.
00018          * \endenglish
00019          */
00020         class ARCS_ABI_EXPORT EtherNetIP : public Io
00021         {
00022         public:
00023             EtherNetIP(EtherNetIP &f);
00024             EtherNetIP(EtherNetIP &&f);
00025             ~EtherNetIP();
00026
00027             /**
00028              * \chinese
00029              * 获取此 EtherNet/IP I/O 的字节偏移量
00030              * @return 字节偏移量。
00031              * \endchinese
00032              * \english
00033              * Returns the byte offset for this EtherNet/IP I/O.
00034              * @return Get the byte offset for this EtherNet/IP I/O.
00035              * \endenglish
00036              */
00037             int getByteOffset();
00038
00039             /**
00040              * \chinese
00041              * 获取此 EtherNet/IP I/O 的位偏移量
00042              * @return 位偏移量。
00043              * \endchinese
00044              * \english
00045              * Returns the bit offset for this EtherNet/IP I/O.
00046              * @return Get the bit offset for this EtherNet/IP I/O.
00047              * \endenglish
00048              */
00049             int getBitOffset();
00050
00051             /**
00052              * \chinese
00053              * 判断此 EtherNet/IP I/O 是否为从站
00054              * @return 是否为从站。
00055              * \endchinese
00056              * \english
00057              * Returns whether this EtherNet/IP I/O is a slave.
00058              * @return Get the is slave for this EtherNet/IP I/O.
00059              * \endenglish
00060              */
00061             bool isSlave();
00062
00063             /**
00064              * \chinese
00065              * 设置 EtherNet/IP 信号的整数值。
00066              * @param value 要设置的整数值。

```

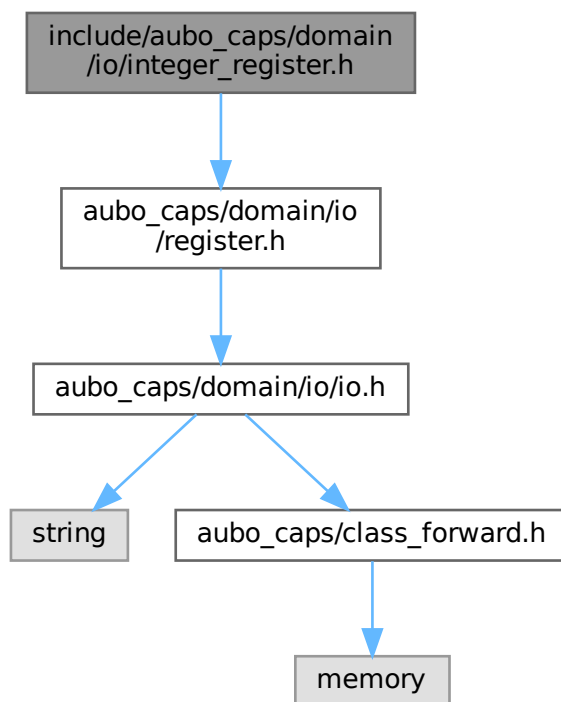
```

00067      * @return 信号下发成功返回 <code>true</code>, 否则返回 <code>false</code>。
00068      * \endchinese
00069      * \english
00070      * Set EtherNet/IP signal to an integer value.
00071      * @param value Set EtherNet/IP signal to an integer value.
00072      * @return <code>true</code> if the EtherNet/IP signal was sent down to the
00073      * controller, and <code>false</code> if the signal was not sent down.
00074      * \endenglish
00075      */
00076      bool setValue(int value);
00077
00078      /**
00079      * \chinese
00080      * 获取 EtherNet/IP 信号的整数值。
00081      * @return EtherNet/IP 信号的整数值。
00082      * \endchinese
00083      * \english
00084      * Returns the integer value of the EtherNet/IP signal.
00085      * @return the integer value of the EtherNet/IP signal.
00086      * \endenglish
00087      */
00088      int getValue();
00089
00090 private:
00091     friend class DataSwitch;
00092     EtherNetIP();
00093     void *d_{ nullptr };
00094 };
00095
00096 struct EtherNetIPFilter
00097 {
00098     EtherNetIPFilter(Io::IoType type, bool is_input)
00099         : type_(type), is_input_(is_input)
00100     {
00101     }
00102     bool operator()(IoPtr io)
00103     {
00104         return std::dynamic_pointer_cast<EtherNetIP>(io) != nullptr &&
00105             io->getType() == type_ && io->isInput() == is_input_;
00106     }
00107
00108 private:
00109     Io::IoType type_;
00110     bool is_input_;
00111 };
00112
00113 } // namespace aubo_scope
00114 } // namespace arcs
00115
00116 #endif // EtherNet/IPIO_H

```


12.106 include/aubo_caps/domain/io/integer_register.h 文件参考

`#include <aubo_caps/domain/io/register.h>`
integer_register.h 的引用 (Include) 关系图:



类

- class `arcs::aubo_scope::IntegerRegister`
This interface provides support for integer registers.

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (IntegerRegister)`

12.107 integer_register.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_INTEGER_REGISTER_H
00002 #define AUBO_SCOPE_INTEGER_REGISTER_H
00003
00004 #include <aubo_caps/domain/io/register.h>
00005
00006 namespace arcs {

```

```

00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(IntegerRegister);
00009
00010 /**
00011  * This interface provides support for integer registers.
00012  *
00013  */
00014 class ARCS_ABI_EXPORT IntegerRegister : public Register
00015 {
00016 public:
00017     virtual ~IntegerRegister() = default;
00018 };
00019
00020 } // namespace aubo_scope
00021 } // namespace arcs
00022
00023 #endif // AUBO_SCOPE_INTEGER_REGISTER_H

```

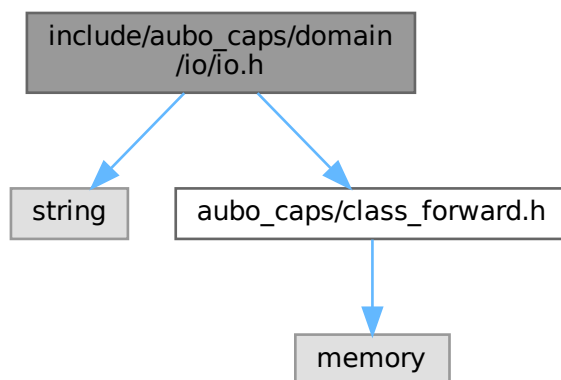
12.108 include/aubo_caps/domain/io/io.h 文件参考

```

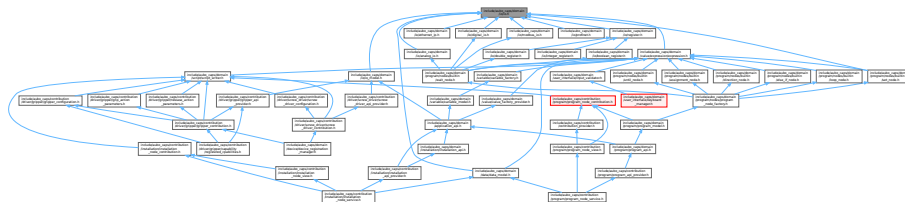
#include <string>
#include <aubo_caps/class_forward.h>

```

io.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::Io`
IO 基础接口这是表示 *AuboScope* 中所有可用 I/O 类型的基础接口。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(Io\)](#)

12.109 io.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_IO_H
00002 #define AUBO_SCOPE_IO_H
00003
00004 #include <string>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(Io);
00010
00011 /**
00012  * \chinese
00013  * IO 基础接口
00014  * 这是表示 AuboScope 中所有可用 I/O 类型的基础接口。
00015  * \endchinese
00016  * \english
00017  * Io
00018  * This is the base interface representing all types of I/Os available in
00019  * AuboScope.
00020  * \endenglish
00021  */
00022 class ARCS_ABI_EXPORT Io
00023 {
00024 public:
00025     enum IoType : int
00026     {
00027         DIGITAL,
00028         ANALOG,
00029         INTEGER,
00030         BOOLEAN,
00031         DOUBLE
00032     };
00033     enum InterfaceType : int
00034     {
00035         STANDARD,
00036         TOOL,
00037         CONFIGURABLE,
00038         LINKIO,
00039         MODBUS,
00040         GENERAL_PURPOSE,
00041         PROFINET,
00042         ETHERNET_IP,
00043     };
00044
00045 public:
00046     Io(Io &f);
00047     Io(Io &&f);
00048     virtual ~Io();
00049
00050     /**
00051     * \chinese
00052     * 获取当前安装中 I/O 的用户自定义名称
00053     * @return I/O 的显示名称。
00054     * \endchinese
00055     * \english
00056     * Returns the user defined name of the IO in the currently selected
00057     * installation.
00058     * @return the user defined name of the IO.
00059     * \endenglish
00060     */
00061     std::string getDisplayName() const;
00062
00063     /**
00064     * \chinese
00065     * 获取 I/O 的机器人默认名称
00066     * @return I/O 的默认名称。
00067     * \endchinese
00068     * \english
00069     * Returns the robot default name of the I/O.
00070     * @return the robot default name of the I/O.
00071     * \endenglish
00072     */
00073     std::string getDefaultName() const;
00074
00075     /**
00076     * \chinese
00077     * 获取 I/O 的当前读数值 (字符串形式)

```

```

00078     * @return I/O 的当前读数。
00079     * \endchinese
00080     * \english
00081     * Returns the current reading of the I/O as a string.
00082     * @return get current reading of the I/O as a string.
00083     * \endenglish
00084     */
00085     std::string getValueStr() const;
00086
00087     /**
00088     * \chinese
00089     * 获取 I/O 类型
00090     * @return I/O 类型枚举。
00091     * \endchinese
00092     * \english
00093     * Returns the I/O type.
00094     * @return the I/O type.
00095     * \endenglish
00096     */
00097     IoType getType() const;
00098
00099     /**
00100     * \chinese
00101     * 获取此 I/O 所属的接口类型
00102     * @return 接口类型枚举。
00103     * \endchinese
00104     * \english
00105     * Returns the interface type this I/O belongs to.
00106     * @return the interface type this I/O belongs to.
00107     * \endenglish
00108     */
00109     InterfaceType getInterfaceType() const;
00110
00111     /**
00112     * \chinese
00113     * 获取 I/O 的动作编号
00114     * @return I/O 的动作编号。
00115     * \endchinese
00116     * \english
00117     * Returns the action number of the I/O.
00118     * @return the action number of the I/O.
00119     * \endenglish
00120     */
00121     int getAction() const;
00122
00123     /**
00124     * \chinese
00125     * 判断 I/O 是否支持读取值
00126     * @return 如果 I/O 支持读取返回 <code>true</code>, 否则返回 <code>false</code>。
00127     * \endchinese
00128     * \english
00129     * Returns <code>true</code> if I/O supports reading of values and
00130     * <code>false</code> otherwise.
00131     * @return <code>true</code> if I/O supports reading of values.
00132     * \endenglish
00133     */
00134     bool isInput() const;
00135
00136     /**
00137     * \chinese
00138     * 某些 I/O (如 MODBUS I/O 和模拟工具输入) 在某些情况下可能不可用。此方法用于判断
00139     * I/O 是否可用。
00140     * @return 如果 I/O 可解析返回 <code>true</code>, 否则返回 <code>false</code>。
00141     * \endchinese
00142     * \english
00143     * Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some
00144     * situations not present in AuboScope or available for use. This method can
00145     * be used to determine if the I/O is available.
00146     * @return <code>true</code>, if I/O can be resolved.
00147     * <code>false</code> otherwise.
00148     * \endenglish
00149     */
00150     bool isResolvable() const;
00151
00152     size_t hashCode();
00153
00154 protected:
00155     Io();
00156
00157 private:
00158     friend class DataSwitch;
00159     void *d_{ nullptr };
00160 };
00161
00162 } // namespace aubo_scope
00163 } // namespace arcs
00164

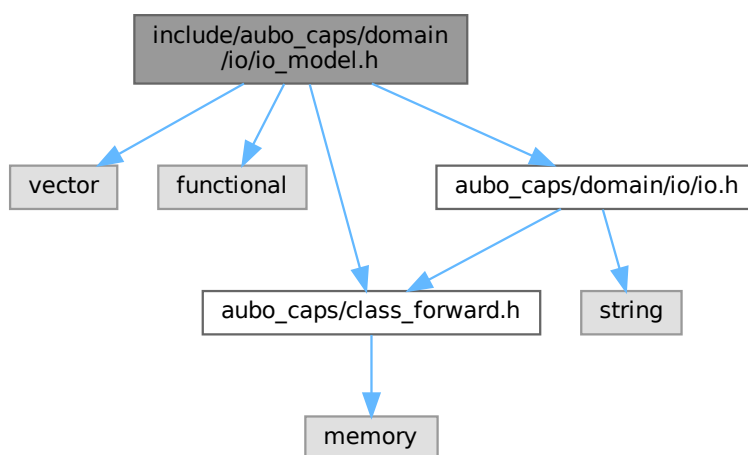
```

```
00165 #endif // AUBO_SCOPE_IO_H
```

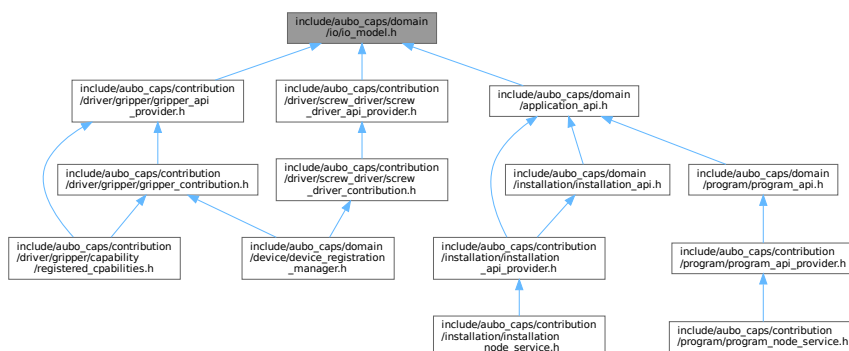
12.110 include/aubo_caps/domain/io/io_model.h 文件参考

```
#include <vector>
#include <functional>
#include <aubo_caps/class_forward.h>
#include <aubo_caps/domain/io/io.h>
```

io_model.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::IoModel](#)
IO 模型提供从当前机器人安装中获取 I/O 的方法。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (IoModel)`

12.111 io__model.h

浏览该文件的文档.

```

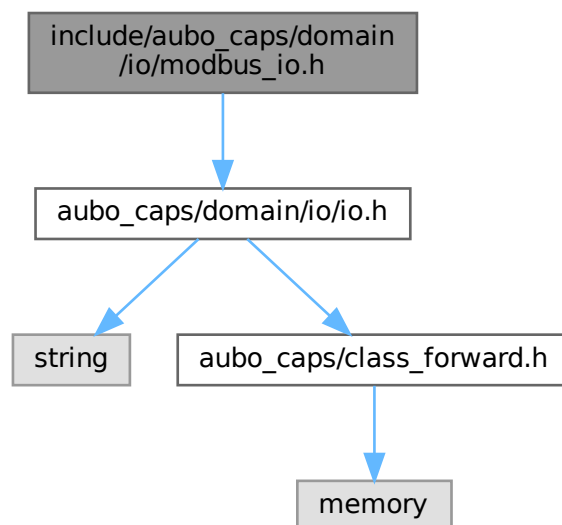
00001 #ifndef AUBO_SCOPE_IO_MODEL_H
00002 #define AUBO_SCOPE_IO_MODEL_H
00003
00004 #include <vector>
00005 #include <functional>
00006 #include <aubo_caps/class_forward.h>
00007 #include <aubo_caps/domain/io/io.h>
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011 ARCS_CLASS_FORWARD(IoModel);
00012
00013 /**
00014  * @ingroup ApplicationApi
00015  * \chinese
00016  * IO 模型
00017  * 提供从当前机器人安装中获取 I/O 的方法。
00018  * \endchinese
00019  * \english
00020  * IoModel
00021  * Provides methods that returns I/Os from the current robot installation.
00022  * \endenglish
00023  */
00024 class ARCS_ABI_EXPORT IoModel
00025 {
00026 public:
00027     IoModel(IoModel &f);
00028     IoModel(IoModel &&f);
00029     ~IoModel();
00030
00031     /**
00032      * @ingroup IoModel
00033      * \chinese
00034      * 获取所有可用的 I/O 集合
00035      * @return 可用 I/O 的集合。
00036      * \endchinese
00037      * \english
00038      * Returns the collection of available I/Os, including Modbus, EuroMap67 and
00039      * named General Purpose Registers.
00040      * @return the collection of available I/Os.
00041      * \endenglish
00042      */
00043     std::vector<IoPtr> getAll();
00044
00045     /**
00046      * @ingroup IoModel
00047      * @ref Io
00048      * \chinese
00049      * 根据名称获取 I/O
00050      * @param name I/O 名称。
00051      * @return 对应名称的 I/O。
00052      * \endchinese
00053      * \english
00054      * Get an I/O by name.
00055      * @param name the name of the I/O.
00056      * @return the I/O with the given name.
00057      * \endenglish
00058      */
00059     IoPtr getIo(const std::string &name);
00060
00061     /**
00062      * @ingroup IoModel
00063      * \chinese
00064      * 使用过滤器获取 I/O 子集
00065      * @param filter 过滤器。
00066      * @return 被过滤器接受的 I/O 集合。
00067      * \endchinese
00068      * \english
00069      * Get a subset of I/Os using a filter.
00070      * @param filter see {@link IOFilterFactory}.
00071      * @return the collection of IOs that are accepted by the filter.
00072      * \endenglish
00073      */
00074     std::vector<IoPtr> getIos(std::function<bool(IoPtr)> filter);
00075
00076 private:
00077     friend class DataSwitch;

```

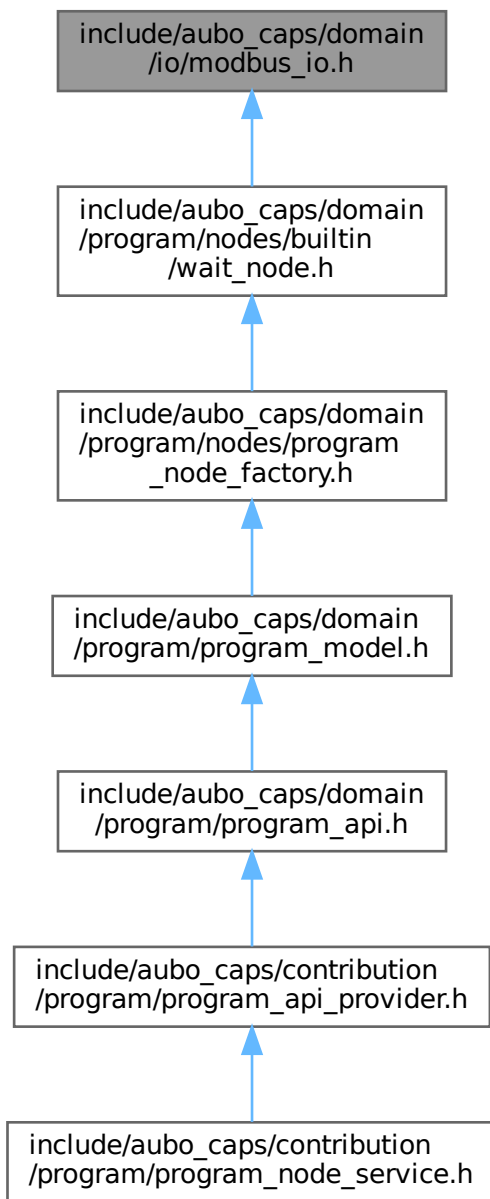
```
00078     IoModel();  
00079     void *d_{ nullptr };  
00080 };  
00081  
00082 } // namespace aubo_scope  
00083 } // namespace arcs  
00084 #endif
```

12.112 include/auco_caps/domain/io/modbus_io.h 文件参考

#include <auco_caps/domain/io/io.h>
modbus_io.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class [arcs::aubo_scope::ModbusIo](#)
*Modbus IO 接口*此接口提供对 *MODBUS I/O* 的支持。
- struct [arcs::aubo_scope::ModbusIoFilter](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ModbusIo\)](#)

12.113 modbus_io.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_MODBUS_IO_H
00002 #define AUBO_SCOPE_MODBUS_IO_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(ModbusIo);
00009
00010 /**
00011 * @ingroup ApplicationApi
00012 * \chinese
00013 * Modbus IO 接口
00014 * 此接口提供对 MODBUS I/O 的支持。
00015 * \endchinese
00016 * \english
00017 * ModbusIo
00018 * This interface provides support for MODBUS I/Os.
00019 * \endenglish
00020 */
00021 class ARCS_ABI_EXPORT ModbusIo : public Io
00022 {
00023 public:
00024     ModbusIo(ModbusIo &f);
00025     ModbusIo(ModbusIo &&f);
00026     ~ModbusIo();
00027
00028     /**
00029      * @ingroup ModbusIo
00030      * \chinese
00031      * 获取此 MODBUS I/O 的 IP 地址
00032      * @return IP 地址。
00033      * \endchinese
00034      * \english
00035      * Returns the IP address for this MODBUS I/O.
00036      * @return Get the IP address for this MODBUS I/O.
00037      * \endenglish
00038      */
00039     std::string getIpAddress();
00040
00041     /**
00042      * @ingroup ModbusIo
00043      * \chinese
00044      * 获取此 MODBUS I/O 的信号地址
00045      * @return 信号地址。
00046      * \endchinese
00047      * \english
00048      * Returns the signal address for this MODBUS I/O.
00049      * @return the signal address for this MODBUS I/O.
00050      * \endenglish
00051      */
00052     std::string getSignalAddress();
00053
00054     /**
00055      * @ingroup ModbusIo
00056      * \chinese
00057      * 判断此 MODBUS I/O 是否为从站
00058      * @return 是否为从站。
00059      * \endchinese
00060      * \english
00061      * Returns whether this MODBUS I/O is a slave.
00062      * @return Get the is slave for this MODBUS I/O.
00063      * \endenglish
00064      */
00065     bool isSlave();
00066
00067     /**
00068      * @ingroup ModbusIo
00069      * \chinese
00070      * 设置 MODBUS 信号的整数值。
00071      * @param value 要设置的整数值。
00072      * @return 信号下发成功返回 <code>true</code>, 否则返回 <code>false</code>。
00073      * \endchinese
00074      * \english
00075      * Set MODBUS signal to an integer value.
00076      * @param value Set MODBUS signal to an integer value.
00077      * @return <code>true</code> if the MODBUS signal was sent down to the

```

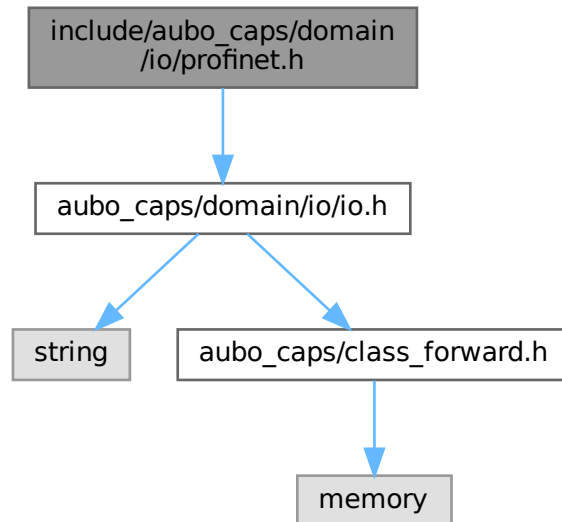
```

00078     * controller, and <code>false</code> if the signal was not sent down.
00079     * \endenglish
00080     */
00081     bool setValue(int value);
00082
00083     /**
00084     * @ingroup ModbusIo
00085     * \chinese
00086     * 获取 MODBUS 信号的整数值。
00087     * @return MODBUS 信号的整数值。
00088     * \endchinese
00089     * \english
00090     * Returns the integer value of the MODBUS signal.
00091     * @return the integer value of the MODBUS signal.
00092     * \endenglish
00093     */
00094     int getValue();
00095
00096 private:
00097     friend class DataSwitch;
00098     ModbusIo();
00099     void *d_{ nullptr };
00100 };
00101
00102 struct ModbusIoFilter
00103 {
00104     ModbusIoFilter(Io::IoType type, bool is_input)
00105         : type_(type), is_input_(is_input)
00106     {
00107     }
00108     bool operator()(IoPtr io)
00109     {
00110         return std::dynamic_pointer_cast<ModbusIo>(io) != nullptr &&
00111             io->getType() == type_ && io->isInput() == is_input_;
00112     }
00113
00114 private:
00115     Io::IoType type_;
00116     bool is_input_;
00117 };
00118
00119 } // namespace aubo_scope
00120 } // namespace arcs
00121
00122 #endif // MODBUSIO_H

```

12.114 include/aubo_caps/domain/io/profinet.h 文件参考

#include <aubo_caps/domain/io/io.h>
profinet.h 的引用 (Include) 关系图:



类

- class `arcs::aubo_scope::Profinet`
PROFINET IO 接口此接口提供对 PROFINET I/O 的支持。
- struct `arcs::aubo_scope::ProfinetFilter`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (Profinet)`

12.115 profinet.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_PROFINET_H
00002 #define AUBO_SCOPE_PROFINET_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(Profinet);
00009     }
00010 /**
00011  * @ingroup ApplicationApi
00012  * \chinese
00013  * PROFINET IO 接口
00014  * 此接口提供对 PROFINET I/O 的支持。
00015  * \endchinese

```

```

00016 * \english
00017 * Profinet
00018 * This interface provides support for PROFINET I/Os.
00019 * \endenglish
00020 */
00021 class ARCS_ABI_EXPORT Profinet : public Io
00022 {
00023 public:
00024     Profinet(Profinet &f);
00025     Profinet(Profinet &&f);
00026     ~Profinet();
00027
00028     /**
00029      * @ingroup Profinet
00030      * \chinese
00031      * 获取此 PROFINET I/O 的字节偏移量
00032      * @return 字节偏移量。
00033      * \endchinese
00034      * \english
00035      * Returns the byte offset for this Profinet I/O.
00036      * @return Get the byte offset for this Profinet I/O.
00037      * \endenglish
00038      */
00039     int getByteOffset();
00040
00041     /**
00042      * @ingroup Profinet
00043      * \chinese
00044      * 获取此 PROFINET I/O 的位偏移量
00045      * @return 位偏移量。
00046      * \endchinese
00047      * \english
00048      * Returns the bit offset for this Profinet I/O.
00049      * @return Get the bit offset for this Profinet I/O.
00050      * \endenglish
00051      */
00052     int getBitOffset();
00053
00054     /**
00055      * @ingroup Profinet
00056      * \chinese
00057      * 判断此 PROFINET I/O 是否为从站
00058      * @return 是否为从站。
00059      * \endchinese
00060      * \english
00061      * Returns whether this Profinet I/O is a slave.
00062      * @return <code>true</code> if this is a slave.
00063      * \endenglish
00064      */
00065     bool isSlave();
00066
00067     /**
00068      * @ingroup Profinet
00069      * \chinese
00070      * 设置 PROFINET 信号的整数值。如果 PROFINET 信号是数字量, 则 <code>false</code> 用 0
00071      * 表示, <code>true</code> 用所有非 0 数字表示。
00072      *
00073      * @param value 要设置的整数值。
00074      * @return 如果 PROFINET 信号已下发到控制器则返回 <code>true</code>, 如果信号未下发
00075      * (例如控制器未运行) 则返回 <code>false</code>。注意 PROFINET 信号必须支持写入,
00076      * 请查阅 Aubo PROFINET 协议。
00077      * \endchinese
00078      * \english
00079      * Set PROFINET signal to an integer value. If the PROFINET signal is
00080      * digital, then <code>false</code> is represented by 0 and
00081      * <code>true</code> are represented by all numbers different from 0.
00082      *
00083      * @param value the integer value to set.
00084      * @return <code>true</code> if the PROFINET signal was sent down to the
00085      * controller, and <code>false</code> if the signal was not sent down, e.g.
00086      * because the controller was not running. Notice that the PROFINET signal
00087      * must support writes, consult the Aubo PROFINET protocol.
00088      * \endenglish
00089      */
00090     bool setValue(int value);
00091
00092     /**
00093      * @ingroup Profinet
00094      * \chinese
00095      * 获取 PROFINET 信号的整数值。如果 PROFINET 信号是数字量, 则 <code>false</code> 用 0
00096      * 表示, <code>true</code> 用值 1 表示。
00097      * @return PROFINET 信号的整数值。
00098      * \endchinese
00099      * \english
00100      * Returns the integer value of the PROFINET signal. If the PROFINET signal
00101      * is digital then <code>false</code> is represented by 0 and
00102      * <code>true</code> is represented by the value 1.

```

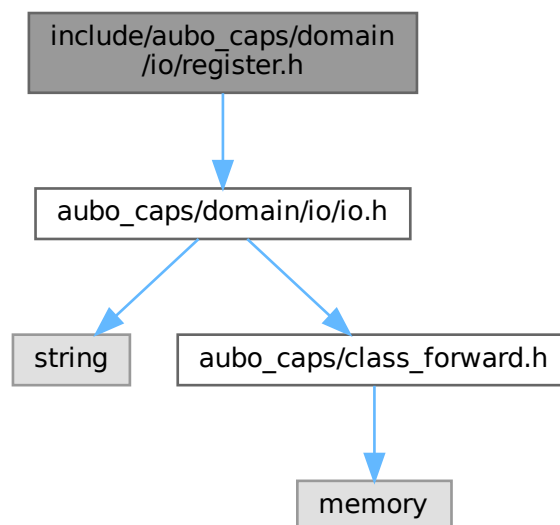
```

00103      * @return the integer value of the PROFINET signal.
00104      * \endenglish
00105      */
00106      int getValue();
00107
00108 private:
00109     friend class DataSwitch;
00110     Profinet();
00111     void *d_{ nullptr };
00112 };
00113
00114 struct ProfinetFilter
00115 {
00116     ProfinetFilter(Io::IoType type, bool is_input)
00117         : type_(type), is_input_(is_input)
00118     {
00119     }
00120     bool operator()(IoPtr io)
00121     {
00122         return std::dynamic_pointer_cast<Profinet>(io) != nullptr &&
00123             io->getType() == type_ && io->isInput() == is_input_;
00124     }
00125
00126 private:
00127     Io::IoType type_;
00128     bool is_input_;
00129 };
00130
00131 } // namespace aubo_scope
00132 } // namespace arcs
00133
00134 #endif // PROFINETIO_H

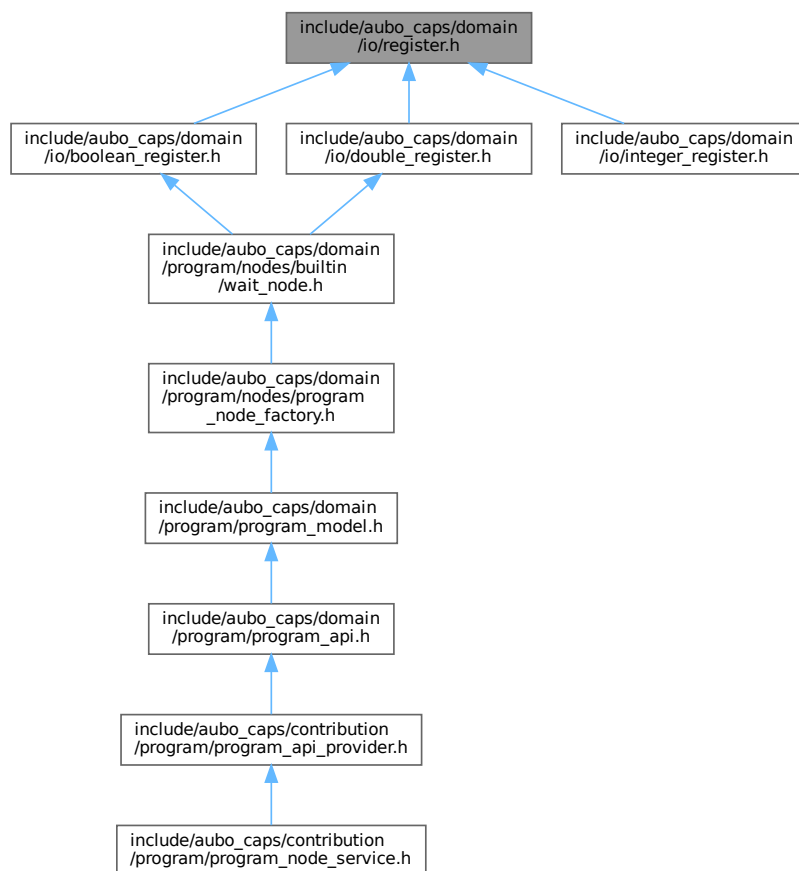
```

12.116 include/auco_caps/domain/io/register.h 文件参考

#include <auco_caps/domain/io/io.h>
register.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::Register`
This interface provides support for registers.

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD(Register)`

12.117 register.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_REGISTER_H
00002 #define AUBO_SCOPE_REGISTER_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(Register);
00009     }
00010 }

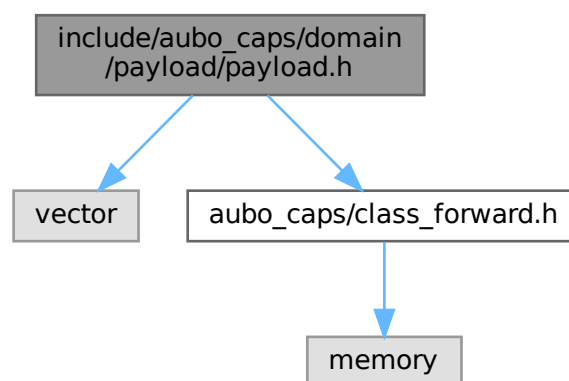
```

```
00010 /**
00011  * This interface provides support for registers.
00012  *
00013  */
00014 class ARCS_ABI_EXPORT Register : public Io
00015 {
00016 public:
00017     virtual ~Register() = default;
00018 };
00019
00020 } // namespace aubo_scope
00021 } // namespace arcs
00022
00023 #endif // REGISTER_H
```

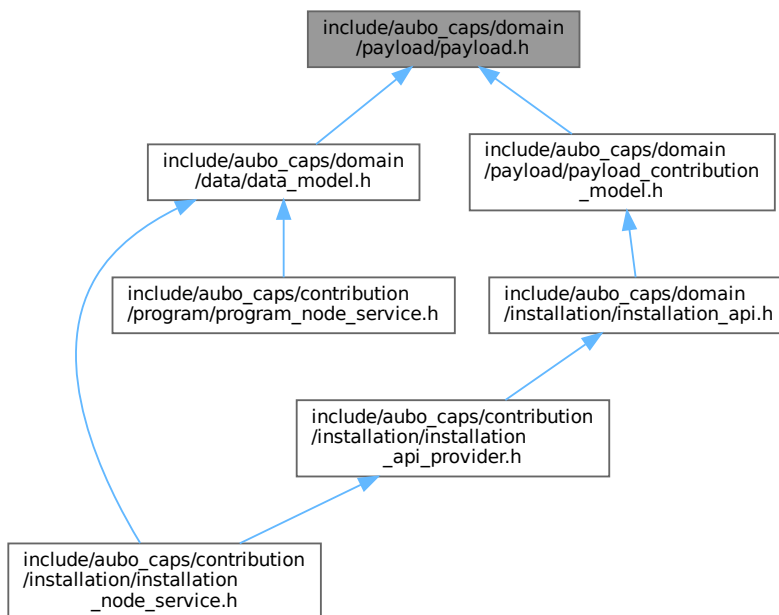
12.118 include/auco_caps/domain/payload/payload.h 文件参考

```
#include <vector>
#include <auco_caps/class_forward.h>
```

payload.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::Payload`
获取负载参数API 接口

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (Payload)`

12.119 payload.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_PAYLOAD_H
00002 #define AUBO_SCOPE_PAYLOAD_H
00003
00004 #include <vector>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(Payload);
00010
00011 /**
00012 * @ingroup InstallationApi
00013 * \~chinese 获取负载参数 API 接口
00014 * \~english API interface for payload parameters
00015 */
00016 class ARCS_ABI_EXPORT Payload
00017 {
00018 public:
00019     Payload(Payload &f);
00020     Payload(Payload &&f);
  
```



```

00021     ~Payload();
00022
00023     /**
00024      * <p>
00025      * This enum represents the different types of warnings that the parameters
00026      * of a payload can have.
00027      * </p>
00028      *
00029      * A payload can be usable, but may have a value for a payload parameter or
00030      * may have a combination of (some of the) payload parameters, that could
00031      * affect performance of the robot, if the payload is used (i.e., if the
00032      * parameters are applied).
00033      */
00034     enum ParametersWarning
00035     {
00036
00037         /**
00038          * None of the parameters of the payload or the combination of payload
00039          * parameters has any warnings.
00040          */
00041         NO_WARNING,
00042
00043         /**
00044          * The payload has a mass of 0 kg which is most likely not
00045          * intended/valid. Note that the value for the payload mass should be
00046          * the total mass of the payload attached to the tool output flange of
00047          * the robot.
00048          */
00049         MASS_OKG,
00050
00051         /**
00052          * The mass of the payload exceeds the rated maximum supported payload
00053          * mass of the robot. Using the payload could affect performance of the
00054          * robot.
00055          */
00056         RATED_MASS_EXCEEDED,
00057
00058         /**
00059          * The combination of the mass and the center of gravity of could affect
00060          * the performance of the robot, if the payload is used.
00061          */
00062         MASS_AND_CENTER_OF_GRAVITY_EXCEEDED,
00063     };
00064
00065     /**
00066      * @ingroup Payload
00067      * @return The total mass of the payload (attached to the tool output flange
00068      * of the robot)
00069      */
00070     double getMass() const;
00071
00072     /**
00073      * Gets the payload's center of gravity (CoG), also referred to as center of
00074      * mass. It is defined as the offset between the center of the tool output
00075      * flange and the center of gravity of the attached payload.
00076      *
00077      * @return The center of gravity (CoG) of the payload relative to the center
00078      * of the tool output flange
00079      */
00080     std::vector<double> getCenterOfGravity() const;
00081
00082     /**
00083      * @ingroup Payload
00084      * <p>
00085      * Gets the inertia matrix of the payload. The inertia matrix is defined in
00086      * a coordinate system with center at the payload's center of gravity (CoG)
00087      * and the axes aligned with the tool output flange coordinate system
00088      * (axes).
00089      * </p>
00090      *
00091      * <b>Note:</b> The inertia matrix specified by the end user can be invalid.
00092      * The {@link #isUsable} method can be used determine, if the payload can be
00093      * used (i.e., has a valid inertia matrix).
00094      *
00095      * @return The inertia matrix of the payload
00096      */
00097     std::vector<double> getInertiaMatrix() const;
00098
00099     /**
00100      * @ingroup Payload
00101      * <p>
00102      * Gets the name of the payload displayed in the AuboScope UI.
00103      * </p>
00104      *
00105      * <b>Note:</b> The payload can change name at any time, and thus the name
00106      * must not be used to uniquely identify a specific payload. The payload
00107      * name can change in the following situations: <ul> <li> The payload is

```

```

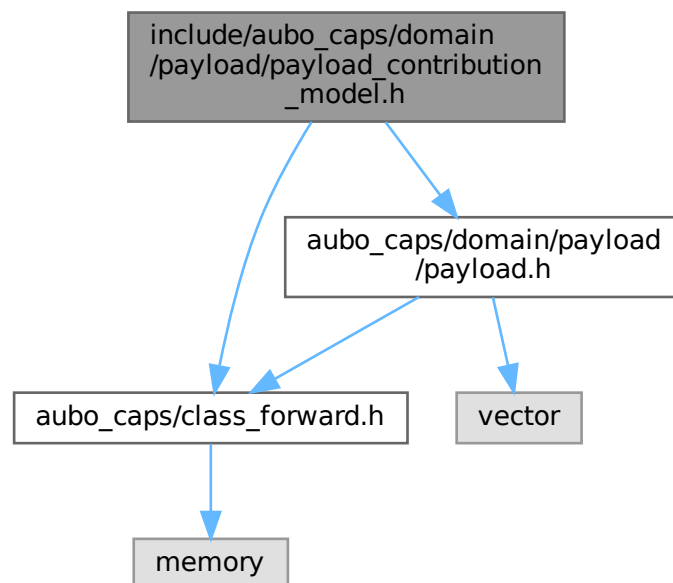
00108     * renamed by the end user </li> <li> The name gets translated (depending on
00109     * the selected language in AuboScope) </li> <li> The name is changed
00110     * between versions of the AuboCap that added the payload</li>
00111     * </ul>
00112     *
00113     * @return The name of the payload displayed in AuboScope
00114     */
00115     std::string getDisplayName() const;
00116
00117     /**
00118     * @ingroup Payload
00119     * <p>
00120     * A payload cannot be guaranteed to be present in AuboScope. This method
00121     * can be used to determine, if the payload is present.
00122     * </p>
00123     *
00124     * <p>
00125     * The payload will not be present, if the end user loads a different
00126     * installation which does not contain the payload, or if the payload is
00127     * removed by the end user or the AuboCap that added the payload.
00128     * </p>
00129     *
00130     * <b>Note:</b> Storing an undefined payload in a program node
00131     * contribution's (a {@link ProgramNodeContribution} instance) data model
00132     * ({@link DataModel} interface) will automatically make the program node
00133     * (and the program) undefined.
00134     *
00135     * @return <code>true</code> if this payload is present in AuboScope,
00136     * otherwise <code>false</code>.
00137     */
00138     bool isResolvable();
00139
00140     /**
00141     * @ingroup Payload
00142     * <p>
00143     * Use this method to determine if the payload can be used. An unresolvable
00144     * payload (see the {@link #isResolvable} method) will always be unusable.
00145     * </p>
00146     *
00147     * <p>
00148     * A payload can have an invalid inertia matrix, where all the individual
00149     * values of the matrix are within the valid range, but the combination of
00150     * values is invalid.
00151     * </p>
00152     *
00153     * <b>Note:</b> Storing an unusable payload in a program node contribution's
00154     * (a {@link ProgramNodeContribution} instance) data model ({@link
00155     * DataModel} interface) will automatically make the program node (and the
00156     * program) undefined.
00157     *
00158     * @return <code>true</code>, if the payload is usable, otherwise
00159     * <code>false</code>. <code>false</code> will always be returned when the
00160     * payload is unresolvable (see the {@link #isResolvable} method).
00161     */
00162     bool isUsable();
00163
00164     /**
00165     * @ingroup Payload
00166     * <p>
00167     * Use this method to check if the parameters of this payload have any
00168     * warnings.
00169     * </p>
00170     *
00171     * <p>
00172     * A payload can be usable, but may have a value for a payload parameter or
00173     * may have a combination of (some of the) payload parameters, that could
00174     * affect performance of the robot, if the payload is used (i.e., if the
00175     * parameters are applied).
00176     * </p>
00177     *
00178     * An unresolvable payload (see {@link #isResolvable()}) will have no
00179     * warning for its parameters (i.e.
00180     * {@link ParametersWarning#NO_WARNING} is returned).
00181     *
00182     * @return The warning (if any) for the parameters of this payload.
00183     */
00184     ParametersWarning getParametersWarning();
00185
00186     size_t hashCode();
00187
00188 private:
00189     friend class DataSwitch;
00190     Payload();
00191     void *d_{ nullptr };
00192 };
00193 } // namespace aubo_scope
00194 } // namespace arcs

```

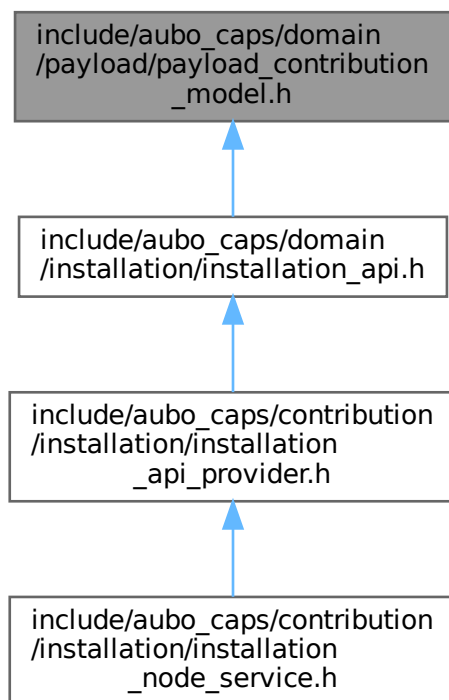
```
00195  
00196 #endif // AUBO_SCOPE_PAYLOAD_H
```

12.120 include/aubo_caps/domain/payload/payload_↵ contribution_model.h 文件参考

```
#include <aubo_caps/class_forward.h>  
#include <aubo_caps/domain/payload/payload.h>  
payload_contribution_model.h 的引用 (Include) 关系图:
```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::PayloadContributionModel`
动态负载管理

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (PayloadContributionModel)`

12.121 payload_contribution_model.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_PAYLOAD_CONTRIBUTION_MODEL_H
00002 #define AUBO_SCOPE_PAYLOAD_CONTRIBUTION_MODEL_H
00003
00004 #include <aubo_caps/class_forward.h>
00005 #include <aubo_caps/domain/payload/payload.h>
00006
00007 namespace arcs {
00008     namespace aubo_scope {
00009         ARCS_CLASS_FORWARD(PayloadContributionModel);
00010     }
00011 }
00012 /**
00013  * @ingroup InstallationApi
00014  * \~chinese 动态负载管理
  
```

```

00014  * \-english API interface for dynamic payload management
00015  */
00016
00017 class ARCS_ABI_EXPORT PayloadContributionModel
00018 {
00019 public:
00020     PayloadContributionModel(PayloadContributionModel &f);
00021     PayloadContributionModel(PayloadContributionModel &&f);
00022     ~PayloadContributionModel();
00023
00024     /**
00025      * @ingroup PayloadContributionModel
00026      * @ref Payload
00027      * \chinese
00028      * <p>
00029      * 向 AuboScope 的当前安装中添加一个负载。
00030      * </p>
00031      *
00032      * <p>
00033      * 添加后终端用户可选择该负载，其他 AuboCap 贡献（通过 {@link PayloadModel} 接口）也可
00034      * 使用。该负载不可被终端用户或其他 aubo_studio 插件修改。
00035      * </p>
00036      *
00037      * <p>
00038      * 系统将根据指定的负载质量和重心自动计算惯性矩阵的默认估算值。如果有负载惯性矩阵
00039      * 的实际数据，可以使用 {@link #addPayload(String, String, Mass, Position,
00040      * InertiaMatrix)} 方法替代（可获得更好的机器人手臂性能）。
00041      * </p>
00042      *
00043      * <p>
00044      * <b> 注意：</b> 指定的负载质量必须是连接到机器人工具输出法兰的负载总质量。
00045      * </p>
00046      *
00047      * 负载质量和重心坐标（CoG）的有效范围可通过 {@link RobotLimits} 接口获取，该接口
00048      * 可通过 {@link RobotModel#getRobotLimits()} 访问。
00049      *
00050      * @param idKey 用于标识此负载的键，不可为 <code>null</code> 或空字符串。该键仅对
00051      * 此 AuboCap 有效，只需在此 AuboCap 中唯一，无需对其他 aubo_studio 插件"全局"唯一。
00052      * @param suggestedName 建议的负载名称，不可为 <code>null</code> 或空字符串。有效
00053      * 名称必须匹配正则表达式 [a-zA-Z][a-zA-Z0-9_]{0,14}，总共 15 个字符。最终名称可
00054      * 从返回的负载实例获取。
00055      * @param mass 连接到机器人工具输出法兰的负载总质量，不可为 <code>null</code>。使
00056      * 用 {@link SimpleValueFactory#createMass(double, Mass.Unit)} 创建。
00057      * @param centerOfGravity 负载的重心（CoG），也称为质心。定义为工具输出法兰中心与
00058      * 所连接负载重心之间的偏移量。不可为 <code>null</code>。使用 {@link PositionFactory}
00059      * 接口创建。
00060      * @return 在 AuboScope 中创建并注册的负载。
00061      * @throws PayloadAlreadyAddedException 如果之前已使用相同的 <code>idKey</code> 标
00062      * 识符添加过负载。使用 {@link #getPayload(String)} 检查负载是否已添加到当前安装中。
00063      * 使用 {@link #updatePayload(String, Mass, Position)} 或
00064      * {@link #updatePayload(String, Mass, Position, InertiaMatrix)} 来更新负载。
00065      * @throws IllegalPayloadNameException 如果建议的名称不匹配所需的正则表达式。
00066      * @throws IllegalMassException 如果指定的质量不在 AuboScope 定义的有效范围内。
00067      * @throws IllegalCenterOfGravityException 如果指定重心的任何值不在 AuboScope 定义
00068      * 的有效范围内。
00069      * \endchinese
00070      * \english
00071      * <p>
00072      * Add a payload to the current installation in AuboScope.
00073      * </p>
00074      *
00075      * <p>
00076      * This makes the payload selectable by the end user as well as available to
00077      * other AuboCap contributions (through the
00078      * {@link PayloadModel} interface). The payload is not modifiable by the end
00079      * user or other aubo_studio plugins.
00080      * </p>
00081      *
00082      * <p>
00083      * A default guess for the inertia matrix for the payload will be computed
00084      * based on the specified payload mass and center of gravity. If data for
00085      * the payload's inertia matrix is available, the method
00086      * {@link #addPayload(String, String, Mass, Position, InertiaMatrix)} can
00087      * be used instead (this might give better performance of the robot arm).
00088      * </p>
00089      *
00090      * <p>
00091      * <b>Note:</b> The specified payload mass must be the total mass of the
00092      * payload attached to the tool output flange of the robot.
00093      * </p>
00094      *
00095      * The valid range for the payload mass and the coordinates of the center of
00096      * gravity (CoG) are available through the
00097      * {@link RobotLimits} interface which can be accessed with {@link
00098      * RobotModel#getRobotLimits()} (an instance of the
00099      * {@link RobotModel} interface can be retrieved through {@link
00100      * SystemAPI#getRobotModel()}).
```

```

00101      *
00102      * @param idKey The key to identify this payload by, not null
00103      * nor an empty string. The key is for this AuboCap only, i.e. it only has
00104      * to be unique for this AuboCap and not "globally" for other aubo_studio
00105      * plugins.
00106      * @param suggestedName Suggested name for the payload, not
00107      * null nor an empty string. Valid names must match regex
00108      * [a-zA-Z][a-zA-Z0-9_]{0,14} for a total of 15 characters. The final name
00109      * can be retrieved from the returned payload instance.
00110      * @param mass The total mass of the payload attached to the tool output
00111      * flange of the robot, not null. Create it using {@link
00112      * SimpleValueFactory#createMass(double, Mass.Unit)}.
00113      * @param centerOfGravity The center of gravity (CoG), also referred to as
00114      * center of mass, for the payload. It is defined as the offset between the
00115      * center of the tool output flange and the center of gravity of the
00116      * attached payload. Cannot be null. Create it using the {@link
00117      * PositionFactory} interface.
00118      * @return The payload created and registered in AuboScope.
00119      * @throws PayloadAlreadyAddedException If a payload has previously been
00120      * added using the same idKey identifier. Use {@link
00121      * #getPayload(String)} to check if the payload has already been added to
00122      * the current installation. Use {@link #updatePayload(String, Mass,
00123      * Position)} or
00124      *      { @link #updatePayload(String, Mass,
00125      * Position, InertiaMatrix)} to update the payload if the payload parameters
00126      * have changed.
00127      * @throws IllegalPayloadNameException If the suggested name does not match
00128      * required regex.
00129      * @throws IllegalMassException If the specified mass is not inside the
00130      * valid range as defined by AuboScope.
00131      * @throws IllegalCenterOfGravityException If any of the values of the
00132      * specified center of gravity are not inside the valid range as defined by
00133      * AuboScope.
00134      * \endenglish
00135      */
00136 PayloadPtr addPayload(const std::string &id_key,
00137                      const std::string &suggested_name, double mass,
00138                      const std::vector<double> &center_of_gravity);
00139
00140 /**
00141  * @ingroup PayloadContributionModel
00142  * @ref Payload
00143  * \chinese
00144  * <p>
00145  * 向 AuboScope 的当前安装中添加一个包含惯性矩阵的负载。
00146  * </p>
00147  *
00148  * <p>
00149  * 添加后终端用户可选择该负载，其他 AuboCap 贡献（通过 {@link PayloadModel} 接口）也可
00150  * 使用。该负载不可被终端用户或其他 aubo_studio 插件修改。
00151  * </p>
00152  *
00153  * 如果没有负载惯性矩阵的数据，请改用 {@link #addPayload(String, String, Mass,
00154  * Position, InertiaMatrix)} 方法。
00155  *
00156  * <p>
00157  * <b> 注意： </b> 指定的负载质量必须是连接到机器人工具输出法兰的负载总质量。
00158  * </p>
00159  *
00160  * 负载质量和重心坐标（CoG）的有效范围可通过 {@link RobotLimits} 接口获取，该接口
00161  * 可通过 {@link RobotModel#getRobotLimits()} 访问。
00162  *
00163  * @param idKey 用于标识此负载的键，不可为 null 或空字符串。该键仅对
00164  * 此 AuboCap 有效。
00165  * @param suggestedName 建议的负载名称，不可为 null 或空字符串。
00166  * @param mass 负载总质量，不可为 null。
00167  * @param centerOfGravity 负载的重心（CoG），不可为 null。
00168  * @param inertiaMatrix 负载的惯性矩阵，定义在以负载重心为中心的坐标系中，坐标轴与
00169  * 工具输出法兰坐标系对齐。不可为 null。
00170  * @return 在 AuboScope 中创建并注册的负载。
00171  * @throws PayloadAlreadyAddedException 如果之前已使用相同的 idKey 标
00172  * 识符添加过负载。
00173  * @throws IllegalPayloadNameException 如果建议的名称不匹配所需的正则表达式。
00174  * @throws IllegalMassException 如果指定的质量不在有效范围内。
00175  * @throws IllegalCenterOfGravityException 如果指定重心的任何值不在有效范围内。
00176  * @throws IllegalInertiaMatrixException 如果指定惯性矩阵的任何值不在有效范围内。
00177  * \endchinese
00178  * \english
00179  * <p>
00180  * Add a payload to the current installation in AuboScope.
00181  * </p>
00182  *
00183  * <p>
00184  * This makes the payload selectable by the end user as well as available to
00185  * other AuboCap contributions (through the
00186  * {@link PayloadModel} interface). The payload is not modifiable by the end
00187  * user or other aubo_studio plugins.

```

```

00188 * </p>
00189 *
00190 * Use the {@link #addPayload(String, String, Mass, Position,
00191 * InertiaMatrix)} method instead if data for the payload's inertia matrix
00192 * is not available.
00193 *
00194 * <p>
00195 * <b>Note:</b> The specified payload mass must be the total mass of the
00196 * payload attached to the tool output flange of the robot.
00197 * </p>
00198 *
00199 * The valid range for the payload mass and the coordinates of the center of
00200 * gravity (CoG) are available through the
00201 * {@link RobotLimits} interface which can be accessed with {@link
00202 * RobotModel#getRobotLimits()} (an instance of the
00203 * {@link RobotModel} interface can be retrieved through {@link
00204 * SystemAPI#getRobotModel()}).
00205 *
00206 * @param idKey The key to identify this payload by, not <code>null</code>
00207 * nor an empty string. The key is for this AuboCap only, i.e. it only has
00208 * to be unique for this AuboCap and not "globally" for other aubo_studio
00209 * plugins.
00210 * @param suggestedName Suggested name for the payload, not
00211 * <code>null</code> nor an empty string. Valid names must match regex
00212 * [a-zA-Z][a-zA-Z0-9_]{0,14} for a total of 15 characters. The final name
00213 * can be retrieved from the returned payload instance.
00214 * @param mass The total mass of the payload attached to the tool output
00215 * flange of the robot, not <code>null</code>. Create it using {@link
00216 * SimpleValueFactory#createMass(double, Mass.Unit)}.
00217 * @param centerOfGravity The center of gravity (CoG), also referred to as
00218 * center of mass, for the payload. It is defined as the offset between the
00219 * center of the tool output flange and the center of gravity of the
00220 * attached payload. Cannot be <code>null</code>. Create it using the {@link
00221 * PositionFactory} interface.
00222 * @param inertiaMatrix The inertia matrix of the payload, defined in a
00223 * coordinate system with center at the payload's center of gravity and the
00224 * axes aligned with the tool output flange coordinate system (axes). Cannot
00225 * be <code>null</code>. Create it using
00226 * {@link #createInertiaMatrix(double, double, double,
00227 * double, double, double, MomentOfInertia.Unit)}.
00228 * @return The payload created and registered in AuboScope.
00229 * @throws PayloadAlreadyAddedException If a payload has previously been
00230 * added using the same <code>idKey</code> identifier. Use {@link
00231 * #getPayload(String)} to check if the payload has already been added to
00232 * the current installation.
00233 * Use {@link #updatePayload(String, Mass,
00234 * Position)} or
00235 * {@link #updatePayload(String, Mass,
00236 * Position, InertiaMatrix)} to update the payload if the payload parameters
00237 * have changed.
00238 * @throws IllegalPayloadNameException If the suggested name does not match
00239 * required regex.
00240 * @throws IllegalMassException If the specified mass is not inside the
00241 * valid range as defined by AuboScope.
00242 * @throws IllegalCenterOfGravityException If any of the values of the
00243 * specified center of gravity are not inside the valid range as defined by
00244 * AuboScope.
00245 * @throws IllegalInertiaMatrixException If any of the values of the
00246 * specified inertia matrix are not inside the valid range as defined by
00247 * AuboScope.
00248 * \endenglish
00249 */
00250 PayloadPtr addPayload(const std::string &id_key,
00251                      const std::string &suggested_name, double mass,
00252                      const std::vector<double> &center_of_gravity,
00253                      const std::vector<double> &inertiaMatrix);
00254
00255 /**
00256 * @ingroup PayloadContributionModel
00257 * @ref Payload
00258 * \chinese
00259 * 返回此前由此 AuboCap 使用相同 <code>idKey</code> 标识符添加的负载。使用此方法验证
00260 * 负载是否存在于当前安装中。
00261 *
00262 * @param idKey 用于标识此负载的键，不可为 <code>null</code> 或空字符串。
00263 * @return 此前由此 AuboCap 添加的负载。如果当前安装中不存在（指定 <code>idKey</code>
00264 * 标识符的）负载，则返回 <code>null</code>。
00265 * \endchinese
00266 * \english
00267 * Returns the payload previously added by this AuboCap using the same
00268 * <code>idKey</code> identifier. Use this to verify if the payload is
00269 * present in the current installation.
00270 *
00271 * @param idKey The key to identify this payload by, not <code>null</code>
00272 * nor an empty string.
00273 * @return The payload previously added by this AuboCap.
00274 * Returns <code>null</code> if no payload exists in current

```



```

00275     * installation (with the specified <code>idKey</code> identifier).
00276     * \endenglish
00277     */
00278 PayloadPtr getPayload(const std::string &id_key);
00279
00280 /**
00281  * @ingroup PayloadContributionModel
00282  * @ref Payload
00283  * \chinese
00284  * <p>
00285  * 更新此 AuboCap 已添加负载的质量和重心 (CoG)。
00286  * </p>
00287  *
00288  * <b> 注意: </b> 如果已为负载指定了惯性矩阵, 调用此方法将用基于新的负载质量和重心
00289  * 计算的默认估算值替换 (覆盖) 它。为避免覆盖现有惯性矩阵 (或更新现有惯性矩阵),
00290  * 请改用 {@link #updatePayload(String, Mass, Position, InertiaMatrix)} 方法。
00291  *
00292  * @param idKey 用于标识负载的键, 不可为 <code>null</code> 或空字符串。调用此方法
00293  * 前必须已使用相同的键添加过负载。
00294  * @param mass 负载总质量, 不可为 <code>null</code>。使用
00295  * {@link SimpleValueFactory#createMass(double, Mass.Unit)} 创建。
00296  * @param centerOfGravity 负载的重心, 不可为 <code>null</code>。使用
00297  * {@link PositionFactory} 接口创建。
00298  * @throws PayloadNotFoundException 如果当前安装中不存在对应 <code>idKey</code>
00299  * 标识符的负载。
00300  * @throws IllegalMassException 如果指定的质量不在 AuboScope 定义的有效范围内。
00301  * @throws IllegalCenterOfGravityException 如果指定重心的任何值不在有效范围内。
00302  * \endchinese
00303  * \english
00304  * <p>
00305  * Update the mass and center of gravity (CoG) of an existing payload added
00306  * by this AuboCap.
00307  * </p>
00308  *
00309  * <b>Note:</b> If an inertia matrix has been specified for the payload,
00310  * calling this method will replace (overwrite) it with a computed default
00311  * guess based the new specified payload mass and center of gravity. To avoid
00312  * overwriting an existing inertia matrix (or to update an existing inertia
00313  * matrix), instead use the method
00314  * {@link #updatePayload(String, Mass, Position, InertiaMatrix)}.
00315  *
00316  * @param idKey The key to identify the payload, not <code>null</code> nor
00317  * an empty string. A payload must have been added using the same key prior
00318  * to calling this method.
00319  * @param mass The total mass of the payload attached to the tool output
00320  * flange of the robot, not <code>null</code>. Create it using {@link
00321  * SimpleValueFactory#createMass(double, Mass.Unit)}.
00322  * @param centerOfGravity The center of gravity of the payload, not {<code>
00323  * null</code>}. Create it using the
00324  * {@link PositionFactory} interface.
00325  * @throws PayloadNotFoundException If no payload exists with the provided
00326  * <code>idKey</code> identifier in the current installation.
00327  * @throws IllegalMassException If the specified mass is not inside the
00328  * valid range as defined by AuboScope.
00329  * @throws IllegalCenterOfGravityException If any of the values of the
00330  * specified center of gravity are not inside the valid range as defined by
00331  * AuboScope.
00332  * \endenglish
00333  */
00334 void updatePayload(const std::string &id_key, double mass,
00335                  const std::vector<double> &center_of_gravity);
00336
00337 /**
00338  * @ingroup PayloadContributionModel
00339  * \chinese
00340  * 更新此 AuboCap 已添加负载的质量、重心 (CoG) 和惯性矩阵。
00341  *
00342  * @param idKey 用于标识负载的键, 不可为 <code>null</code> 或空字符串。调用此方法
00343  * 前必须已使用相同的键添加过负载。
00344  * @param mass 负载总质量, 不可为 <code>null</code>。使用
00345  * {@link SimpleValueFactory#createMass(double, Mass.Unit)} 创建。
00346  * @param centerOfGravity 负载的重心, 不可为 <code>null</code>。使用
00347  * {@link PositionFactory} 接口创建。
00348  * @param inertiaMatrix 负载的惯性矩阵, 不可为 <code>null</code>。
00349  * @throws PayloadNotFoundException 如果当前安装中不存在对应 <code>idKey</code>
00350  * 标识符的负载。
00351  * @throws IllegalMassException 如果指定的质量不在有效范围内。
00352  * @throws IllegalCenterOfGravityException 如果指定重心的任何值不在有效范围内。
00353  * @throws IllegalInertiaMatrixException 如果指定惯性矩阵的任何值不在有效范围内。
00354  * \endchinese
00355  * \english
00356  * Update the mass, center of gravity (CoG) and inertia matrix of an
00357  * existing payload added by this AuboCap.
00358  *
00359  * @param idKey The key to identify the payload, not <code>null</code> nor
00360  * an empty string. A payload must have been added using the same key prior
00361  * to calling this method.

```



```

00362      * @param mass The total mass of the payload attached to the tool output
00363      * flange of the robot, not null. Create it using {@link
00364      * SimpleValueFactory#createMass(double, Mass.Unit)}.
00365      * @param centerOfGravity The center of gravity of the payload, not {code
00366      * null}. Create it using the
00367      * {@link PositionFactory} interface.
00368      * @param inertiaMatrix The inertia matrix of the payload, not
00369      * null. Create it using
00370      * {@link #createInertiaMatrix(double, double, double,
00371      * double, double, double, MomentOfInertia.Unit)}
00372      * @throws PayloadNotFoundException If no payload exists with the provided
00373      * idKey identifier in the current installation.
00374      * @throws IllegalMassException If the specified mass is not inside the
00375      * valid range as defined by AuboScope.
00376      * @throws IllegalCenterOfGravityException If any of the values of the
00377      * specified center of gravity are not inside the valid range as defined by
00378      * AuboScope.
00379      * @throws IllegalInertiaMatrixException If any of the values of the
00380      * specified inertia matrix are not inside the valid range as defined by
00381      * AuboScope.
00382      * \endenglish
00383      */
00384      void updatePayload(const std::string id_key, double mass,
00385                        const std::vector<double> &center_of_gravity,
00386                        const std::vector<double> &inertia_matrix);
00387
00388      /**
00389      * @ingroup PayloadContributionModel
00390      * \chinese
00391      * 从此 AuboCap 中移除已添加的负载。使用该负载的程序节点将变为未定义状态，因为负载
00392      * 不再可解析。
00393      *
00394      * @param idKey 添加负载时使用的标识符键，不可为 null 或空字符串。
00395      * @throws PayloadNotFoundException 如果当前安装中不存在对应 idKey
00396      * 标识符的负载。
00397      * \endchinese
00398      * \english
00399      * Remove a payload added by this AuboCap from AuboScope. Program nodes
00400      * using the payload will be become undefined because the payload is no
00401      * longer resolvable.
00402      *
00403      * @param idKey The identifier key used to add the payload with, not {code
00404      * null} nor an empty string.
00405      * @throws PayloadNotFoundException If no payload exists with the provided
00406      * idKey identifier in the current installation.
00407      * \endenglish
00408      */
00409      void removePayload(const std::string &id_key);
00410
00411     private:
00412         friend class DataSwitch;
00413         PayloadContributionModel();
00414         void *d_{ nullptr };
00415     };
00416 } // namespace aubo_scope
00417 } // namespace arcs
00418
00419 #endif // AUBO_SCOPE_PAYLOAD_CONTRIBUTION_MODEL_H

```

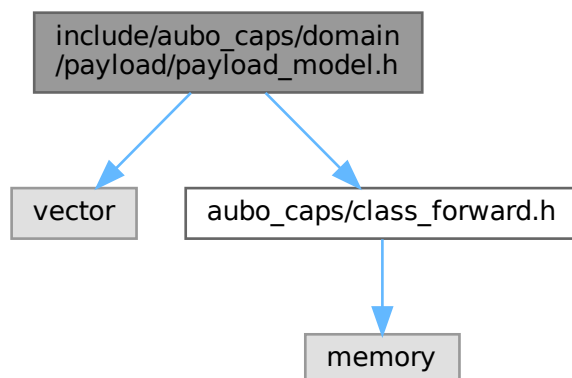
12.122 include/auco_caps/domain/payload/payload_model.h 文件参考

```

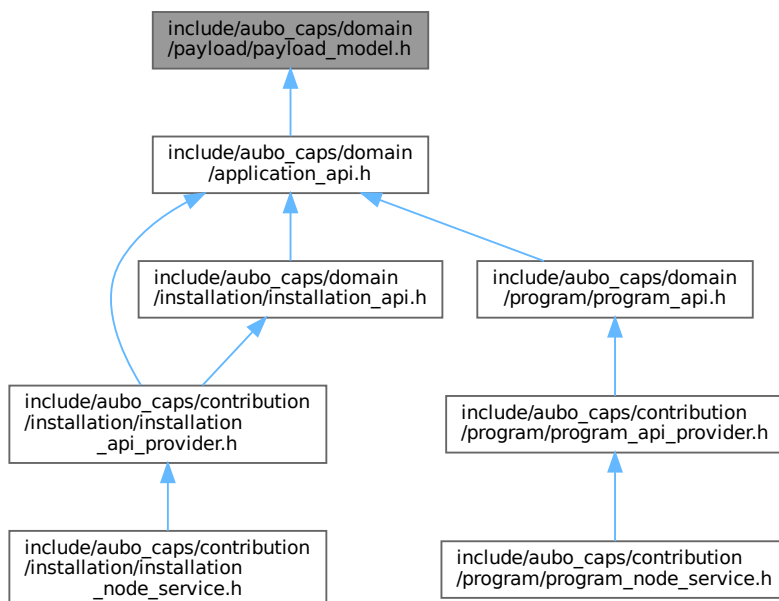
#include <vector>
#include <auco_caps/class_forward.h>

```

payload_model.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::PayloadModel`
获取负载参数API 接口

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- arcs::aubo_scope::ARCS_CLASS_FORWARD (PayloadModel)
- arcs::aubo_scope::ARCS_CLASS_FORWARD (Payload)

12.123 payload_model.h

[浏览该文件的文档.](#)

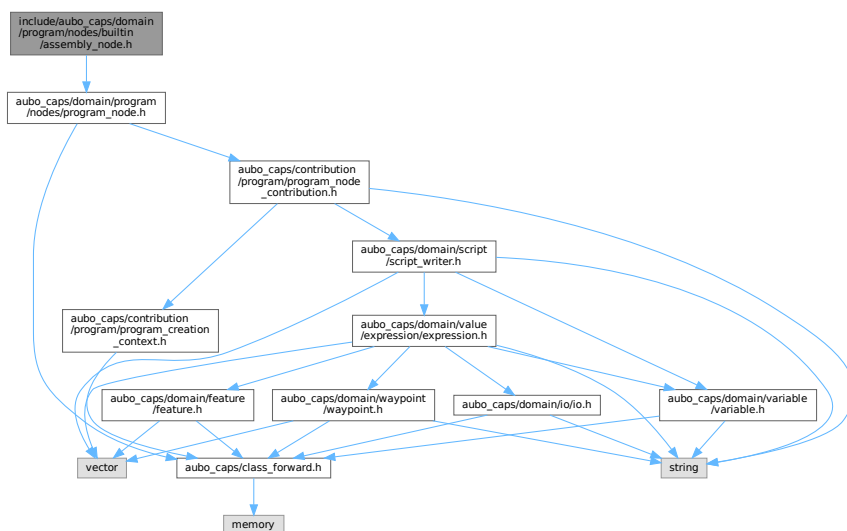
```

00001 #ifndef AUBO_SCOPE_PAYLOAD_MODEL_H
00002 #define AUBO_SCOPE_PAYLOAD_MODEL_H
00003
00004 #include <vector>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008     namespace aubo_scope {
00009         ARCS_CLASS_FORWARD(PayloadModel);
00010         ARCS_CLASS_FORWARD(Payload);
00011
00012         /**
00013          * @ingroup ApplicationApi
00014          * \-chinese 获取负载参数 API 接口
00015          * \-english API interface for payload parameters
00016          */
00017
00018         class ARCS_ABI_EXPORT PayloadModel
00019         {
00020         public:
00021             PayloadModel(PayloadModel &f);
00022             PayloadModel(PayloadModel &&f);
00023             virtual ~PayloadModel();
00024
00025             /**
00026              * @ingroup PayloadModel
00027              * @brief getPayloads
00028              * @return A collection of all the payloads from the current
00029              *         installation in AuboScope, including user-defined payloads and
00030              *         payloads added by aubo_studio plugins.
00031              */
00032             std::vector<PayloadPtr> getPayloads() const;
00033
00034             /**
00035              * @ingroup PayloadModel
00036              * @ref Payload
00037              * @brief getPayload
00038              * @param name
00039              * @return
00040              */
00041             PayloadPtr getPayload(const std::string &name) const;
00042
00043             /**
00044              * @ingroup PayloadModel
00045              * @ref Payload
00046              * @brief getDefaultPayload
00047              * @return
00048              */
00049             PayloadPtr getDefaultPayload() const;
00050
00051             /**
00052              * @ingroup PayloadModel
00053              * @ref Payload
00054              * @brief getCurrentPayload
00055              * @return
00056              */
00057             PayloadPtr getCurrentPayload() const;
00058
00059         private:
00060             friend class DataSwitch;
00061             PayloadModel();
00062             void *d_{ nullptr };
00063         };
00064     } // namespace aubo_scope
00065 } // namespace arcs
00066
00067 #endif // PAYLOADMODEL_H

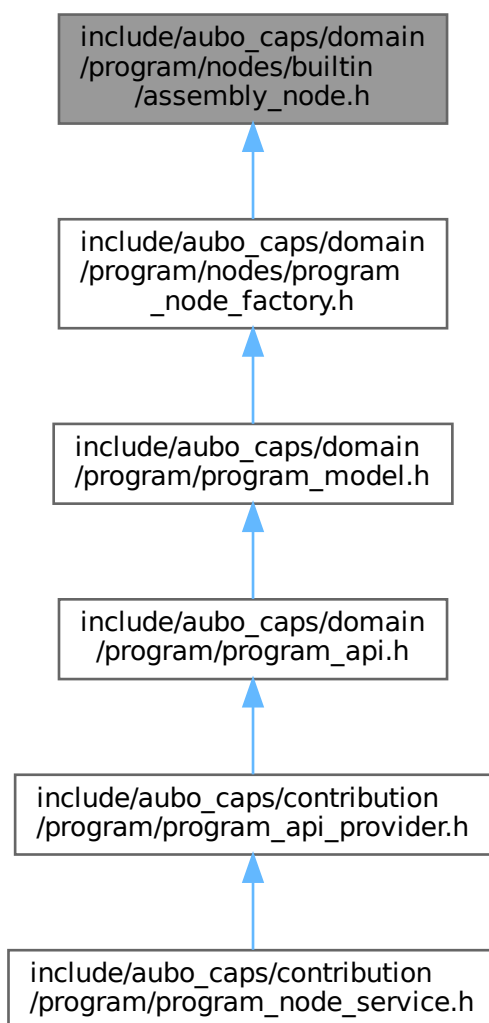
```

12.124 include/aubo_caps/domain/program/nodes/builtin/assembly_node.h 文件参考

`#include <aubo_caps/domain/program/nodes/program_node.h>`
 assembly_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::AssemblyNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(AssemblyNode\)](#)

12.125 assembly__node.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_ASSEMBLY_NODE_H
00002 #define AUBO_SCOPE_ASSEMBLY_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(AssemblyNode);
00009
00010         class ARCS_ABI_EXPORT AssemblyNode : public ProgramNode
00011         {
00012         public:
00013             enum ConfigType
00014             {
00015                 NONE,
00016                 HOLE_SEARCH,
00017                 COMPLIANT_INSERTION
00018             };
00019
00020             AssemblyNode(AssemblyNode &f);
00021             AssemblyNode(AssemblyNode &&f);
00022             virtual ~AssemblyNode();
00023
00024             void addAssemblyUntilNode();
00025
00026         private:
00027             friend class DataSwitch;
00028             AssemblyNode();
00029             void *d_{ nullptr };
00030         };
00031
00032     } // namespace aubo_scope
00033 } // namespace arcs
00034 #endif // AUBO_SCOPE_ASSEMBLY_NODE_H

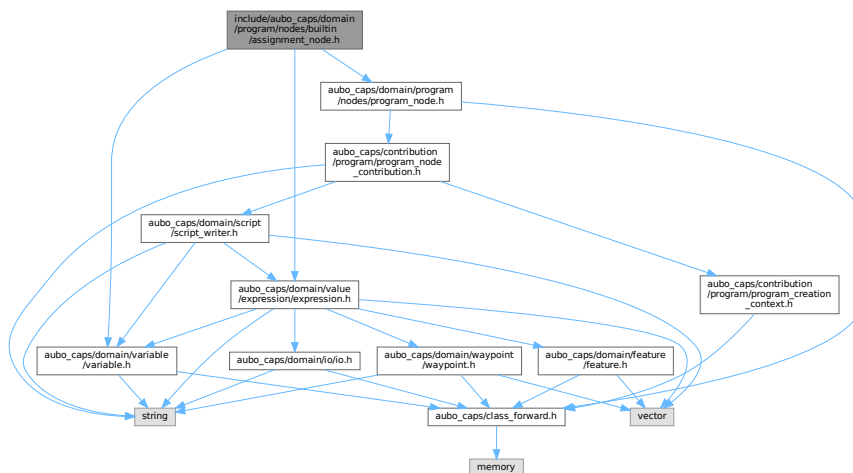
```

12.126 include/aubo_caps/domain/program/nodes/builtin/assignment_node.h 文件参考

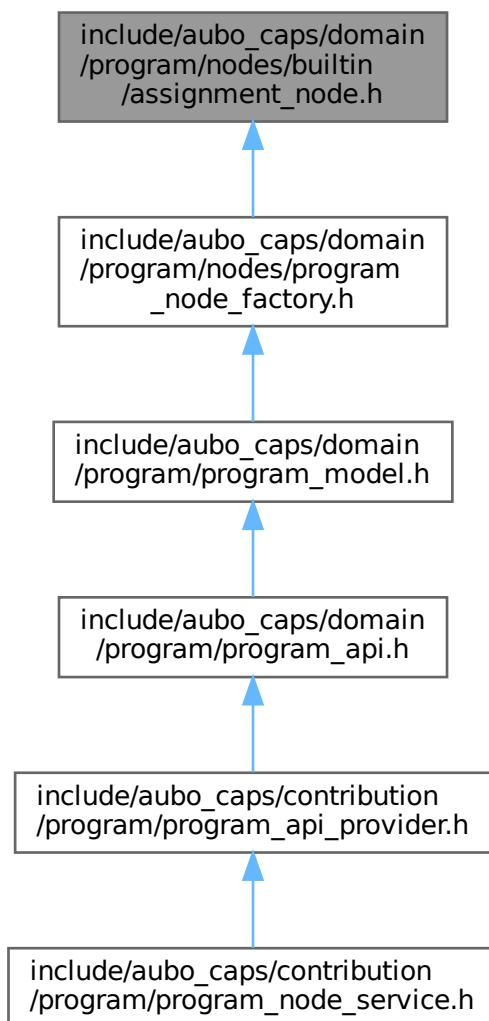
```

#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/variable/variable.h>
#include <aubo_caps/domain/value/expression/expression.h>
assignment_node.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::AssignmentNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(AssignmentNode\)](#)

12.127 assignment_node.h

[浏览该文件的文档.](#)

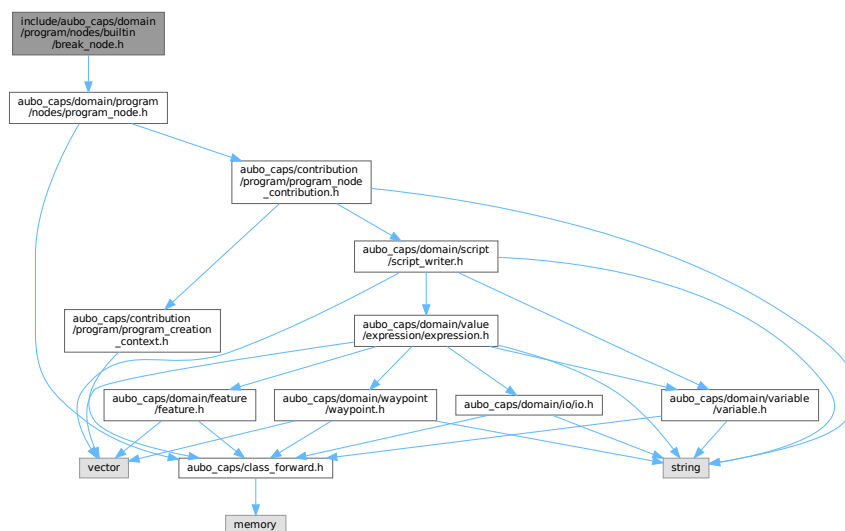
```

00001 #ifndef AUBO_SCOPE_ASSIGNMENT_NODE_H
00002 #define AUBO_SCOPE_ASSIGNMENT_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/variable/variable.h>
00006 #include <aubo_caps/domain/value/expression/expression.h>
00007
00008 namespace arcs {
00009     namespace aubo_scope {
00010         ARCS_CLASS_FORWARD(AssignmentNode);
00011
00012         class ARCS_ABI_EXPORT AssignmentNode : public ProgramNode
00013         {
00014         public:
00015             /**
00016              * The configuration type used to determine which type of configuration this
00017              * instance is.
00018              */
00019             enum ConfigType
00020             {
00021                 /**
00022                  * <p>
00023                  * Expression assignment has been selected.
00024                  * </p>
00025                  *
00026                  * The config instance can be cast to {@link
00027                  * ExpressionAssignmentNodeConfig}.
00028                  */
00029                 EXPRESSION,
00030
00031                 /**
00032                  * Operator assignment has been selected. This type has no further
00033                  * information. Setting this type of config will be ignored and will
00034                  * have no effect on the node.
00035                  */
00036                 OPERATOR
00037             };
00038
00039             enum InputMode
00040             {
00041                 INPUT_BOOL,
00042                 INPUT_INTEGER,
00043                 INPUT_FLOAT,
00044                 INPUT_STRING,
00045             };
00046
00047             AssignmentNode(AssignmentNode &f);
00048             AssignmentNode(AssignmentNode &&f);
00049             virtual ~AssignmentNode();
00050
00051             /**
00052              * <p>
00053              *
00054              * </p>
00055              * @return
00056              */
00057             ConfigType getConfigType();
00058             void setConfigType(ConfigType type);
00059
00060             /**
00061              *
00062              * @return the variable assigned to the expression evaluation.
00063              */
00064             VariablePtr getVariable();
00065             void setVariable(VariablePtr var);
00066
00067             /**
00068              *
00069              * @return the expression whose evaluation is assigned to the variable.
00070              */
00071             ExpressionPtr getExpression();
00072             void setExpression(ExpressionPtr exp);
00073
00074         private:
00075             friend class DataSwitch;
00076             AssignmentNode();
00077             void *d_{ nullptr };
00078         };
00079
00080     } // namespace aubo_scope
00081 } // namespace arcs
00082 #endif // AUBO_SCOPE_ASSIGNMENT_NODE_H

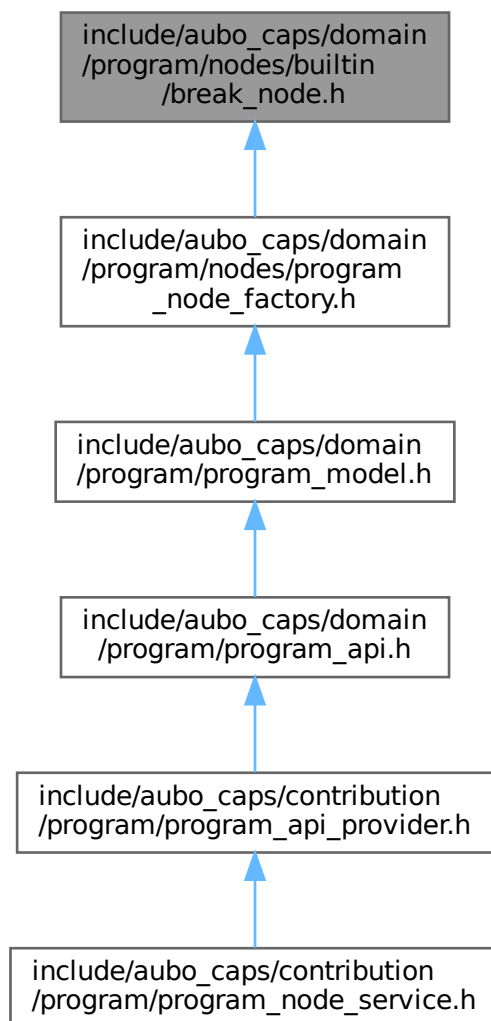
```


12.128 include/aubo_caps/domain/program/nodes/builtin/↵ break_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
break_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::BreakNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(BreakNode\)](#)

12.129 break_node.h

[浏览该文件的文档.](#)

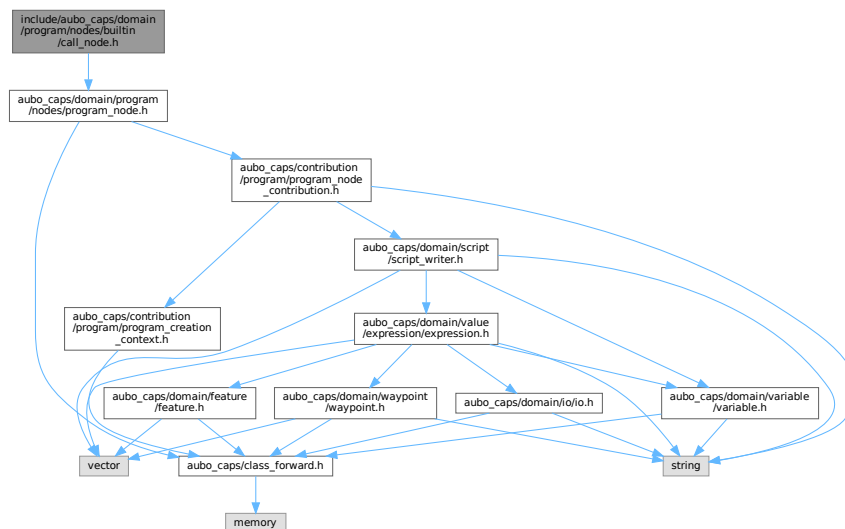
```

00001 #ifndef AUBO_SCOPE_BREAK_NODE_H
00002 #define AUBO_SCOPE_BREAK_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(BreakNode);
00009
00010 class ARCS_ABI_EXPORT BreakNode : public ProgramNode
00011 {
00012 public:
00013     BreakNode(BreakNode &f);
00014     BreakNode(BreakNode &&f);
00015     virtual ~BreakNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     BreakNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_BREAK_NODE_H

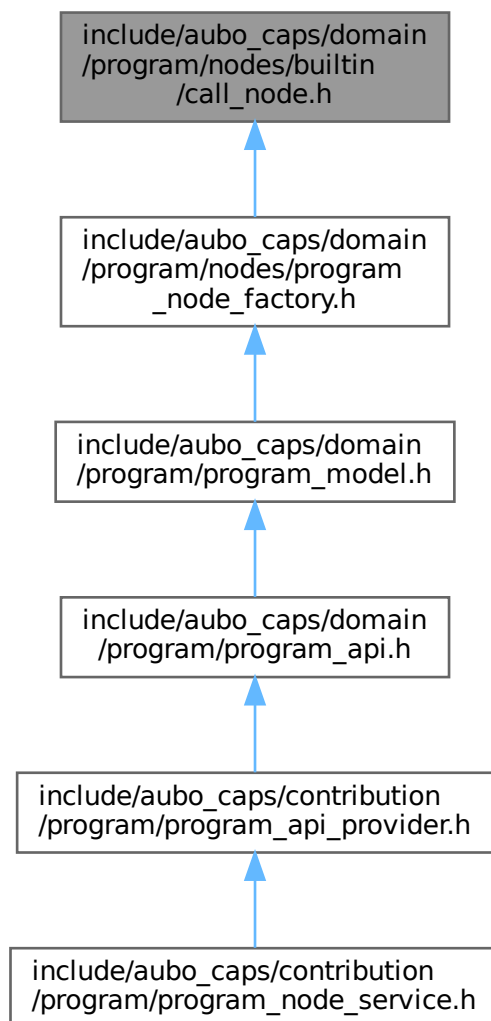
```

12.130 include/aubo_caps/domain/program/nodes/builtin/↵ call_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
call_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::CallNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(CallNode\)](#)

12.131 call_node.h

[浏览该文件的文档.](#)

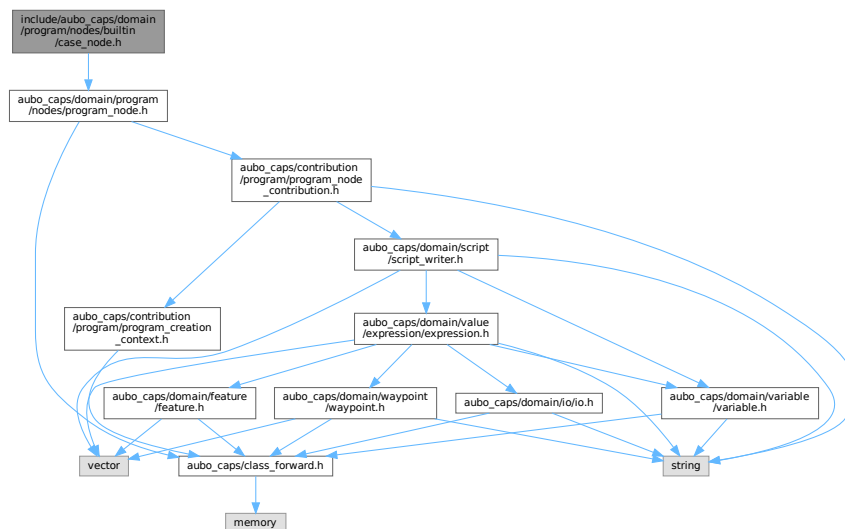
```

00001 #ifndef AUBO_SCOPE_CALL_NODE_H
00002 #define AUBO_SCOPE_CALL_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(CallNode);
00009
00010 class ARCS_ABI_EXPORT CallNode : public ProgramNode
00011 {
00012 public:
00013     CallNode(CallNode &f);
00014     CallNode(CallNode &&f);
00015     virtual ~CallNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     CallNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_CALL_NODE_H

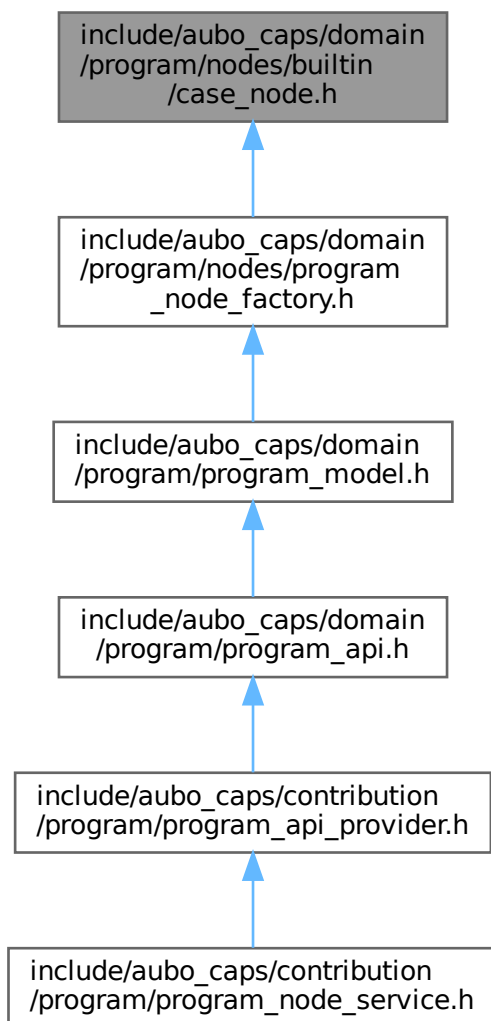
```

12.132 include/aubo_caps/domain/program/nodes/builtin/↵ case_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
case_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::CaseNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(CaseNode\)](#)

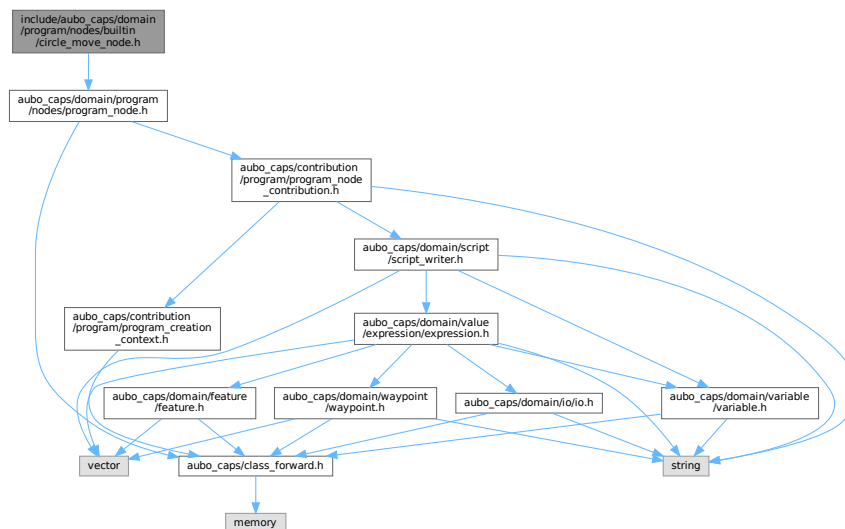
12.133 case_node.h

[浏览该文件的文档.](#)

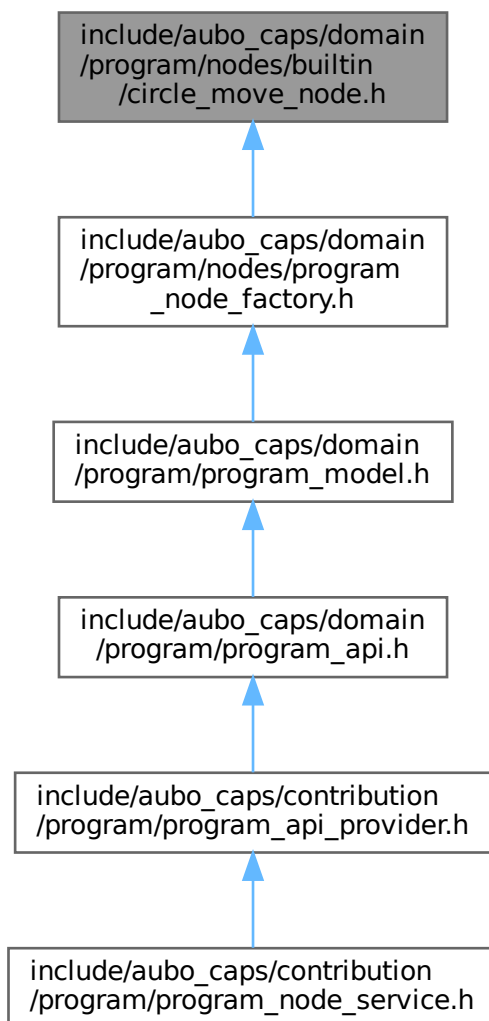
```
00001 #ifndef AUBO_SCOPE_CASE_NODE_H
00002 #define AUBO_SCOPE_CASE_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(CaseNode);
00009
00010 class ARCS_ABI_EXPORT CaseNode : public ProgramNode
00011 {
00012 public:
00013     CaseNode(CaseNode &f);
00014     CaseNode(CaseNode &&f);
00015     virtual ~CaseNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     CaseNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_CASE_NODE_H
```

12.134 include/aubo_caps/domain/program/nodes/builtin/↵ circle_move_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
circle_move_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::CircleMoveNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(CircleMoveNode\)](#)

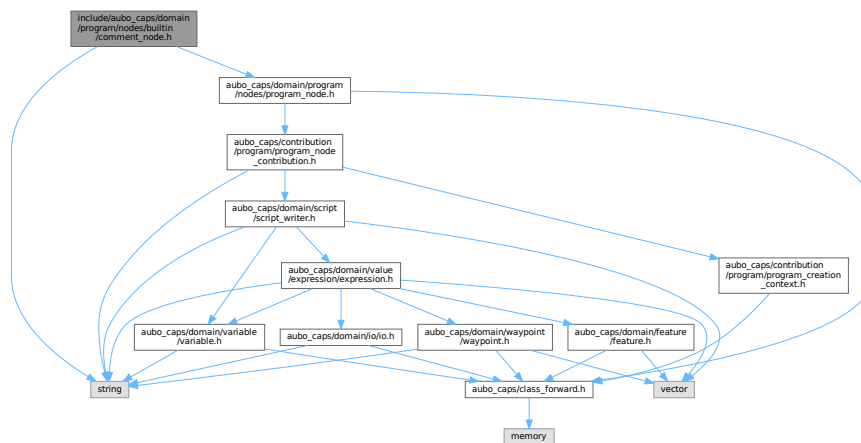
12.135 circle_move_node.h

[浏览该文件的文档.](#)

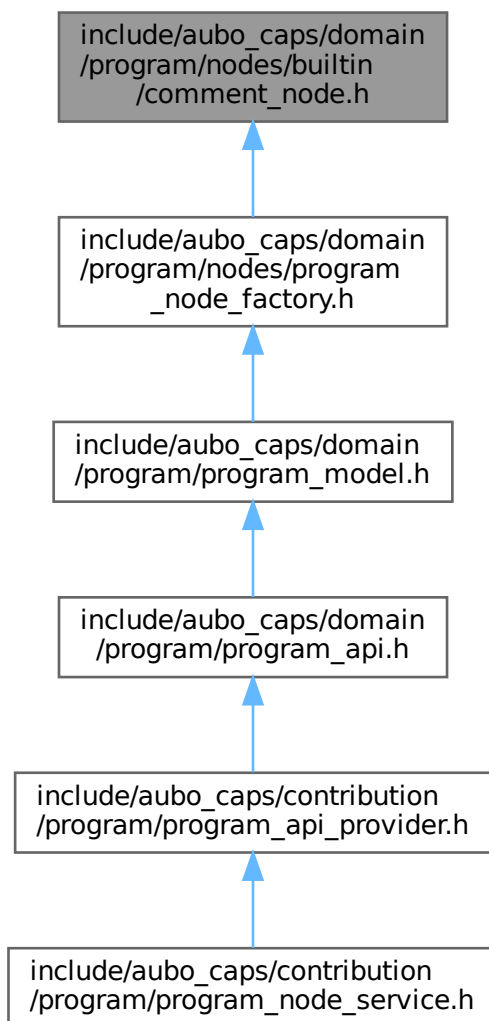

```
00001 #ifndef AUBO_SCOPE_CIRCLE_MOVE_NODE_H
00002 #define AUBO_SCOPE_CIRCLE_MOVE_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(CircleMoveNode);
00009
00010         class ARCS_ABI_EXPORT CircleMoveNode : public ProgramNode
00011         {
00012         public:
00013             CircleMoveNode(CircleMoveNode &f);
00014             CircleMoveNode(CircleMoveNode &&f);
00015             virtual ~CircleMoveNode();
00016
00017         private:
00018             friend class DataSwitch;
00019             CircleMoveNode();
00020             void *d_{ nullptr };
00021         };
00022
00023     } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_CIRCLE_MOVE_NODE_H
```

12.136 include/aubo_caps/domain/program/nodes/builtin/↵ comment_node.h 文件参考

#include <string>
#include <aubo_caps/domain/program/nodes/program_node.h>
comment_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::CommentNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(CommentNode\)](#)

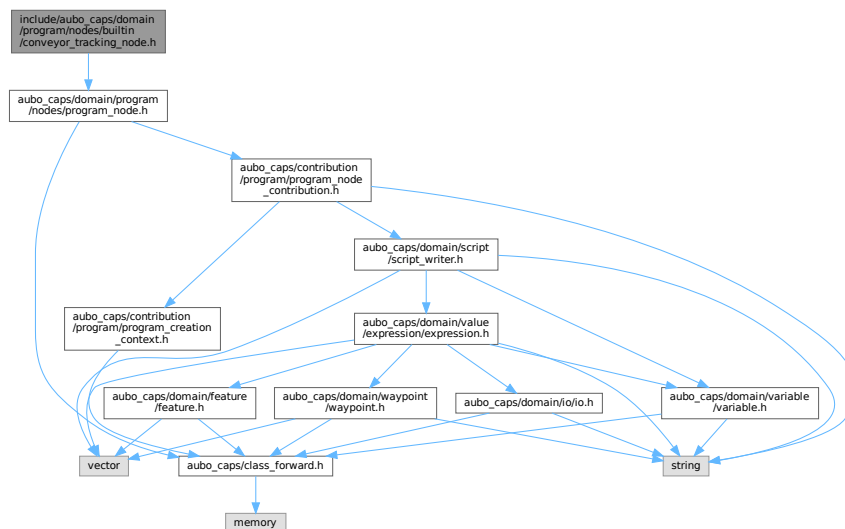
12.137 comment_node.h

[浏览该文件的文档.](#)

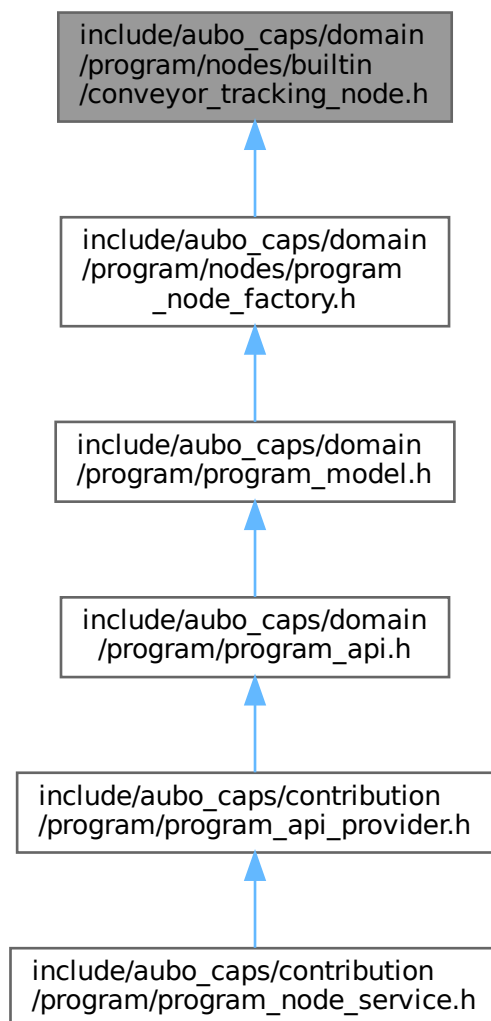
```
00001 #ifndef AUBO_SCOPE_COMMENT_NODE_H
00002 #define AUBO_SCOPE_COMMENT_NODE_H
00003
00004 #include <string>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(CommentNode);
00010
00011 class ARCS_ABI_EXPORT CommentNode : public ProgramNode
00012 {
00013 public:
00014     CommentNode(CommentNode &f);
00015     CommentNode(CommentNode &&f);
00016     virtual ~CommentNode();
00017
00018     /**
00019      *
00020      * @return The text of the Comment node.
00021      */
00022     std::string getComment() const;
00023
00024     /**
00025      * * Set the text for the Comment node.
00026      * @param comment The Comment for the node.
00027      * @return This node.
00028      */
00029     void setComment(const std::string &param_string);
00030
00031 private:
00032     friend class DataSwitch;
00033     CommentNode();
00034     void *d_{ nullptr };
00035 };
00036
00037 } // namespace aubo_scope
00038 } // namespace arcs
00039 #endif // AUBO_SCOPE_COMMENT_NODE_H
```

12.138 include/aubo_caps/domain/program/nodes/builtin/↵ conveyor_tracking_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
conveyor_tracking_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ConveyorTrackingNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ConveyorTrackingNode\)](#)

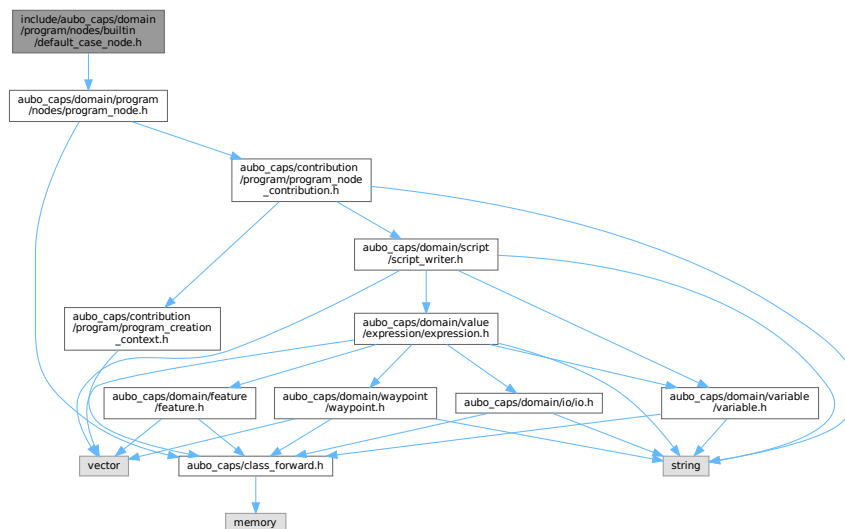
12.139 conveyor_tracking_node.h

[浏览该文件的文档.](#)

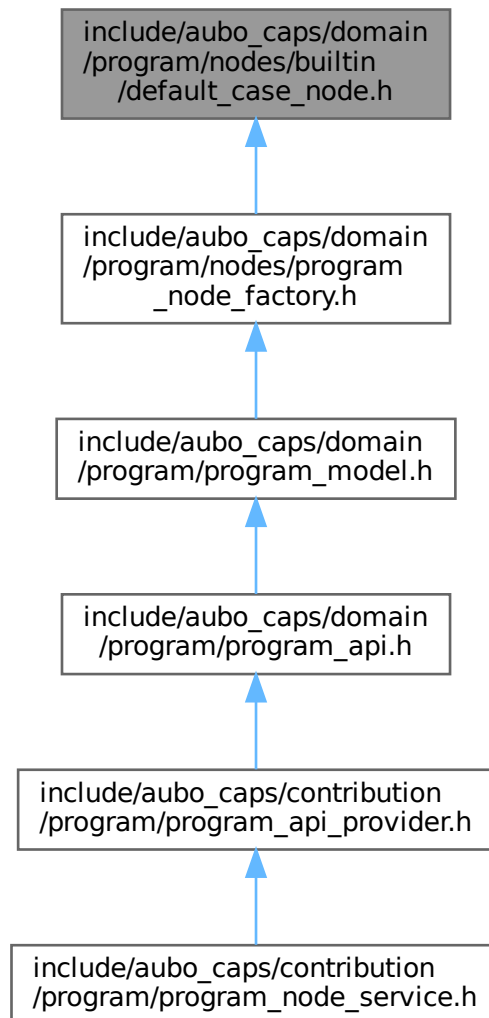
```
00001 #ifndef AUBO_SCOPE_CONVEYOR_TRACKING_NODE_H
00002 #define AUBO_SCOPE_CONVEYOR_TRACKING_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(ConveyorTrackingNode);
00009
00010         class ARCS_ABI_EXPORT ConveyorTrackingNode : public ProgramNode
00011         {
00012         public:
00013             enum ConveyorTrackingTargetType
00014             {
00015                 NEW_WORKPIECE,
00016                 PREVIOUS_WORKPIECE,
00017             };
00018
00019             enum ConveyorStartActionQueueType
00020             {
00021                 CLEAR_QUEUE,
00022                 RESERVE_QUEUE,
00023                 ASK
00024             };
00025
00026             enum ConveyorStartActionType
00027             {
00028                 CONTINUE_TRACKING,
00029                 ABANDON_TRACKING
00030             };
00031
00032             ConveyorTrackingNode(ConveyorTrackingNode &f);
00033             ConveyorTrackingNode(ConveyorTrackingNode &&f);
00034             virtual ~ConveyorTrackingNode();
00035
00036         private:
00037             friend class DataSwitch;
00038             ConveyorTrackingNode();
00039             void *d_{ nullptr };
00040         };
00041
00042     } // namespace aubo_scope
00043 } // namespace arcs
00044 #endif // AUBO_SCOPE_CONVEYOR_TRACKING_NODE_H
```

12.140 include/aubo_caps/domain/program/nodes/builtin/↔ default_case_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
default_case_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::DefaultCaseNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(DefaultCaseNode\)](#)

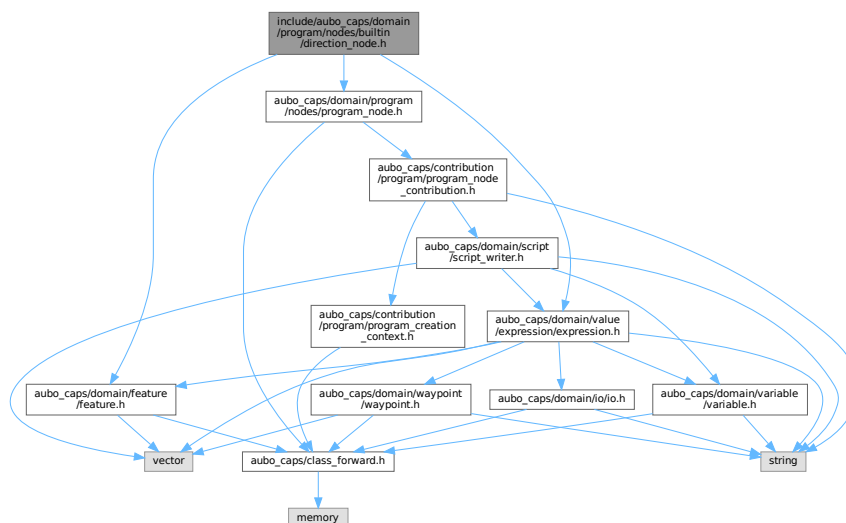
12.141 default_case_node.h

[浏览该文件的文档.](#)

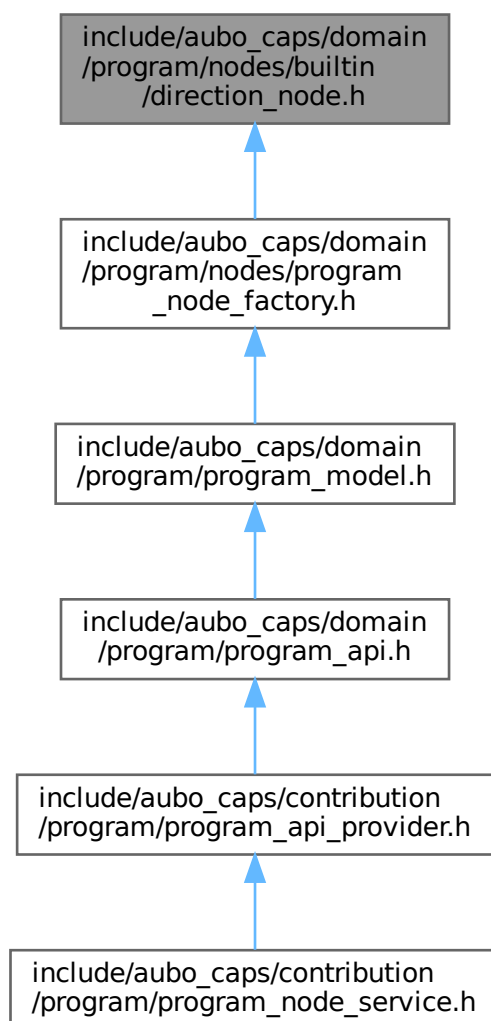
```
00001 #ifndef AUBO_SCOPE_DEFAULT_CASE_NODE_H
00002 #define AUBO_SCOPE_DEFAULT_CASE_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(DefaultCaseNode);
00009
00010         class ARCS_ABI_EXPORT DefaultCaseNode : public ProgramNode
00011         {
00012         public:
00013             DefaultCaseNode(DefaultCaseNode &f);
00014             DefaultCaseNode(DefaultCaseNode &&f);
00015             virtual ~DefaultCaseNode();
00016
00017         private:
00018             friend class DataSwitch;
00019             DefaultCaseNode();
00020             void *d_{ nullptr };
00021         };
00022
00023     } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_DEFAULT_CASE_NODE_H
```

12.142 include/aubo_caps/domain/program/nodes/builtin/↔ direction_node.h 文件参考

```
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/value/expression/expression.h>
#include <aubo_caps/domain/feature/feature.h>
direction_node.h 的引用 (Include) 关系图:
```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::DirectionNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(DirectionNode\)](#)

12.143 direction_node.h

[浏览该文件的文档.](#)


```

00001 #ifndef AUBO_SCOPE_DIRECTION_NODE_H
00002 #define AUBO_SCOPE_DIRECTION_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006 #include <aubo_caps/domain/feature/feature.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(DirectionNode);
00011
00012 class ARCS_ABI_EXPORT DirectionNode : public ProgramNode
00013 {
00014 public:
00015     enum ParameterType : int
00016     {
00017         /**
00018          * The tool motion parameters are inherited from the MoveL or
00019          * MoveP
00020          * parent.
00021          */
00022         SHARED_MOTION,
00023
00024         /**
00025          * The tool motion parameters are selected. <br>
00026          *
00027          * The Selection instance can be cast to {@link
00028          * ToolMotionParameters}.
00029          */
00030         TOOL_MOTION
00031     };
00032
00033     enum Axis : int
00034     {
00035         /**
00036          * Move in the positive X direction relative to the feature.
00037          */
00038         X_PLUS,
00039
00040         /**
00041          * Move in the negative X direction relative to the feature.
00042          */
00043         X_MINUS,
00044
00045         /**
00046          * Move in the positive Y direction relative to the feature.
00047          */
00048         Y_PLUS,
00049
00050         /**
00051          * Move in the negative Y direction relative to the feature.
00052          */
00053         Y_MINUS,
00054
00055         /**
00056          * Move in the positive Z direction relative to the feature.
00057          */
00058         Z_PLUS,
00059
00060         /**
00061          * Move in the negative Z direction relative to the feature.
00062          */
00063         Z_MINUS
00064     };
00065
00066     enum DirectionSelectionType : int
00067     {
00068         /**
00069          * A direction vector is specified using an Expression.
00070          * The Selection instance can be cast to {@link
00071          * DirectionVectorExpression}
00072          */
00073         EXPRESSION,
00074
00075         /**
00076          * Direction is specified using an Axis.
00077          * The Selection instance can be cast to {@link DirectionAxis}
00078          */
00079         AXIS
00080     };
00081
00082     enum FeatureSelectionType : int
00083     {
00084         /**
00085          * The selected feature is no longer available (e.g. because the user
00086          * has removed the feature in the Installation or the needed
00087          * Installation is not currently loaded)

```

```

00088      */
00089      UNRESOLVED_FEATURE,
00090
00091      /**
00092       * Feature is inherited from the MoveL or MoveP parent.
00093       */
00094      SHARED,
00095
00096      /**
00097       * Feature is selected. <br>
00098       * The Selection instance can be cast to {@link DirectionNodeFeature}
00099       */
00100      FEATURE
00101 };
00102
00103 DirectionNode(DirectionNode &f);
00104 DirectionNode(DirectionNode &&f);
00105 virtual ~DirectionNode();
00106
00107 ParameterType getType();
00108 void setType(ParameterType type);
00109
00110 /**
00111  * @return the Axis used to define the direction.
00112  */
00113 Axis getAxis();
00114 void setAxis(Axis axis);
00115
00116 /**
00117  *
00118  * @return the feature selected for this Direction node.
00119  */
00120 FeaturePtr getFeature();
00121 void setFeature(FeaturePtr feature);
00122
00123 DirectionSelectionType getDirectionSelectionType();
00124 void setDirectionSelectionType(DirectionSelectionType selection);
00125 /**
00126  *
00127  * @return the expression used to define the direction vector.
00128  */
00129 ExpressionPtr getExpression();
00130 void setExpression(ExpressionPtr expression);
00131
00132 FeatureSelectionType getFeatureSelectionType();
00133 void setFeatureSelectionType(FeatureSelectionType selection);
00134 /**
00135  * @return the tool speed to be achieved.
00136  */
00137 double getToolSpeed();
00138 void setToolSpeed(double speed);
00139
00140 /**
00141  * @return the tool acceleration to use for the movement
00142  */
00143 double getToolAcceleration();
00144 void setToolAcceleration(double acc);
00145
00146 private:
00147     friend class DataSwitch;
00148     DirectionNode();
00149     void *d_{ nullptr };
00150 };
00151
00152 } // namespace aubo_scope
00153 } // namespace arcs
00154 #endif // AUBO_SCOPE_DIRECTION_NODE_H

```

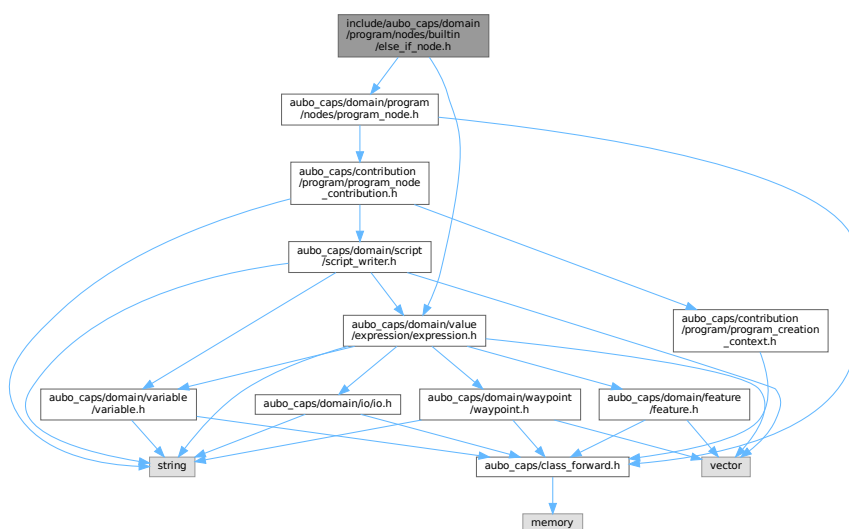
12.144 include/auo_caps/domain/program/nodes/builtin/else_if_node.h 文件参考

```

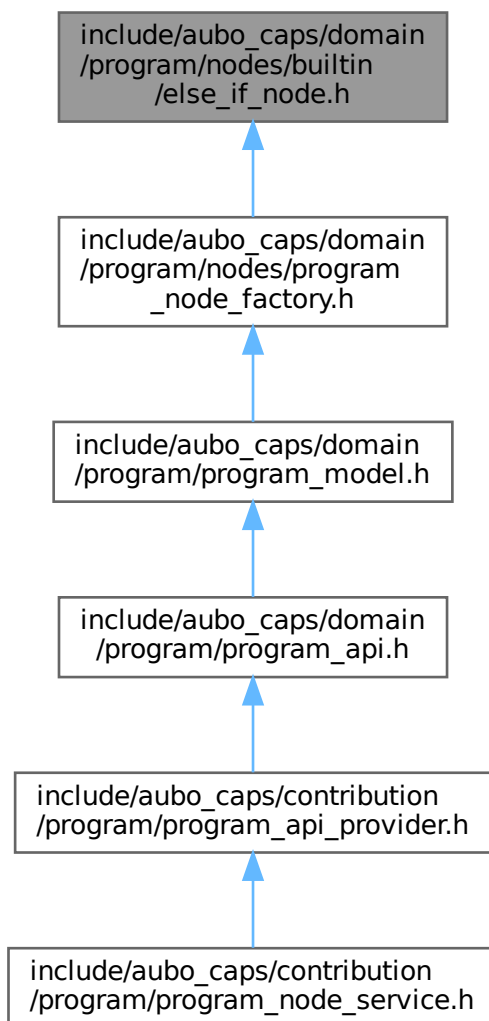
#include <auo_caps/domain/program/nodes/program_node.h>
#include <auo_caps/domain/value/expression/expression.h>

```

else_if_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ElseIfNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ElseIfNode\)](#)

12.145 else_if_node.h

[浏览该文件的文档.](#)

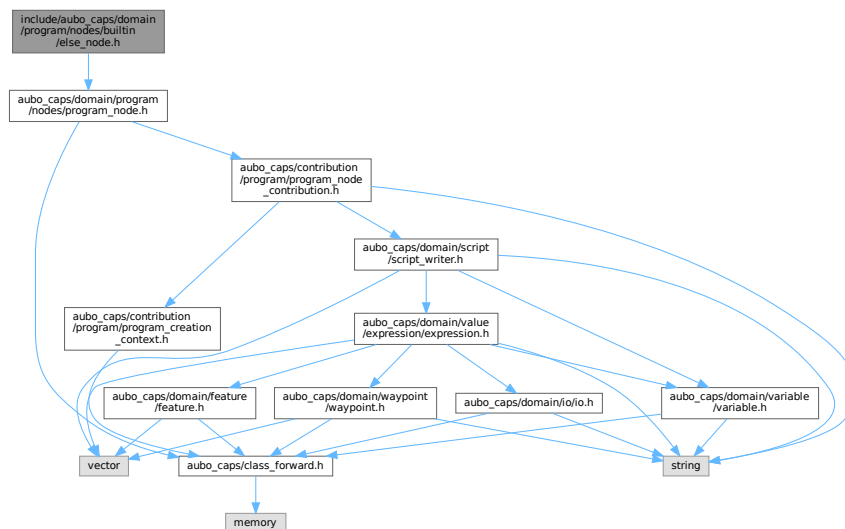
```

00001 #ifndef AUBO_SCOPE_ELSE_IF_NODE_H
00002 #define AUBO_SCOPE_ELSE_IF_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(ElseIfNode);
00010
00011 class ARCS_ABI_EXPORT ElseIfNode : public ProgramNode
00012 {
00013 public:
00014     ElseIfNode(ElseIfNode &f);
00015     ElseIfNode(ElseIfNode &&f);
00016     virtual ~ElseIfNode();
00017
00018     /**
00019      * @param expression The expression that will be evaluated in this
00020      * ElseIfNode.
00021      * @return The same ElseIfNode.
00022      */
00023     void setExpression(ExpressionPtr expression);
00024
00025     /**
00026      * @return the Expression to be evaluated in this ElseIf node.
00027      */
00028     ExpressionPtr getExpression();
00029 private:
00030     friend class DataSwitch;
00031     ElseIfNode();
00032     void *d_{ nullptr };
00033 };
00034 } // namespace aubo_scope
00035 } // namespace arcs
00036 #endif // AUBO_SCOPE_ELSE_IF_NODE_H

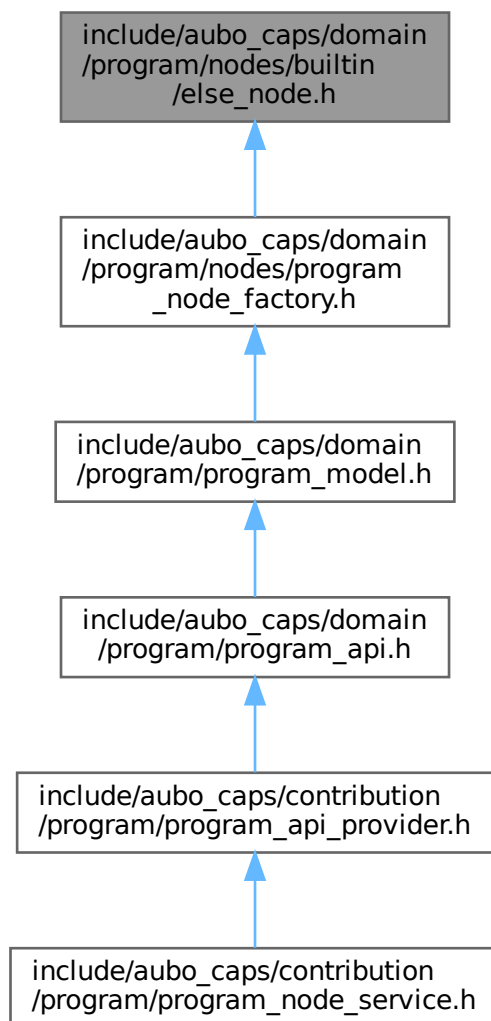
```

12.146 include/aubo_caps/domain/program/nodes/builtin/else_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
else_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ElseNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ElseNode\)](#)

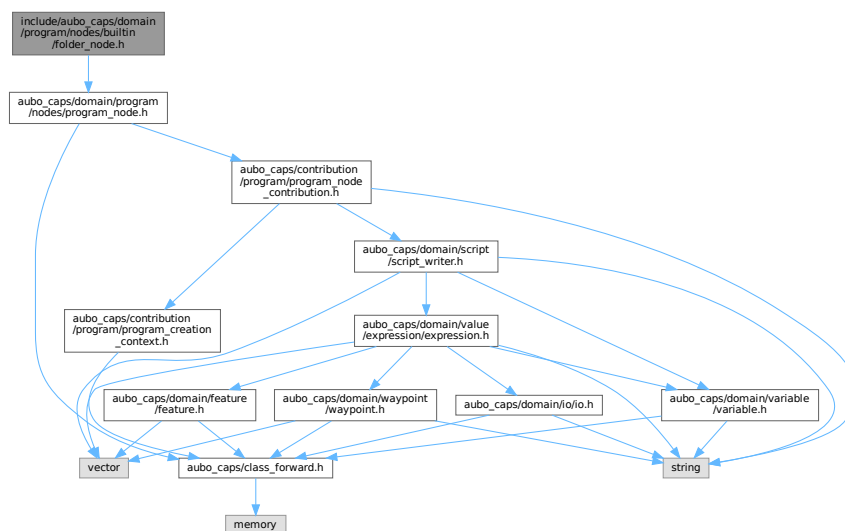
12.147 else_node.h

[浏览该文件的文档.](#)

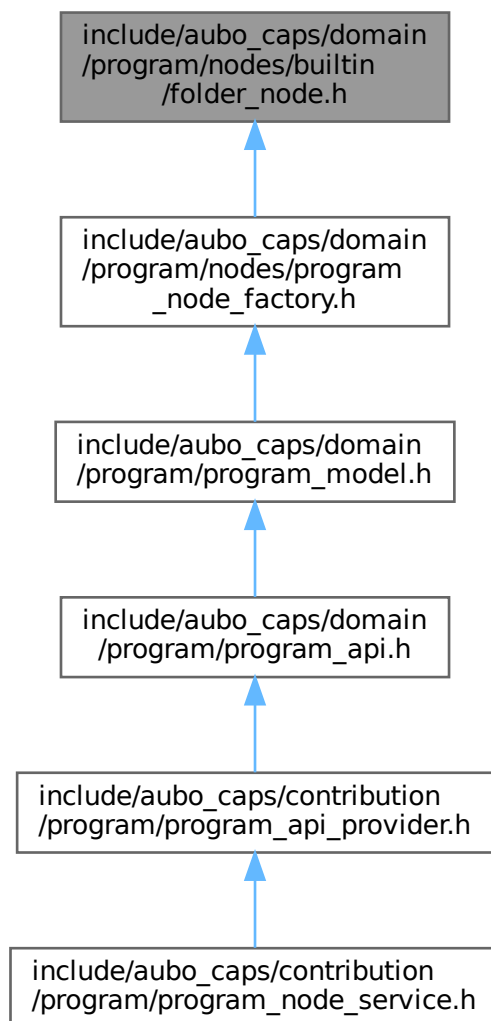
```
00001 #ifndef AUBO_SCOPE_ELSE_NODE_H
00002 #define AUBO_SCOPE_ELSE_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(ElseNode);
00009
00010         class ARCS_ABI_EXPORT ElseNode : public ProgramNode
00011         {
00012         public:
00013             ElseNode(ElseNode &f);
00014             ElseNode(ElseNode &&f);
00015             virtual ~ElseNode();
00016
00017         private:
00018             friend class DataSwitch;
00019             ElseNode();
00020             void *d_{ nullptr };
00021         };
00022
00023     } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_ELSE_NODE_H
```

12.148 include/aubo_caps/domain/program/nodes/builtin/↔ folder_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
folder_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::FolderNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(FolderNode\)](#)

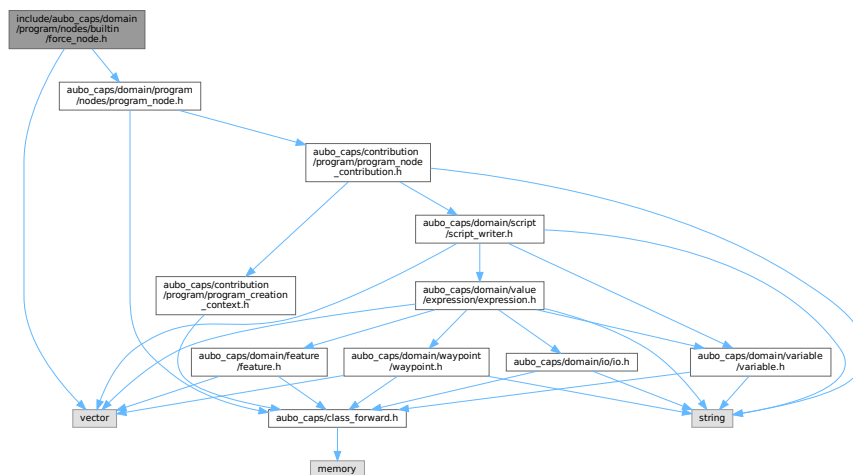
12.149 folder_node.h

[浏览该文件的文档.](#)

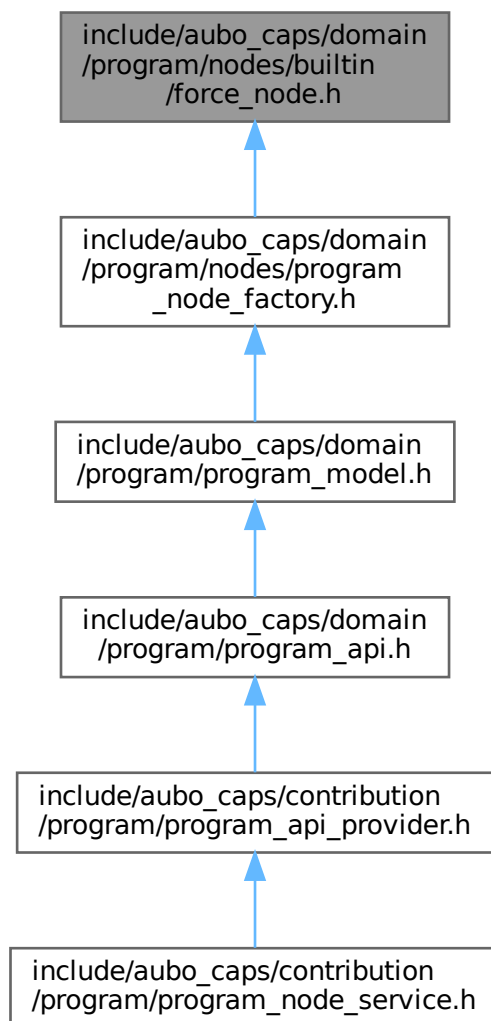

```
00001 #ifndef AUBO_SCOPE_FOLDER_NODE_H
00002 #define AUBO_SCOPE_FOLDER_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(FolderNode);
00009
00010         class ARCS_ABI_EXPORT FolderNode : public ProgramNode
00011         {
00012         public:
00013             FolderNode(FolderNode &f);
00014             FolderNode(FolderNode &&f);
00015             virtual ~FolderNode();
00016
00017             /**
00018              * * Set the name for the Folder node.
00019              * @param set The name for the node.
00020              * @return This node.
00021              */
00022             void setFolderName(const std::string &folder_name);
00023
00024         private:
00025             friend class DataSwitch;
00026             FolderNode();
00027             void *d_{ nullptr };
00028         };
00029
00030     } // namespace aubo_scope
00031 } // namespace arcs
00032 #endif // AUBO_SCOPE_FOLDER_NODE_H
```

12.150 include/aubo_caps/domain/program/nodes/builtin/↵ force_node.h 文件参考

```
#include <vector>
#include <aubo_caps/domain/program/nodes/program_node.h>
force_node.h 的引用 (Include) 关系图:
```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ForceNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ForceNode\)](#)

12.151 force_node.h

[浏览该文件的文档.](#)

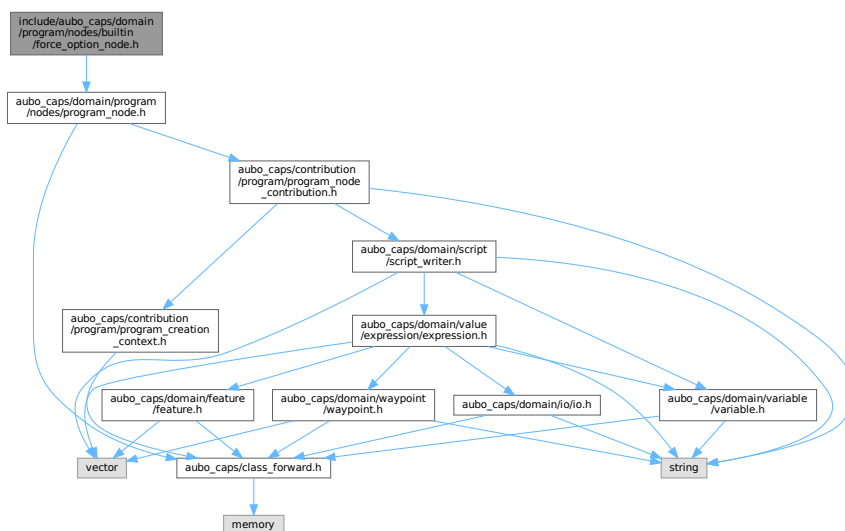
```

00001 #ifndef AUBO_SCOPE_FORCE_NODE_H
00002 #define AUBO_SCOPE_FORCE_NODE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(ForceNode);
00010
00011 class ARCS_ABI_EXPORT ForceNode : public ProgramNode
00012 {
00013 public:
00014     /**
00015      * The configuration type used to determine which type of configuration this
00016      * instance is.
00017      */
00018     enum ConfigType
00019     {
00020
00021         /**
00022          * <p>
00023          * Simple type is selected. This type has no further information.
00024          * </p>
00025          *
00026          * Setting this type of config will be ignored and will have no effect
00027          * on the node.
00028          */
00029         SIMPLE,
00030
00031         /**
00032          * <p>
00033          * Frame type is selected.
00034          * </p>
00035          *
00036          * The config instance can be cast to {@link FrameForceNodeConfig}.
00037          */
00038         FRAME,
00039
00040         /**
00041          * <p>
00042          * Point type is selected. This type has no further information.
00043          * </p>
00044          *
00045          * Setting this type of config will be ignored and will have no effect
00046          * on the node.
00047          */
00048         POINT,
00049
00050         /**
00051          * <p>
00052          * Motion type is selected. This type has no further information.
00053          * </p>
00054          *
00055          * Setting this type of config will be ignored and will have no effect
00056          * on the node.
00057          */
00058         MOTION
00059     };
00060
00061     ForceNode(ForceNode &f);
00062     ForceNode(ForceNode &&f);
00063     virtual ~ForceNode();
00064
00065 private:
00066     friend class DataSwitch;
00067     ForceNode();
00068     void *d_{ nullptr };
00069 };
00070
00071 } // namespace aubo_scope
00072 } // namespace arcs
00073 #endif // AUBO_SCOPE_FORCE_NODE_H

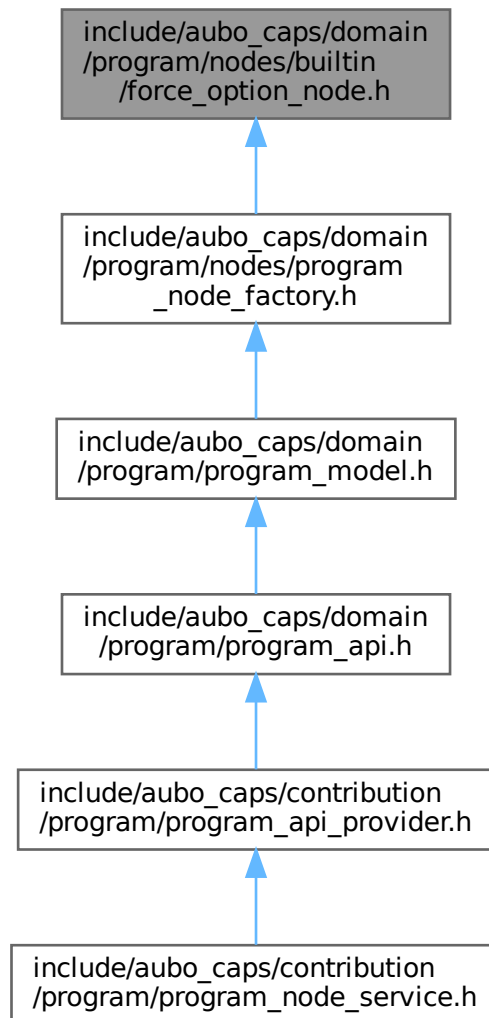
```

12.152 include/aubo_caps/domain/program/nodes/builtin/force_option_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
force_option_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ForceOptionNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ForceOptionNode\)](#)

12.153 force_option_node.h

[浏览该文件的文档.](#)

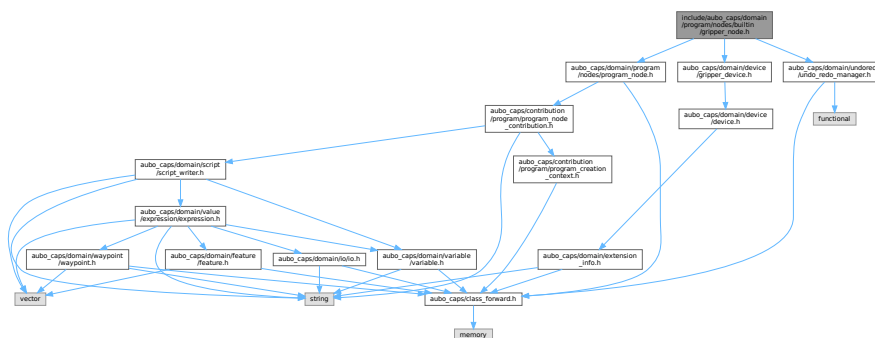
```

00001 #ifndef AUBO_SCOPE_FORCE_OPTION_NODE_H
00002 #define AUBO_SCOPE_FORCE_OPTION_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(ForceOptionNode);
00009
00010         class ARCS_ABI_EXPORT ForceOptionNode : public ProgramNode
00011         {
00012         public:
00013             ForceOptionNode(ForceOptionNode &f);
00014             ForceOptionNode(ForceOptionNode &&f);
00015             virtual ~ForceOptionNode();
00016
00017         private:
00018             friend class DataSwitch;
00019             ForceOptionNode();
00020             void *d_{ nullptr };
00021         };
00022
00023     } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_FORCE_OPTION__NODE_H

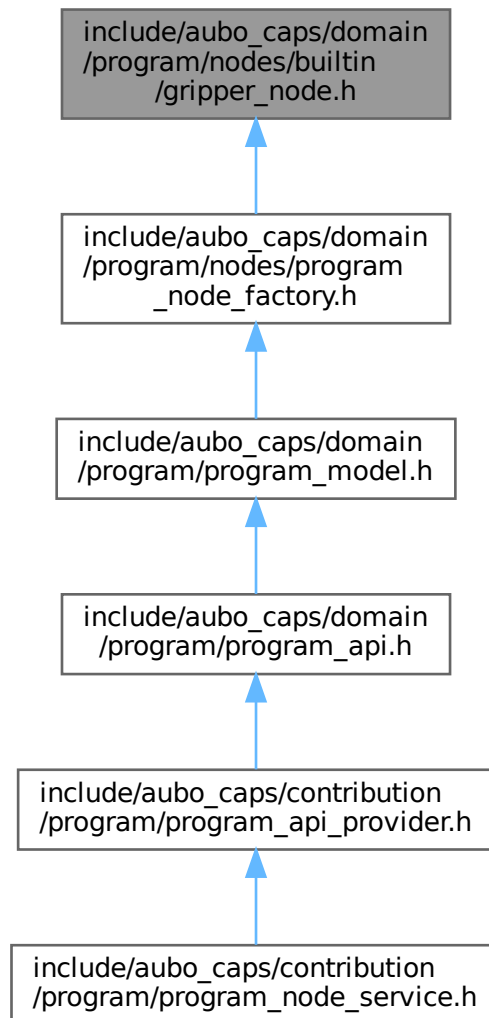
```

12.154 include/aubo_caps/domain/program/nodes/builtin/← gripper_node.h 文件参考

#include <aubo_caps/domain/device/gripper_device.h>
 #include <aubo_caps/domain/program/nodes/program_node.h>
 #include <aubo_caps/domain/undoredo/undo_redo_manager.h>
 gripper_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::GripperNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(GripperNode\)](#)

12.155 gripper_node.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_GRIPPER_NODE_H
00002 #define AUBO_SCOPE_GRIPPER_NODE_H
00003
00004 #include <aubo_caps/domain/device/gripper_device.h>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006 #include <aubo_caps/domain/undoredo/undo_redo_manager.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(GripperNode);
00011
00012 /**
00013  * \chinese
00014  * <p> 此接口表示一个程序节点，可用于编程所选夹爪设备的夹取和释放动作。</p>
00015  * <p> 此接口的实例可使用 GripperNodeFactory 接口构建。</p>
00016  * \endchinese
00017  * \english
00018  * <p>This interface represents a program node which can be used for programming
00019  * grip and release actions with a selected gripper device.</p>
00020  * <p>An instance of this interface can be built using the {@link
00021  * GripperNodeFactory} interface and is returned as a result when {@link
00022  * GripperNodeFactory#createGripperNode(GripperDevice)} is called.</p>
00023  * \endenglish
00024  */
00025 class ARCS_ABI_EXPORT GripperNode : public ProgramNode
00026 {
00027 public:
00028 /**
00029  * The configuration type used to determine which type of configuration this
00030  * instance is.
00031  */
00032 enum ConfigType
00033 {
00034
00035 /**
00036  * <p>
00037  * A Gripper action (i.e. grip or release) has not been selected
00038  * </p>
00039  *
00040  * The config instance can be cast to {@link UndefinedActionConfig}.
00041  * This is only relevant if the Gripper node uses a multi-gripper
00042  * device, i.e. a gripper device supporting the multi-gripper capability
00043  * (see the
00044  * {@link MultiGripperSupport} interface). The instance will provide
00045  * access to the individual gripper selection.
00046  */
00047 UNDEFINED,
00048
00049 /**
00050  * <p>
00051  * The gripper is configured for a grip action
00052  * </p>
00053  *
00054  * The config instance can be cast to {@link GripActionConfig}.
00055  */
00056 GRIP_ACTION,
00057
00058 /**
00059  * <p>
00060  * The gripper is configured for a release action
00061  * </p>
00062  *
00063  * The config instance can be cast to {@link ReleaseActionConfig}.
00064  */
00065 RELEASE_ACTION
00066 };
00067
00068 GripperNode(GripperNode &f);
00069 GripperNode(GripperNode &&f);
00070 virtual ~GripperNode();
00071
00072 /**
00073  * \chinese
00074  * 返回配置类型。适当转换此实例以访问特定的 getter。
00075  * @return 此配置的类型
00076  * \endchinese
00077  * \english
00078  * This method returns the type of configuration. Cast this instance
00079  * appropriately to have access to specific getters.
00080  * @return The type of this config.
00081  * \endenglish
00082  */
00083 ConfigType getConfigType();
00084
00085 void setConfigType(ConfigType type);
00086
00087 /**

```



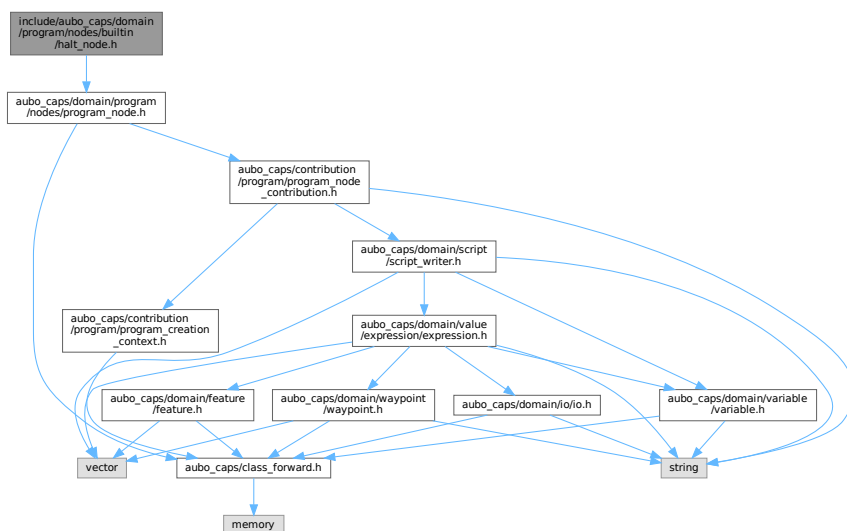
```

00088     * \chinese
00089     * @return 此夹爪节点使用的夹爪设备
00090     * \endchinese
00091     * \english
00092     * @return The gripper device used by this Gripper node
00093     * \endenglish
00094     */
00095     GripperDevicePtr getGripperDevice();
00096
00097     /**
00098     * The setting type used to determine which type of setting this instance
00099     * is.
00100     */
00101     enum SettingType
00102     {
00103
00104         /**
00105          * <p>
00106          * No selection has been made.
00107          * </p>
00108          */
00109         UNDEFINED0,
00110
00111         /**
00112          * <p>
00113          * A payload (from the installation) is selected.
00114          * </p>
00115          *
00116          * The setting instance can be cast to {@link PayloadSelection}.
00117          */
00118         PAYLOAD_SELECTION,
00119
00120         /**
00121          * <p>
00122          * Custom payload parameters are specified.
00123          * </p>
00124          *
00125          * The setting instance can be cast to {@link CustomPayloadParameters}.
00126          */
00127         CUSTOM_PARAMETERS
00128     };
00129
00130     /**
00131     * \chinese
00132     * 返回设置类型。适当转换此实例以访问特定的 getter。
00133     * @return 此负载设置的类型
00134     * \endchinese
00135     * \english
00136     * This method returns the type of setting. Cast this instance appropriately
00137     * to have access to specific getters.
00138     * @return The type of this payload setting.
00139     * \endenglish
00140     */
00141     SettingType getSettingType();
00142
00143 private:
00144     friend class DataSwitch;
00145     GripperNode();
00146     void *d_{ nullptr };
00147 };
00148 } // namespace aubo_scope
00149 } // namespace arcs
00150
00151 #endif

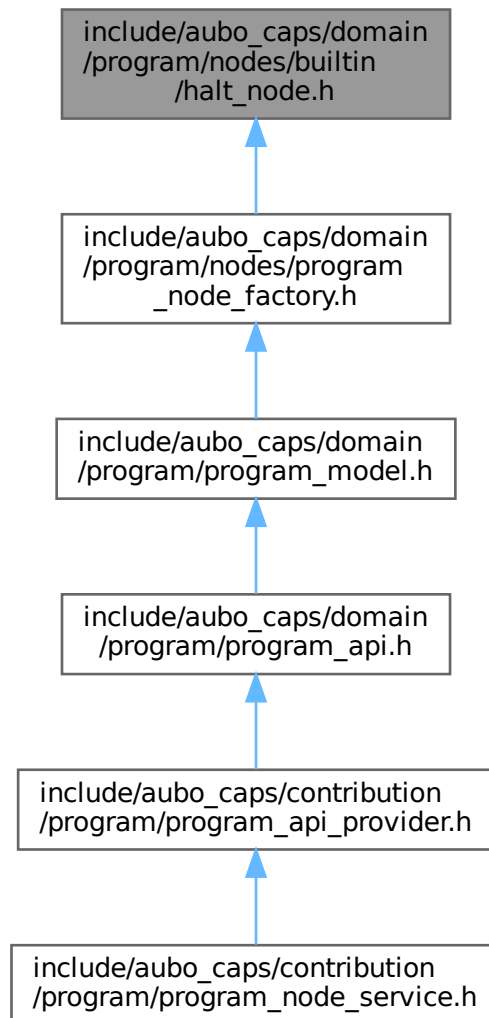
```

12.156 include/aubo_caps/domain/program/nodes/builtin/halt_node.h 文件参考

`#include <aubo_caps/domain/program/nodes/program_node.h>`
 halt_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo__scope::HaltNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo__scope](#)

函数

- [arcs::aubo__scope::ARCS_CLASS_FORWARD \(HaltNode\)](#)

12.157 halt__node.h

[浏览该文件的文档.](#)

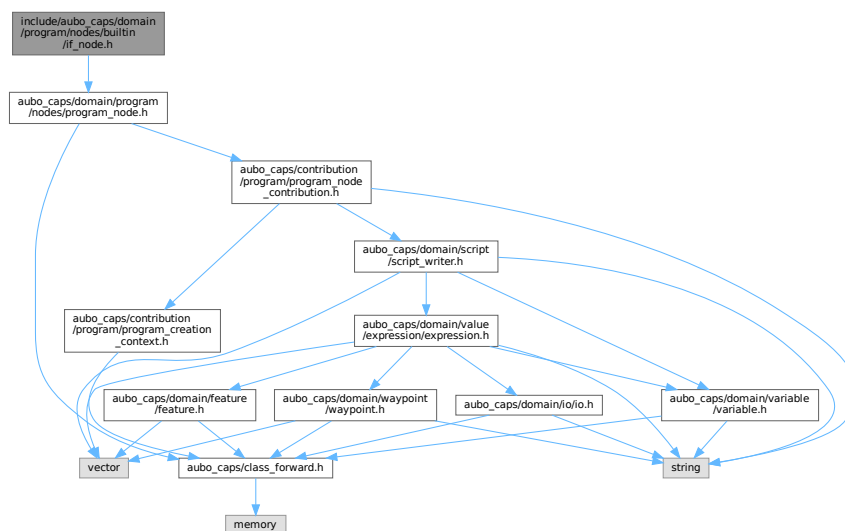
```

00001 #ifndef AUBO_SCOPE_HALT_NODE_H
00002 #define AUBO_SCOPE_HALT_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(HaltNode);
00009
00010 class ARCS_ABI_EXPORT HaltNode : public ProgramNode
00011 {
00012 public:
00013     HaltNode(HaltNode &f);
00014     HaltNode(HaltNode &&f);
00015     virtual ~HaltNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     HaltNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_HALT_NODE_H

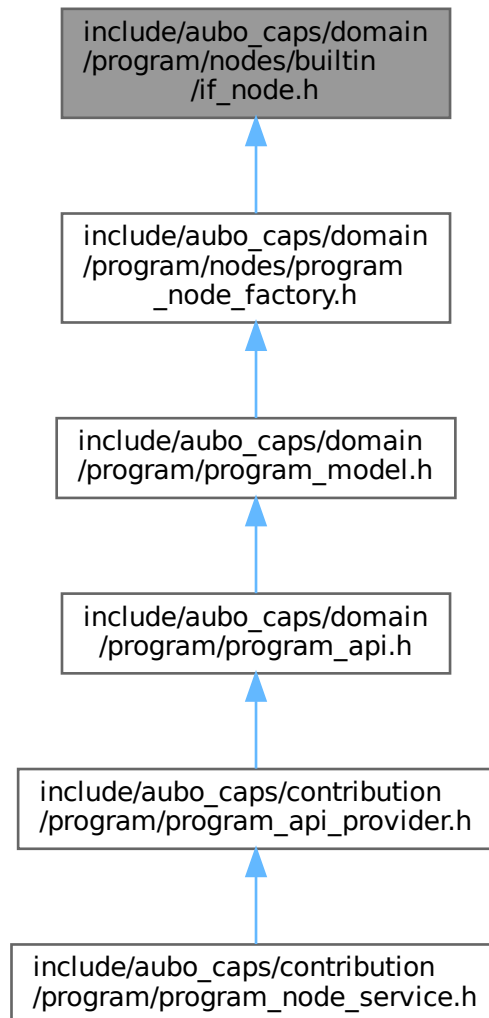
```

12.158 include/aubo_caps/domain/program/nodes/builtin/if_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
if_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::IfNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (Expression)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (IfNode)

12.159 if_node.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_IF_NODE_H
00002 #define AUBO_SCOPE_IF_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(Expression);
00009 ARCS_CLASS_FORWARD(IfNode);
00010
00011 class ARCS_ABI_EXPORT IfNode : public ProgramNode
00012 {
00013 public:
00014     IfNode(IfNode &f);
00015     IfNode(IfNode &&f);
00016     virtual ~IfNode();
00017
00018     /**
00019      * \chinese
00020      * 设置节点使用的表达式。
00021      * @param expression 将在 If 节点中求值的表达式
00022      * @throws IllegalStateException 如果在 UndoableChanges 范围外调用
00023      * \endchinese
00024      * \english
00025      * Sets the expression used by the node.
00026      * @param expression The expression that will be evaluated in this If Node.
00027      * @throws IllegalStateException if called from a Qt-based AuboCap
00028      * outside of an {@link UndoableChanges} scope.
00029      * \endenglish
00030      */
00031     void setExpression(ExpressionPtr expression);
00032
00033     /**
00034      * \chinese
00035      * @return 在此 If 节点中求值的表达式
00036      * \endchinese
00037      * \english
00038      * @return the Expression to be evaluated in this If node.
00039      * \endenglish
00040      */
00041     ExpressionPtr getExpression();
00042
00043 private:
00044     friend class DataSwitch;
00045     IfNode();
00046     void *d_{ nullptr };
00047 };
00048
00049 } // namespace aubo_scope
00050 } // namespace arcs
00051 #endif // AUBO_SCOPE_IF_NODE_H

```

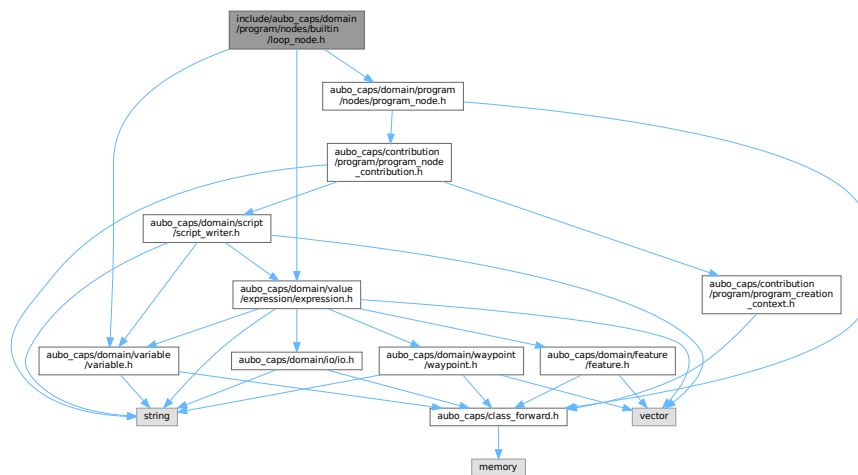
12.160 include/aubo_caps/domain/program/nodes/builtin/loop_node.h 文件参考

```

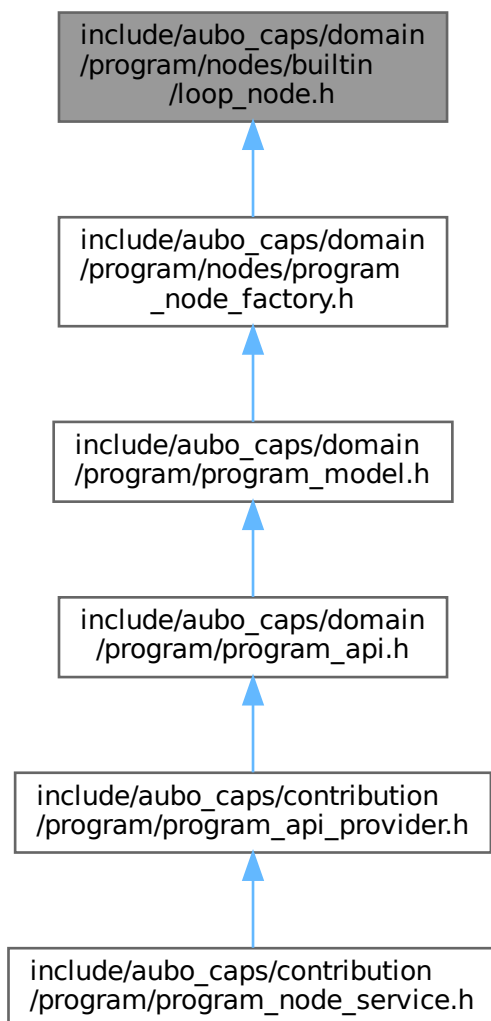
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/value/expression/expression.h>
#include <aubo_caps/domain/variable/variable.h>

```

loop_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::LoopNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(LoopNode\)](#)

12.161 loop_node.h

[浏览该文件的文档.](#)


```

00001 #ifndef AUBO_SCOPE_LOOP_NODE_H
00002 #define AUBO_SCOPE_LOOP_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006 #include <aubo_caps/domain/variable/variable.h>
00007
00008 namespace arcs {
00009     namespace aubo_scope {
00010         ARCS_CLASS_FORWARD(LoopNode);
00011
00012         class ARCS_ABI_EXPORT LoopNode : public ProgramNode
00013         {
00014         public:
00015             /**
00016              * The configuration type used to determine which type of configuration this
00017              * instance is.
00018              */
00019             enum ConfigType
00020             {
00021
00022                 /**
00023                  * <p>
00024                  * Looping infinitely has been selected.
00025                  * </p>
00026                  *
00027                  * The config instance can be cast to {@link AlwaysLoopNodeConfig}
00028                  */
00029                 ALWAYS,
00030
00031                 /**
00032                  * <p>
00033                  * Looping a certain number of times has been selected.
00034                  * </p>
00035                  *
00036                  * The config instance can be cast to {@link CounterLoopNodeConfig}
00037                  */
00038                 COUNTER,
00039
00040                 /**
00041                  * <p>
00042                  * Looping as long as the evaluation of an expression is true has been
00043                  * selected.
00044                  * </p>
00045                  *
00046                  * The config instance can be cast to {@link ExpressionLoopNodeConfig}
00047                  */
00048                 EXPRESSION
00049             };
00050
00051             LoopNode(LoopNode &f);
00052             LoopNode(LoopNode &&f);
00053             virtual ~LoopNode();
00054
00055             /**
00056              * This method returns the type of configuration. Cast this instance
00057              * appropriately to have access to specific getters.
00058              *
00059              * @return the type of this config.
00060              */
00061             ConfigType getConfigType();
00062             void setConfigType(ConfigType type);
00063
00064             /**
00065              * Creates a loop counter configuration.
00066              *
00067              * @param loopCount the number of iterations through the loop node.
00068              * @param errorHandler error handler for handling validation. If using
00069              *   {@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00070              *   nearest valid loop count value.
00071              * @return the configuration.
00072              */
00073             void setLoopCount(int loop_count);
00074             int getLoopCount();
00075
00076             /**
00077              * Create a loop expression configuration.
00078              *
00079              * @param expression the expression whose evaluation must be true to
00080              *   continue looping, not <code>null</code>.
00081              * @param evaluateContinuously if <code>true</code> the expression will be
00082              *   evaluated continuously during the looping (i.e. the loop can be
00083              *   interrupted anytime during its execution), otherwise <code>false</code>
00084              *   and the expression will be evaluated once per loop iteration.
00085              * @return the configuration.
00086              */
00087             void setExpression(ExpressionPtr expression);

```

```

00088 ExpressionPtr getExpression();
00089
00090 /**
00091  *
00092  * @return <code>true</code> if the expression will be evaluated
00093  * continuously during the looping (i.e. the loop can be interrupted anytime
00094  * during its execution), otherwise <code>false</code> and the expression
00095  * will be evaluated once per loop iteration.
00096  */
00097 bool isEvaluateContinously();
00098 void setEvaluateContinously(bool cont);
00099
00100 /**
00101  *
00102  * @return the variable for storing the loop count
00103  */
00104 void setLoopCountVariable(VariablePtr var);
00105 VariablePtr getLoopCountVariable();
00106
00107 private:
00108 friend class DataSwitch;
00109 LoopNode();
00110 void *d_{ nullptr };
00111 };
00112
00113 } // namespace aubo_scope
00114 } // namespace arcs
00115 #endif // AUBO_SCOPE_LOOP_NODE_H

```

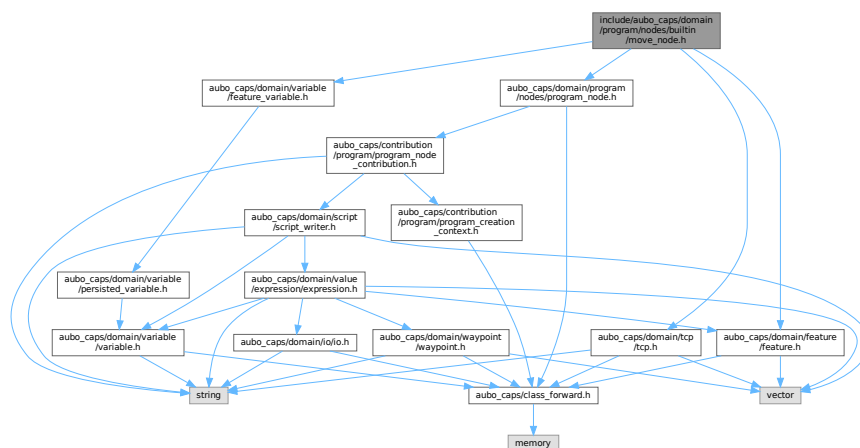
12.162 include/auco_caps/domain/program/nodes/builtin/↵ move_node.h 文件参考

```

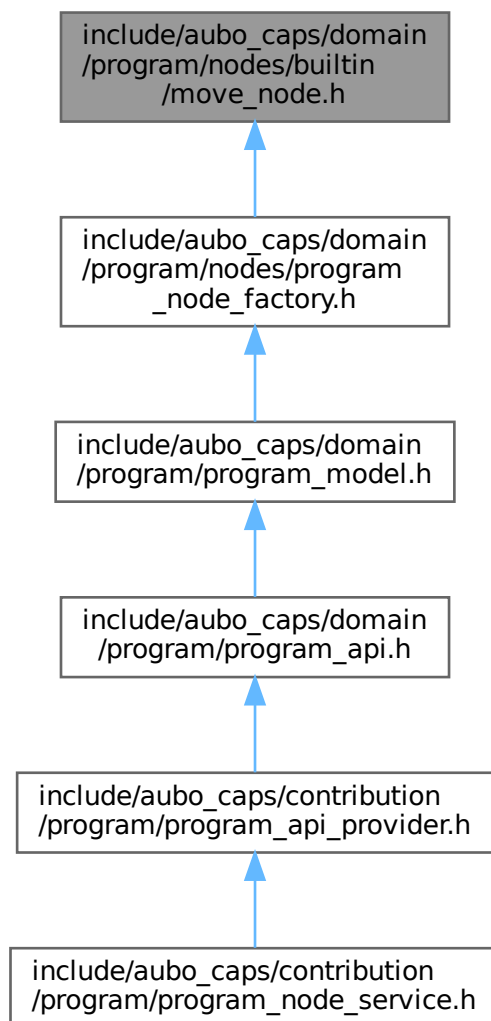
#include <auco_caps/domain/program/nodes/program_node.h>
#include <auco_caps/domain/feature/feature.h>
#include <auco_caps/domain/tcp/tcp.h>
#include <auco_caps/domain/variable/feature_variable.h>

```

move_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::MoveNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(MoveNode\)](#)

12.163 move_node.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_MOVE_NODE_H
00002 #define AUBO_SCOPE_MOVE_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/feature/feature.h>
00006 #include <aubo_caps/domain/tcp/tcp.h>
00007 #include <aubo_caps/domain/variable/feature_variable.h>
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011 ARCS_CLASS_FORWARD(MoveNode);
00012
00013 class ARCS_ABI_EXPORT MoveNode : public ProgramNode
00014 {
00015 public:
00016     enum ConfigType : int
00017     {
00018
00019         /**
00020          * <p>
00021          * Movement type is MoveJ.
00022          * </p>
00023          *
00024          * The config instance can be cast to {@link MoveJMoveNodeConfig}.
00025          */
00026         MOVE_J = 0,
00027
00028         /**
00029          * <p>
00030          * Movement type is MoveL.
00031          * </p>
00032          *
00033          * The config instance can be cast to {@link MoveLMoveNodeConfig}.
00034          */
00035         MOVE_L,
00036         MOVE_S,
00037     };
00038
00039     enum TcpSelectionType : int
00040     {
00041
00042         /**
00043          * Motions are adjusted to the currently active TCP. The TCP is
00044          * determined during runtime of the program.
00045          */
00046         USE_ACTIVE_TCP = 0,
00047
00048         /**
00049          * No TCP is used and the motion under this Move command will be with
00050          * respect to the tool output flange.
00051          */
00052         IGNORE_ACTIVE_TCP,
00053
00054         /**
00055          * A specific TCP is selected. <br>
00056          * The TCPSelection instance can be cast to {@link MoveNodeTCP}.
00057          */
00058         MOVE_TCP
00059     };
00060
00061     enum FeatureSelectionType : int
00062     {
00063
00064         /**
00065          * The selected feature is no longer available (e.g. because the user
00066          * has removed the feature in the Installation or the needed
00067          * Installation is not currently loaded)
00068          */
00069         UNRESOLVED_FEATURE,
00070
00071         /**
00072          * Feature is selected.
00073          * The Selection instance can be cast to {@link MoveNodeFeature}
00074          */
00075         FEATURE
00076     };
00077
00078     MoveNode(MoveNode &f);
00079     MoveNode(MoveNode &&f);
00080     virtual ~MoveNode();
00081
00082     /**
00083      * \chinese
00084      * 返回配置类型。适当转换此实例以访问特定的 getter。
00085      * @return 此配置的类型
00086      * \endchinese
00087      * \english

```

```

00088      * This method returns the type of configuration. Cast this instance
00089      * appropriately to have access to specific getters.
00090      * @return the type of this config.
00091      * \endenglish
00092      */
00093      ConfigType getConfigType();
00094      void setConfigType(ConfigType type);
00095
00096      bool getUseJoints();
00097      void setUseJoints(bool use);
00098
00099      /**
00100      * \chinese
00101      * <p> 为 MoveJ 配置设置关节速度参数。</p>
00102      * 该参数适用于机械臂从上一个位置经过 Move 节点下的路点的运动。
00103      *
00104      * @param joint_speed 要达到的共享关节速度
00105      * \endchinese
00106      * \english
00107      * <p>Set joint speed parameter for the MoveJ configuration.</p>
00108      * The parameter applies to the movement of the robot arm, from the previous
00109      * position through the waypoints under the Move node.
00110      *
00111      * @param joint_speed The shared joint speed to be achieved.
00112      * \endenglish
00113      */
00114      void setJointSpeed(double joint_speed);
00115      double getJointSpeed();
00116
00117      /**
00118      * \chinese
00119      * <p> 为 MoveJ 配置设置关节加速度参数。</p>
00120      * 该参数适用于机械臂从上一个位置经过 Move 节点下的路点的运动。
00121      * @param joint_acceleration 要使用的共享关节加速度
00122      * \endchinese
00123      * \english
00124      * <p>Set joint acceleration parameter for the MoveJ configuration.</p>
00125      * The parameter applies to the movement of the robot arm, from the previous
00126      * position through the waypoints under the Move node.
00127      * @param joint_acceleration The shared joint acceleration to be used.
00128      * \endenglish
00129      */
00130      void setJointAcceleration(double joint_acceleration);
00131      double getJointAcceleration();
00132
00133      /**
00134      * \chinese
00135      * <p> 为 MoveJ 配置设置 TCP 选择。</p>
00136      * <p> 在执行机械臂运动之前, TCP 将被设置为活动 TCP。</p>
00137      * @param tcp_selection 要应用的 TCP 设置
00138      * \endchinese
00139      * \english
00140      * <p>Set the TCP selection for the MoveJ configuration.</p>
00141      * <p>The TCP will be set as the active TCP before the movement of the robot
00142      * arm is performed.</p>
00143      * @param tcp_selection The TCP setting to be applied.
00144      * \endenglish
00145      */
00146      void setTCPSelection(TCPPtr tcp_selection);
00147      TCPPtr getTCPSelection();
00148
00149      /**
00150      * \chinese
00151      * <p> 为 MoveL 配置设置工具速度参数。</p>
00152      * 该参数适用于机械臂从上一个位置经过 Move 节点下的路点的运动。
00153      * @param tool_speed 要达到的共享工具速度
00154      * \endchinese
00155      * \english
00156      * <p>Set tool speed parameter for the MoveL configuration.</p>
00157      * The parameter applies to the movement of the robot arm, from the previous
00158      * position through the waypoints under the Move node.
00159      * @param tool_speed The shared tool speed to be achieved.
00160      * \endenglish
00161      */
00162      void setToolSpeed(double tool_speed);
00163      double getToolSpeed();
00164
00165      /**
00166      * \chinese
00167      * <p> 为 MoveL 配置设置工具加速度参数。</p>
00168      * 该参数适用于机械臂从上一个位置经过 Move 节点下的路点的运动。
00169      * @param tool_acceleration 要使用的共享工具加速度
00170      * \endchinese
00171      * \english
00172      * <p>Set tool acceleration parameter for the MoveL configuration.</p>
00173      * The parameter applies to the movement of the robot arm, from the previous
00174      * position through the waypoints under the Move node.

```

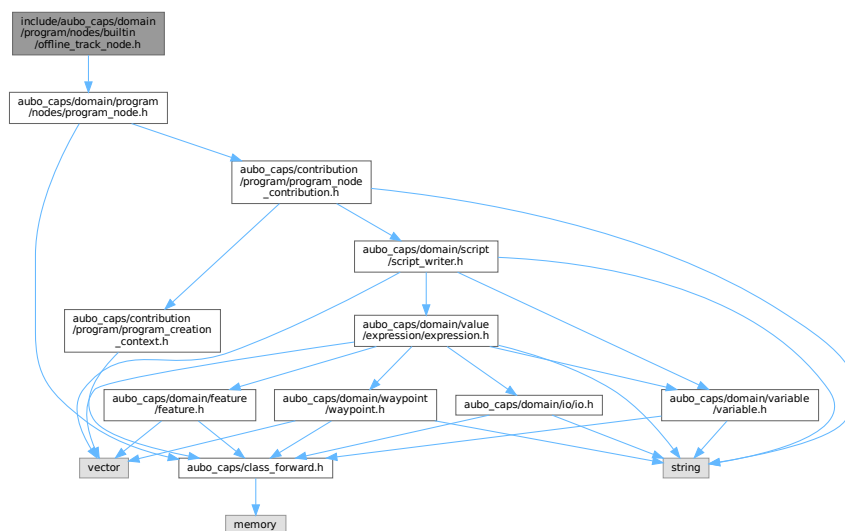
```

00175     * @param tool_acceleration The shared tool acceleration to be used.
00176     * \endenglish
00177     */
00178 void setToolAcceleration(double tool_acceleration);
00179 double getToolAcceleration();
00180
00181 /**
00182  * \chinese
00183  * <p> 为 MoveL 配置设置坐标系。</p>
00184  * 该坐标系将用于机械臂从上一个位置经过 Move 节点下的路点的运动。
00185  * @param feature 此运动的共享坐标系
00186  * \endchinese
00187  * \english
00188  * <p>Set feature for the MoveL configuration.</p>
00189  * The feature will be used for the movement of the robot arm, from the
00190  * previous position through the waypoints under the Move node.
00191  * @param feature The shared feature for this movement.
00192  * \endenglish
00193  */
00194 void setFeature(FeaturePtr feature);
00195 void setFeature(FeatureVariablePtr feature_variable);
00196 FeaturePtr getFeature();
00197
00198 /**
00199  * \chinese
00200  * <p> 为 MoveP 配置设置交融参数。</p>
00201  * 交融参数将用于机械臂运动中所有路点之间的过渡。
00202  * @param blend 此 Move 节点运动中使用的共享交融值
00203  * \endchinese
00204  * \english
00205  * <p>Set the blend parameter for the MoveP configuration.</p>
00206  * The blend parameter will be used between all waypoints in the movement of
00207  * the robot arm.
00208  * @param blend The shared blend used in the motion of this Move node.
00209  * \endenglish
00210  */
00211 void setBlend(double blend);
00212 double getBlend();
00213
00214 void setTcpSelectionType(TcpSelectionType type);
00215 TcpSelectionType getTcpSelectionType();
00216
00217 private:
00218     friend class DataSwitch;
00219     MoveNode();
00220     void *d_{ nullptr };
00221 };
00222
00223 } // namespace aubo_scope
00224 } // namespace arcs
00225 #endif // AUBO_SCOPE_MOVE_NODE_H

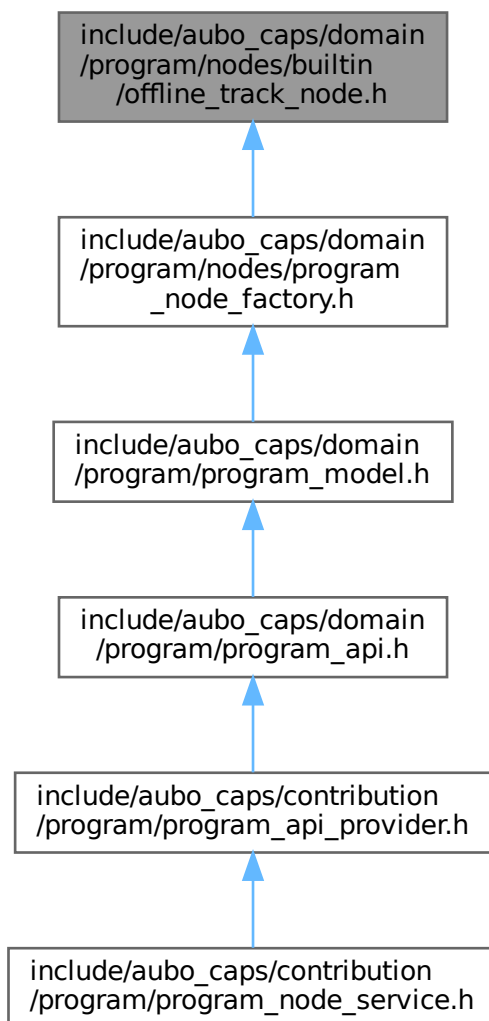
```

12.164 include/aubo_caps/domain/program/nodes/builtin/↵ offline_track_node.h 文件参考

`#include <aubo_caps/domain/program/nodes/program_node.h>`
offline_track_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::OfflineTrackNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(OfflineTrackNode\)](#)

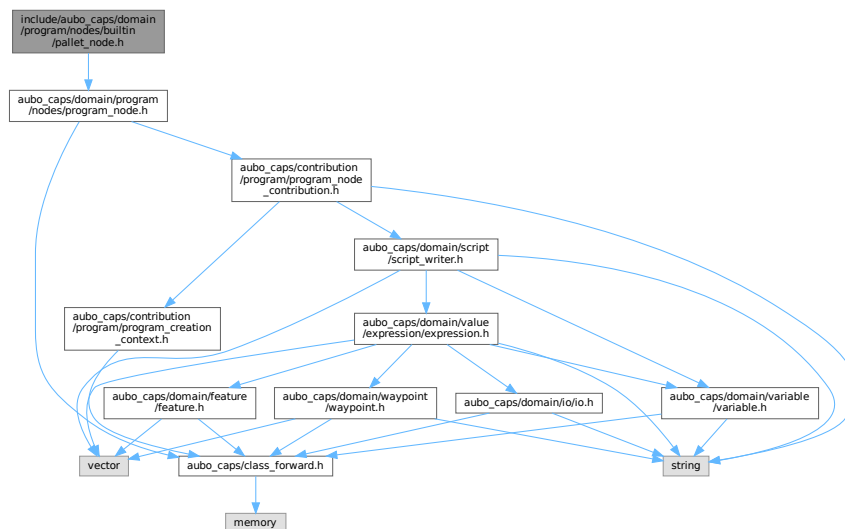
12.165 offline_track_node.h

[浏览该文件的文档.](#)

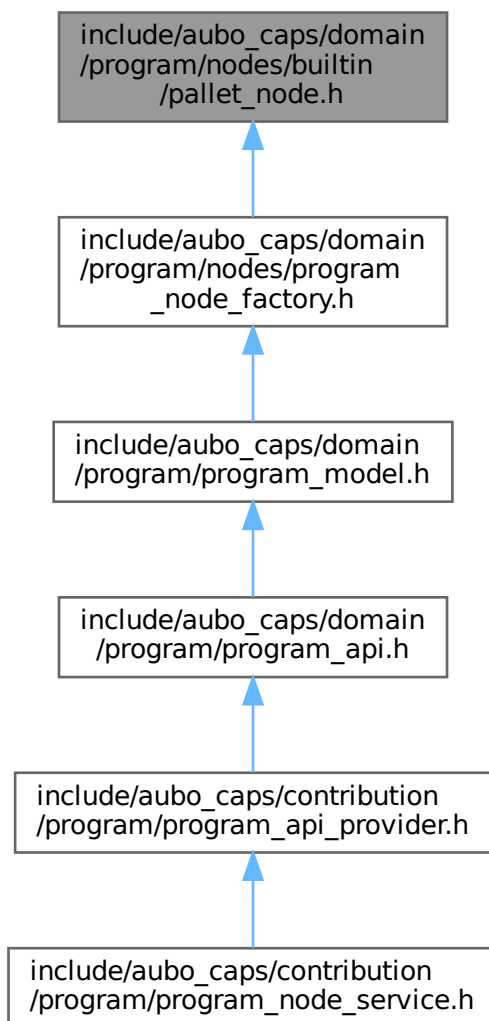

```
00001 #ifndef AUBO_SCOPE_OFFLINE_TRACK_NODE_H
00002 #define AUBO_SCOPE_OFFLINE_TRACK_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(OfflineTrackNode);
00009
00010 class ARCS_ABI_EXPORT OfflineTrackNode : public ProgramNode
00011 {
00012 public:
00013     OfflineTrackNode(OfflineTrackNode &f);
00014     OfflineTrackNode(OfflineTrackNode &&f);
00015     virtual ~OfflineTrackNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     OfflineTrackNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_OFFLINE_TRACK_NODE_H
```

12.166 include/aubo_caps/domain/program/nodes/builtin/↔ pallet_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
pallet_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::PalletNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(PalletNode\)](#)

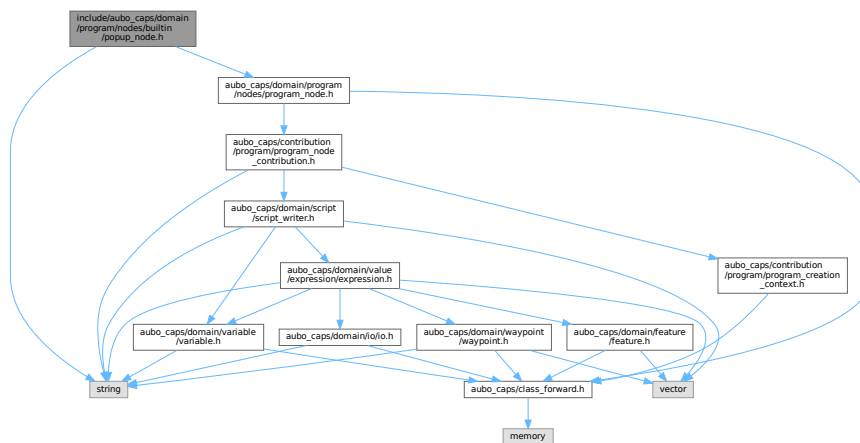
12.167 pallet_node.h

[浏览该文件的文档.](#)

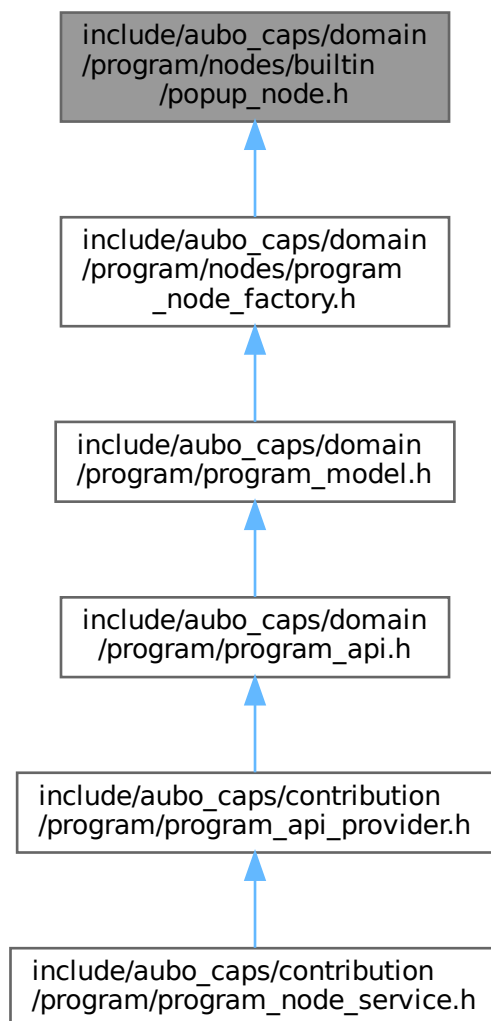
```
00001 #ifndef AUBO_SCOPE_PALLET_NODE_H
00002 #define AUBO_SCOPE_PALLET_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(PalletNode);
00009
00010         class ARCS_ABI_EXPORT PalletNode : public ProgramNode
00011         {
00012         public:
00013             PalletNode(PalletNode &f);
00014             PalletNode(PalletNode &&f);
00015             virtual ~PalletNode();
00016
00017         private:
00018             friend class DataSwitch;
00019             PalletNode();
00020             void *d_{ nullptr };
00021         };
00022
00023     } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_PALLET_NODE_H
```

12.168 include/aubo_caps/domain/program/nodes/builtin/↵ popup_node.h 文件参考

```
#include <string>
#include <aubo_caps/domain/program/nodes/program_node.h>
popup_node.h 的引用 (Include) 关系图:
```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::PopupNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(PopupNode\)](#)

12.169 popup_node.h

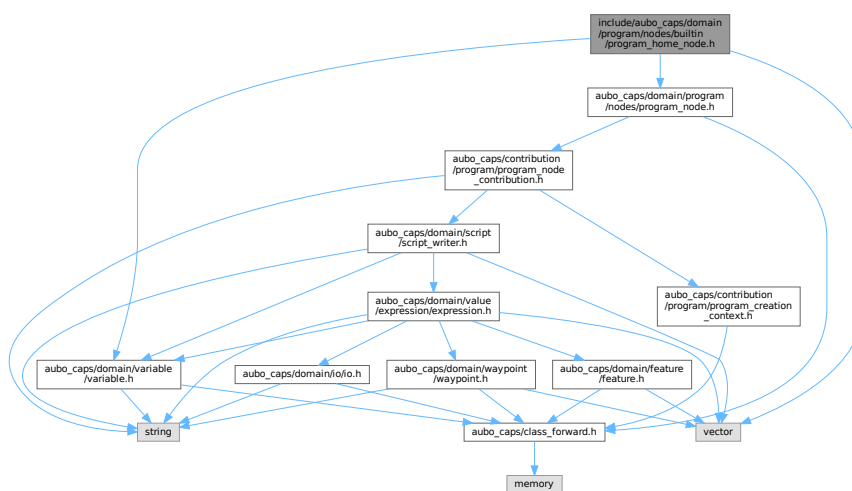
[浏览该文件的文档.](#)

```
00001 #ifndef AUBO_SCOPE_POPUP_NODE_H
00002 #define AUBO_SCOPE_POPUP_NODE_H
00003
00004 #include <string>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(PopupNode);
00010
00011 class ARCS_ABI_EXPORT PopupNode : public ProgramNode
00012 {
00013 public:
00014     enum Type
00015     {
00016         MESSAGE,
00017         WARNING,
00018         ERROR
00019     };
00020
00021     PopupNode(PopupNode &f);
00022     PopupNode(PopupNode &&f);
00023     virtual ~PopupNode();
00024
00025     /**
00026      *
00027      * @param message the message in the pop up for the end user
00028      * @return this node
00029      * @throws IllegalStateException if called from a Qt-based AuboCap
00030      * outside of an {@link UndoableChanges} scope (see also {@link
00031      * UndoRedoManager})
00032      */
00033
00034     void setMessage(std::string message);
00035     std::string getMessage();
00036     /**
00037      *
00038      * @param type the type of pop up
00039      * @return this node
00040      * @throws IllegalStateException if called from a Qt-based AuboCap
00041      * outside of an {@link UndoableChanges} scope (see also {@link
00042      * UndoRedoManager})
00043      */
00044     void setType(Type type);
00045     Type getType() const;
00046
00047 private:
00048     friend class DataSwitch;
00049     PopupNode();
00050     void *d_{ nullptr };
00051 };
00052
00053 } // namespace aubo_scope
00054 } // namespace arcs
00055 #endif // AUBO_SCOPE_POPUP_NODE_H
```

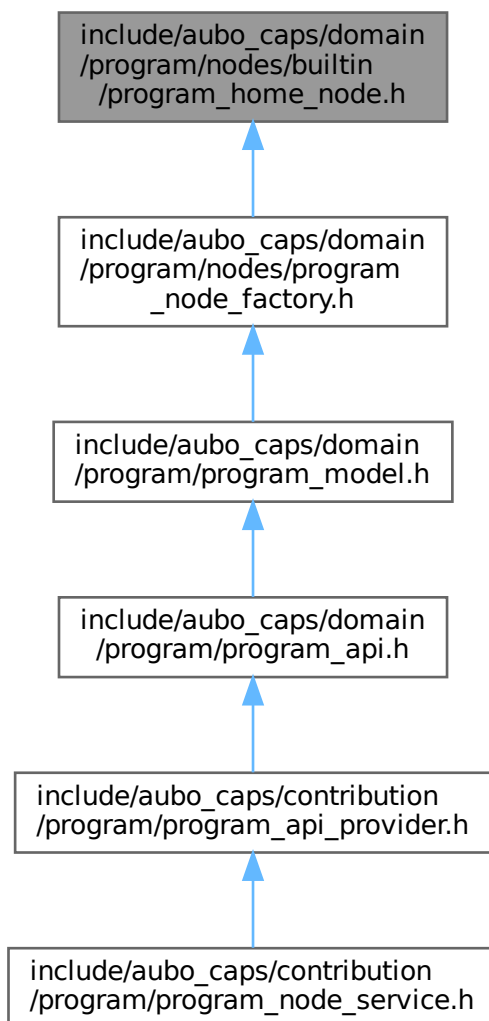
12.170 include/aubo_caps/domain/program/nodes/builtin/↵ program_home_node.h 文件参考

```
#include <vector>
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/variable/variable.h>
```

program_home_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ProgramHomeNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ProgramHomeNode\)](#)

12.171 program_home_node.h

[浏览该文件的文档.](#)

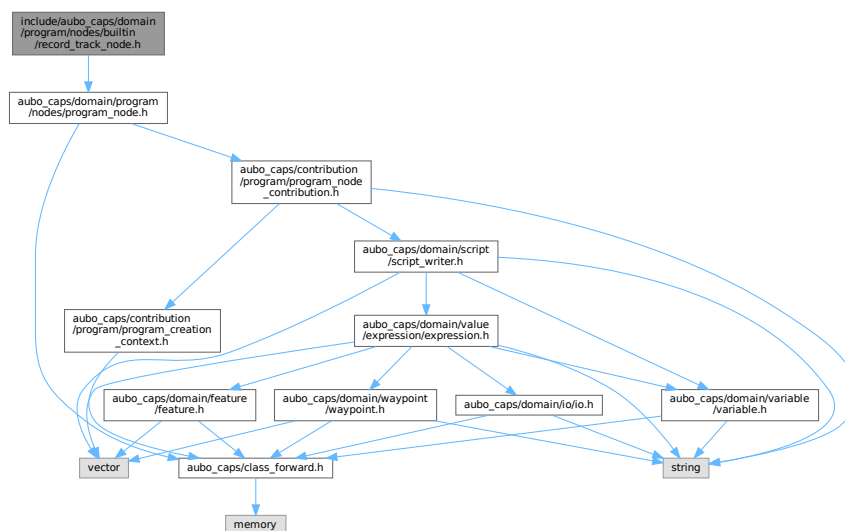
```

00001 #ifndef AUBO_SCOPE_PROGRAM_HOME_NODE_H
00002 #define AUBO_SCOPE_PROGRAM_HOME_NODE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006 #include <aubo_caps/domain/variable/variable.h>
00007
00008 namespace arcs {
00009     namespace aubo_scope {
00010         ARCS_CLASS_FORWARD(ProgramHomeNode);
00011
00012         class ARCS_ABI_EXPORT ProgramHomeNode : public ProgramNode
00013         {
00014         public:
00015
00016             ProgramHomeNode(ProgramHomeNode &f);
00017             ProgramHomeNode(ProgramHomeNode &&f);
00018             virtual ~ProgramHomeNode();
00019
00020         private:
00021             friend class DataSwitch;
00022             ProgramHomeNode();
00023             void *d_{ nullptr };
00024         };
00025     } // namespace aubo_scope
00026 } // namespace arcs
00027 #endif // AUBO_SCOPE_PROGRAM_HOME_NODE_H

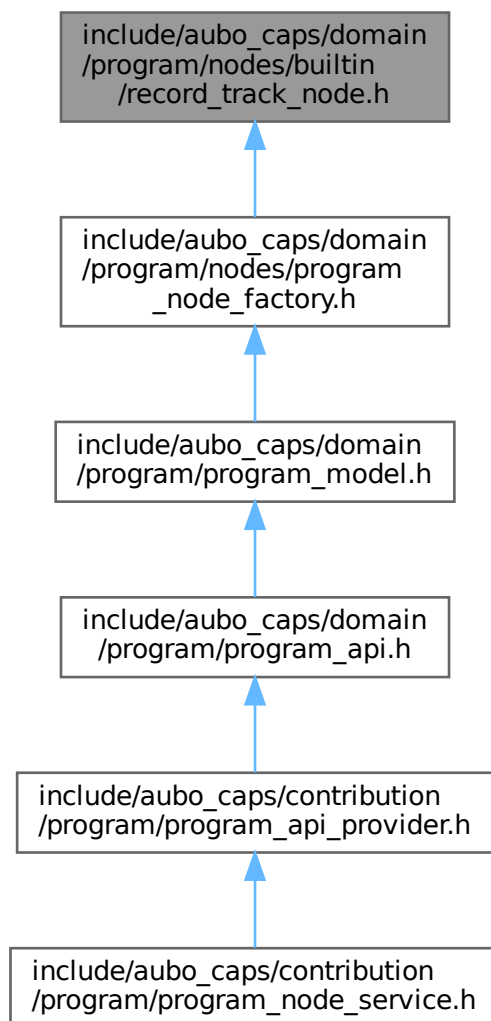
```

12.172 include/aubo_caps/domain/program/nodes/builtin/record_track_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
record_track_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::RecordTrackNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(RecordTrackNode\)](#)

12.173 record_track_node.h

[浏览该文件的文档.](#)

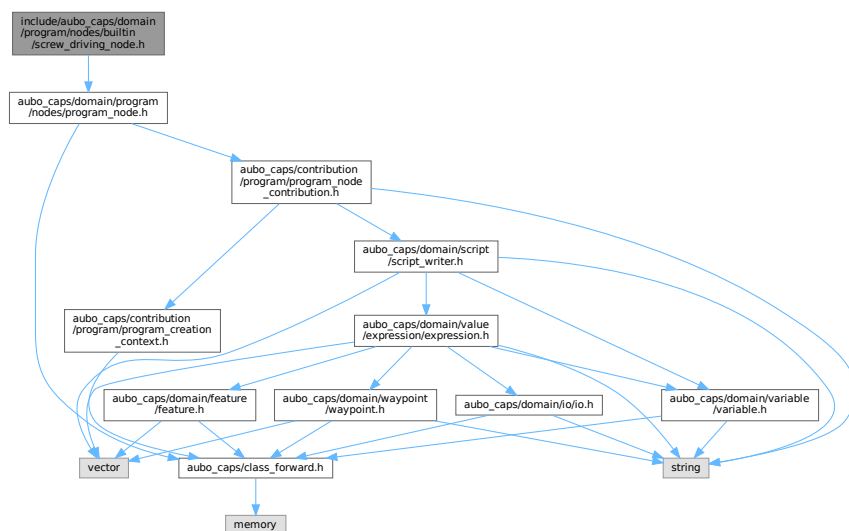
```

00001 #ifndef AUBO_SCOPE_RECORD_TRACK_NODE_H
00002 #define AUBO_SCOPE_RECORD_TRACK_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(RecordTrackNode);
00009
00010         class ARCS_ABI_EXPORT RecordTrackNode : public ProgramNode
00011         {
00012         public:
00013             RecordTrackNode(RecordTrackNode &f);
00014             RecordTrackNode(RecordTrackNode &&f);
00015             virtual ~RecordTrackNode();
00016
00017         private:
00018             friend class DataSwitch;
00019             RecordTrackNode();
00020             void *d_{ nullptr };
00021         };
00022
00023     } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_RECORD_TRACK_NODE_H

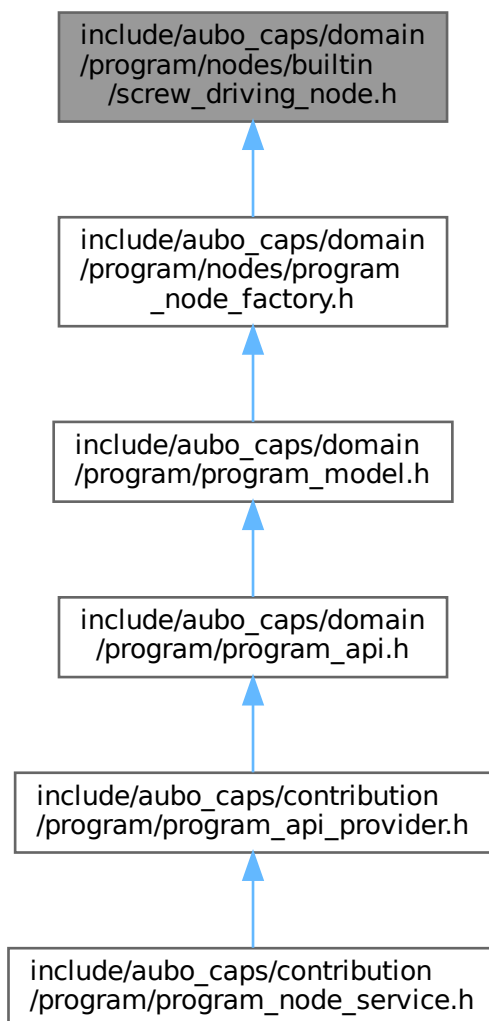
```

12.174 include/aubo_caps/domain/program/nodes/builtin/ ← screw_driving_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
screw_driving_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ScrewdrivingNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ScrewdrivingNode](#))

12.175 screw_driving_node.h

[浏览该文件的文档.](#)

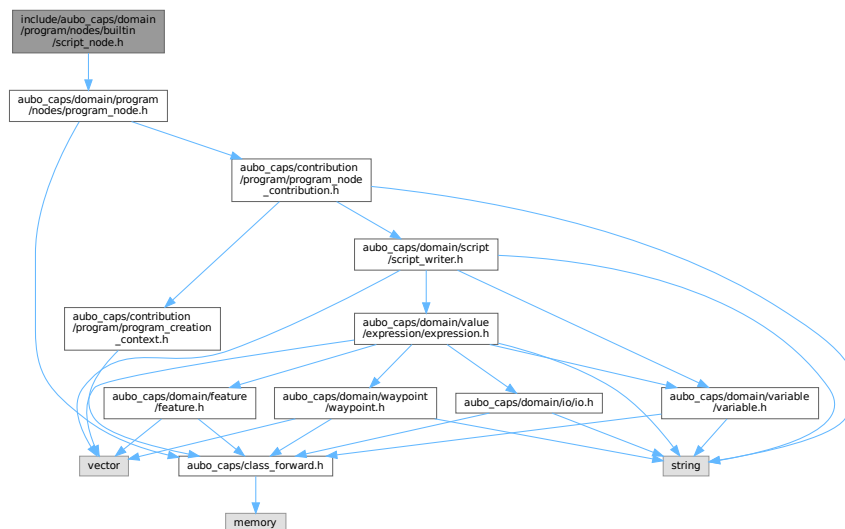
```

00001 #ifndef AUBO_SCOPE_SCREW_DRIVING_NODE_H
00002 #define AUBO_SCOPE_SCREW_DRIVING_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(ScrewdrivingNode);
00009
00010 class ARCS_ABI_EXPORT ScrewdrivingNode : public ProgramNode
00011 {
00012 public:
00013     ScrewdrivingNode(ScrewdrivingNode &f);
00014     ScrewdrivingNode(ScrewdrivingNode &&f);
00015     virtual ~ScrewdrivingNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     ScrewdrivingNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_SCREW_DRIVING_NODE_H

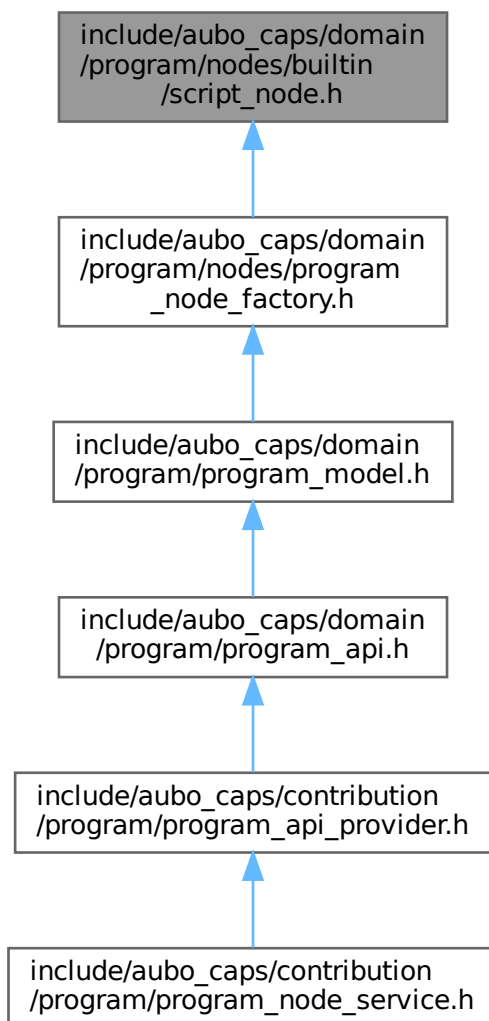
```

12.176 include/aubo_caps/domain/program/nodes/builtin/script_node.h 文件参考 ↵

#include <aubo_caps/domain/program/nodes/program_node.h>
script_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ScriptNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ScriptNode\)](#)

12.177 script_node.h

[浏览该文件的文档.](#)

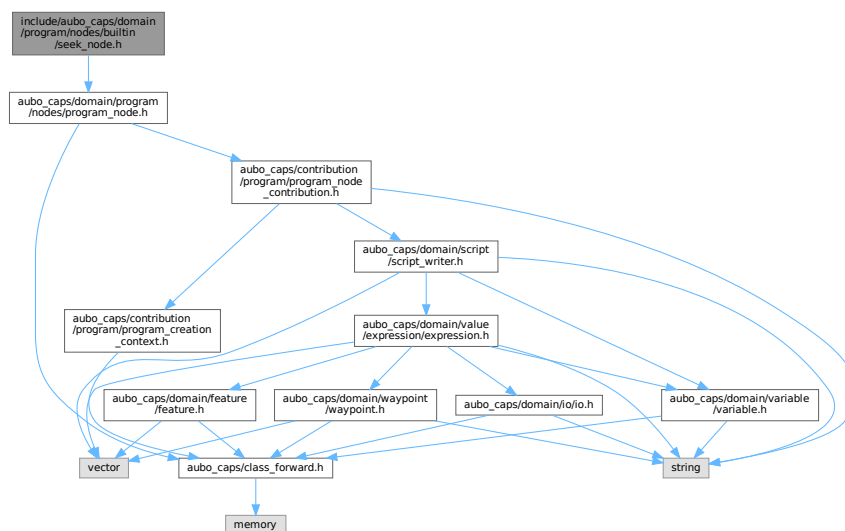
```

00001 #ifndef AUBO_SCOPE_SCRIPT_NODE_H
00002 #define AUBO_SCOPE_SCRIPT_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(ScriptNode);
00009
00010 class ARCS_ABI_EXPORT ScriptNode : public ProgramNode
00011 {
00012 public:
00013     ScriptNode(ScriptNode &f);
00014     ScriptNode(ScriptNode &&f);
00015     virtual ~ScriptNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     ScriptNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_SCRIPT_NODE_H

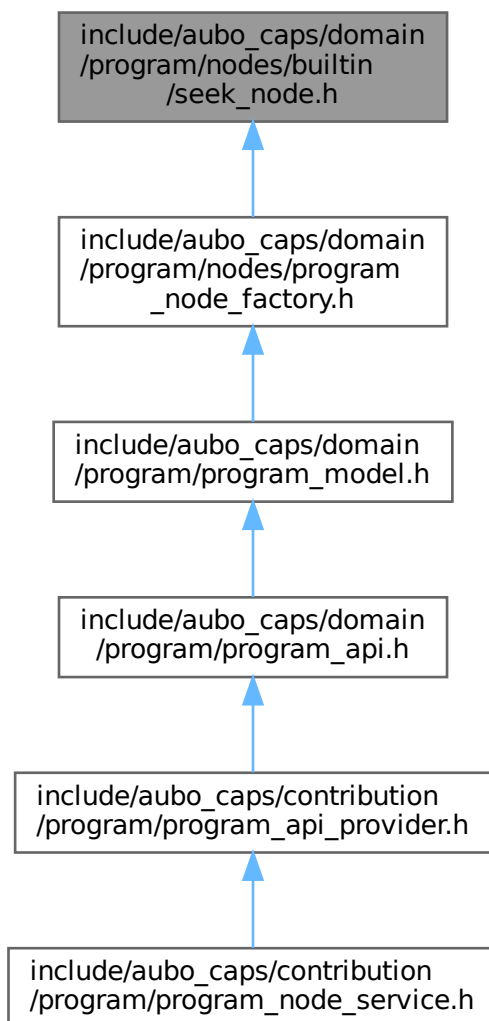
```

12.178 include/aubo_caps/domain/program/nodes/builtin/seek_node.h 文件参考 ↩

#include <aubo_caps/domain/program/nodes/program_node.h>
 seek_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::SeekNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(SeekNode\)](#)

12.179 seek_node.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_SEEK_NODE_H
00002 #define AUBO_SCOPE_SEEK_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(SeekNode);
00009
00010         class ARCS_ABI_EXPORT SeekNode : public ProgramNode
00011         {
00012         public:
00013             SeekNode(SeekNode &f);
00014             SeekNode(SeekNode &&f);
00015             virtual ~SeekNode();
00016
00017         private:
00018             friend class DataSwitch;
00019             SeekNode();
00020             void *d_{ nullptr };
00021         };
00022
00023     } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_SEEK_NODE_H

```

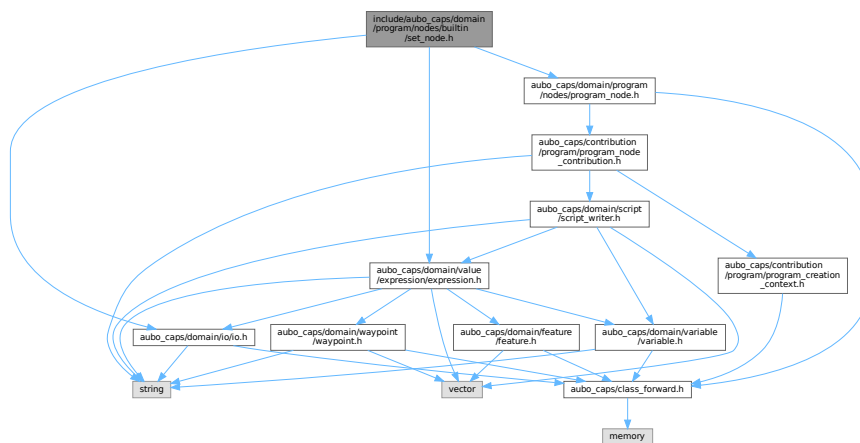
12.180 include/aubo_caps/domain/program/nodes/builtin/set_node.h 文件参考

```

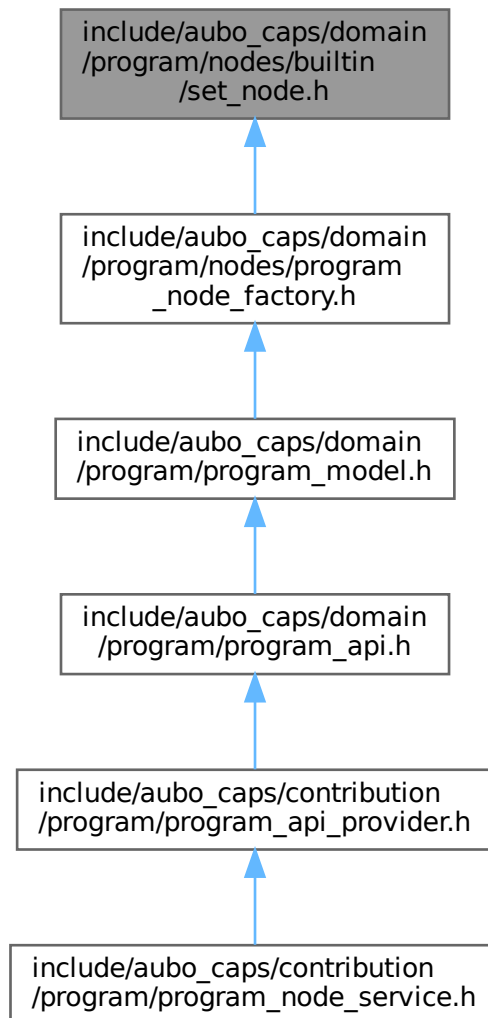
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/value/expression/expression.h>
#include <aubo_caps/domain/io/io.h>

```

set_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::SetNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(SetNode\)](#)

12.181 set_node.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_SET_NODE_H
00002 #define AUBO_SCOPE_SET_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006 #include <aubo_caps/domain/io/io.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(SetNode);
00011
00012 class ARCS_ABI_EXPORT SetNode : public ProgramNode
00013 {
00014 public:
00015     /**
00016      * The configuration type used to determine which type of configuration this
00017      * instance is.
00018      */
00019     enum ConfigType
00020     {
00021         /**
00022          * No selection has been made. This type has no further information.
00023          */
00024         NO_TYPE,
00025
00026         /**
00027          * No action has been selected. This type has no further information.
00028          */
00029         NO_ACTION,
00030
00031         /**
00032          * <p>
00033          * Digital output has been selected.
00034          * </p>
00035          *
00036          * The config instance can be cast to {@link
00037          * DigitalOutputSetNodeConfig}.
00038          */
00039         DIGITAL_OUTPUT,
00040
00041         /**
00042          * <p>
00043          * Analog output configured for electric current or no output has been
00044          * selected.
00045          * </p>
00046          *
00047          * The config instance can be cast to {@link
00048          * AnalogOutputCurrentSetNodeConfig}.
00049          */
00050         ANALOG_OUTPUT_CURRENT,
00051
00052         /**
00053          * <p>
00054          * Analog output configured for voltage has been selected.
00055          * </p>
00056          *
00057          * The config instance can be cast to {@link
00058          * AnalogOutputVoltageSetNodeConfig}.
00059          */
00060         ANALOG_OUTPUT_VOLTAGE,
00061
00062         /**
00063          * <p>
00064          * Register using float type values has been selected.
00065          * </p>
00066          *
00067          * The config instance can be cast to {@link
00068          * FloatRegisterOutputSetNodeConfig}.
00069          */
00070         FLOAT_REGISTER_OUTPUT,
00071
00072         /**
00073          * <p>
00074          * Expression output has been selected.
00075          * </p>
00076          *
00077          * The config instance can be cast to {@link
00078          * ExpressionOutputSetNodeConfig}.
00079          */
00080         EXPRESSION_OUTPUT,
00081
00082         /**
00083          * <p>
00084          * Single digital pulse output has been selected.
00085          * </p>
00086          *
00087          * The config instance can be cast to {@link

```

```

00088     * SingleDigitalPulseSetNodeConfig}.
00089     */
00090     SINGLE_DIGITAL_PULSE,
00091
00092     /**
00093     * <p>
00094     * Increment variable has been selected. This type has no further
00095     * information.
00096     * </p>
00097     *
00098     * Setting this type of config will be ignored and will have no effect
00099     * on the node.
00100     */
00101     INCREMENT_VARIABLE,
00102 };
00103
00104 enum OutputSelectionType
00105 {
00106
00107     /**
00108     * No output is selected.
00109     */
00110     NO_OUTPUT,
00111
00112     /**
00113     * The selected output is no longer available (e.g. because the end user
00114     * has removed the output in the Installation or the needed Installation
00115     * is not currently loaded).
00116     */
00117     UNRESOLVED_OUTPUT,
00118
00119     /**
00120     * <p>
00121     * Output is selected.
00122     * </p>
00123     *
00124     * The Selection instance can be cast to {@link ExpressionOutput}.
00125     */
00126     OUTPUT
00127 };
00128 SetNode(SetNode &f);
00129 SetNode(SetNode &&f);
00130 virtual ~SetNode();
00131
00132 /**
00133  * \chinese
00134  * 返回配置类型。适当转换此实例以访问特定的 getter。
00135  * @return 此配置的类型
00136  * \endchinese
00137  * \english
00138  * This method returns the type of configuration. Cast this instance
00139  * appropriately to have access to specific getters.
00140  * @return the type of this config.
00141  * \endenglish
00142  */
00143 ConfigType getConfigType();
00144 void setConfigType(ConfigType type);
00145
00146 /**
00147  * @return the time for the length of the pulse.
00148  */
00149 double getPulseTime();
00150 void setPulseTime(double time);
00151
00152 /**
00153  *
00154  * @return the float value to be applied to the float register output.
00155  */
00156 float getFloatValueToSet();
00157 void setFloatValueToSet(float value);
00158
00159 /**
00160  * @return the analog output value.
00161  */
00162 double getAnalogOutput();
00163 void setAnalogOutput(double value);
00164
00165 OutputSelectionType getOutputSelectionType();
00166
00167 /**
00168  *
00169  * @return the expression whose evaluation will be applied to the output.
00170  */
00171 ExpressionPtr getExpression();
00172 void setExpression(ExpressionPtr expression);
00173
00174 /**

```

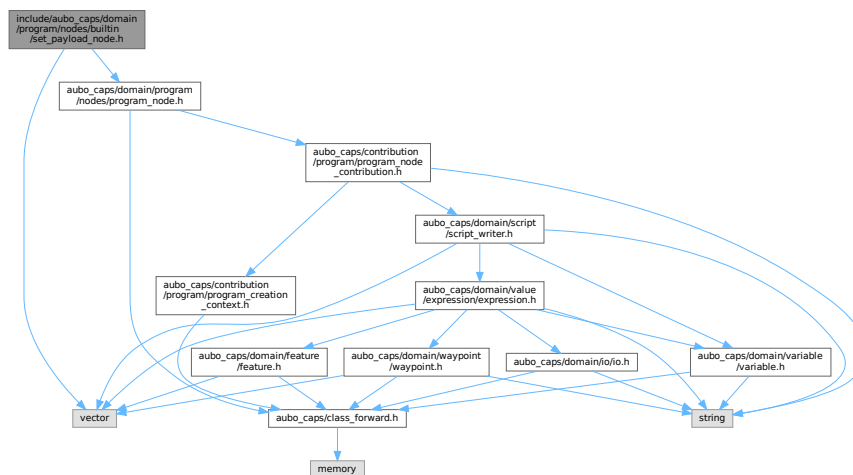
```

00175     * The output selected can be an analog output, a digital output, a MODBUS
00176     * output or a register output.
00177     *
00178     * @return the output to which the result of the evaluation of the
00179     * expression will be applied.
00180     */
00181     IntPtr getOutput();
00182     void setOutput(IntPtr io);
00183
00184     /**
00185     * Depending on the type of output this is interpreted in different ways.
00186     * For digital output and MODBUS output,
00187     * <code>true</code> means the output will be set to HIGH,
00188     * <code>false</code> means the output will be set to LOW. For boolean
00189     * register the value is interpreted literally.
00190     *
00191     * @return the output value.
00192     */
00193     bool getBoolValueToSet();
00194     void setBoolValueToSet(bool value);
00195
00196     /**
00197     *
00198     * @return the voltage value/level to be applied to the output
00199     */
00200     double getVoltage();
00201     void setVoltage(double voltage);
00202
00203     /**
00204     *
00205     * @return the electric current value/level to be applied to the output.
00206     */
00207     double getCurrent();
00208     void setCurrent(double curr);
00209
00210 private:
00211     friend class DataSwitch;
00212     SetNode();
00213     void *d_{ nullptr };
00214 };
00215
00216 } // namespace aubo_scope
00217 } // namespace arcs
00218 #endif // AUBO_SCOPE_SET_NODE_H

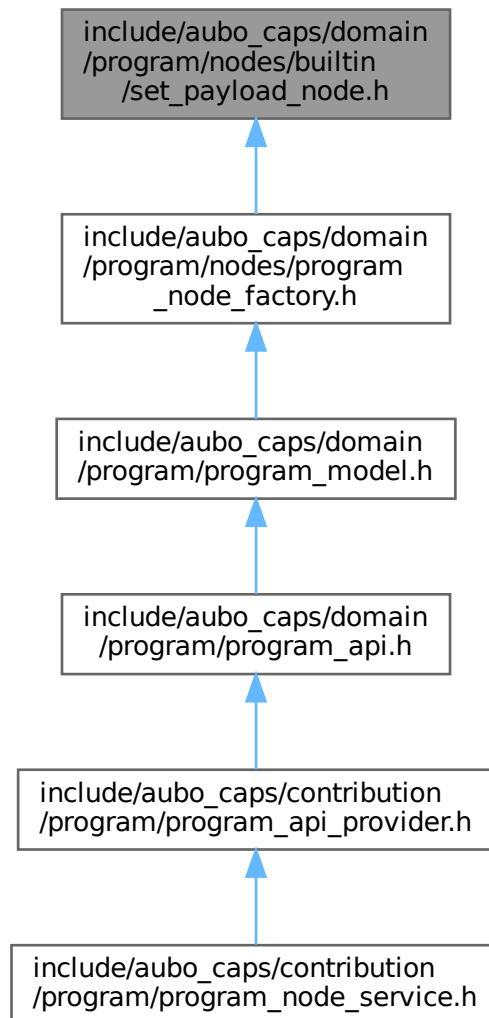
```

12.182 include/aubo_caps/domain/program/nodes/builtin/↵ set_payload_node.h 文件参考

#include <vector>
#include <aubo_caps/domain/program/nodes/program_node.h>
set_payload_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::SetPayloadNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(SetPayloadNode\)](#)

12.183 set_payload_node.h

[浏览该文件的文档.](#)

```

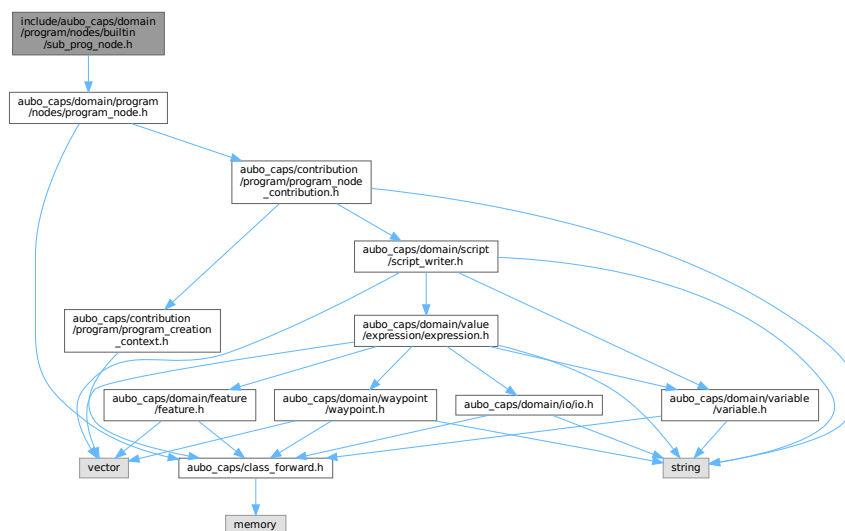
00001 #ifndef AUBO_SCOPE_SET_PAYLOAD_NODE_H
00002 #define AUBO_SCOPE_SET_PAYLOAD_NODE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(SetPayloadNode);
00010
00011 class ARCS_ABI_EXPORT SetPayloadNode : public ProgramNode
00012 {
00013 public:
00014 /**
00015  * This base interface represents a configuration of the Set Payload program
00016  * node (i.e. a {@link SetPayloadNode}
00017  * instance).
00018  */
00019 enum ConfigType
00020 {
00021 /**
00022  * <p>
00023  * A payload (from the installation) is selected.
00024  * </p>
00025  *
00026  * The config instance can be cast to {@link
00027  * SelectionPayloadNodeConfig}.
00028  */
00029 PAYLOAD_SELECTION,
00030
00031 /**
00032  * <p>
00033  * Custom payload parameters are specified.
00034  * </p>
00035  *
00036  * The config instance can be cast to {@link
00037  * CustomParametersPayloadNodeConfig}.
00038  */
00039 CUSTOM_PARAMETERS
00040 };
00041
00042 SetPayloadNode(SetPayloadNode &f);
00043 SetPayloadNode(SetPayloadNode &&f);
00044 virtual ~SetPayloadNode();
00045
00046 /**
00047  * \chinese
00048  * 返回配置类型。适当转换此实例以访问特定的 getter。
00049  * @return 此配置的类型
00050  * \endchinese
00051  * \english
00052  * This method returns the type of configuration. Cast this instance
00053  * appropriately to have access to specific getters.
00054  * @return The type of this config.
00055  * \endenglish
00056  */
00057 ConfigType getConfigType();
00058 void setConfigType(ConfigType type);
00059
00060 /**
00061  * \chinese
00062  * @return 负载的总质量
00063  * \endchinese
00064  * \english
00065  * @return The total mass of the payload
00066  * \endenglish
00067  */
00068 double getMass();
00069 void setMass(double mass);
00070
00071 /**
00072  * \chinese
00073  * 获取负载的重心 (CoG)。它定义为工具输出法兰中心与附着负载重心之间的偏移量。
00074  * @return 相对于工具输出法兰中心的负载重心 (CoG)
00075  * \endchinese
00076  * \english
00077  * Gets the payload's center of gravity (CoG), also referred to as center of
00078  * mass. It is defined as the offset between the center of the tool output
00079  * flange and the center of gravity of the attached payload.
00080  * @return The center of gravity (CoG) of the payload relative to the center
00081  * of the tool output flange
00082  * \endenglish
00083  */
00084 std::vector<double> getCenterOfGravity();
00085 void setCenterOfGravity(const std::vector<double> &cog);
00086
00087 private:

```

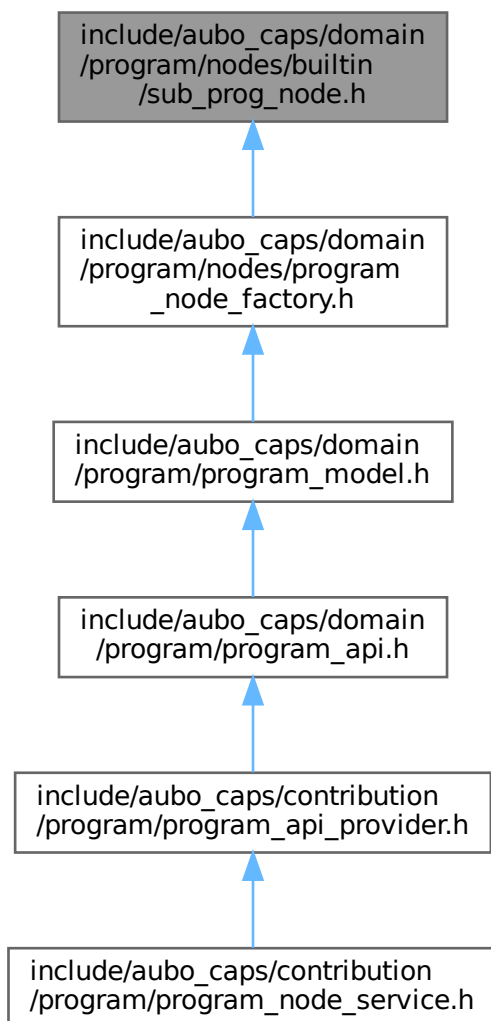
```
00088     friend class DataSwitch;
00089     SetPayloadNode();
00090     void *d_{ nullptr };
00091 };
00092
00093 } // namespace aubo_scope
00094 } // namespace arcs
00095 #endif // AUBO_SCOPE_SET_PAYLOAD_NODE_H
```

12.184 include/auco_caps/domain/program/nodes/builtin/↵ sub_prog_node.h 文件参考

#include <auco_caps/domain/program/nodes/program_node.h>
 sub_prog_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::SubProgNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(SubProgNode\)](#)

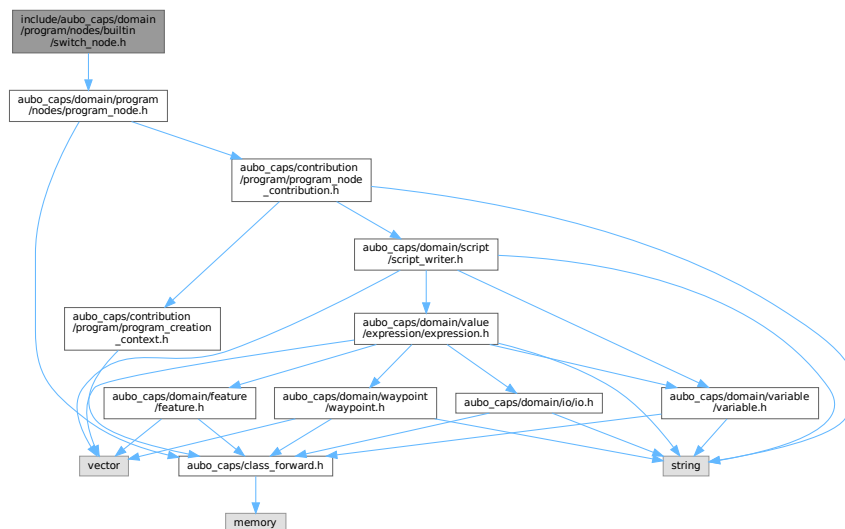
12.185 sub_prog_node.h

[浏览该文件的文档.](#)

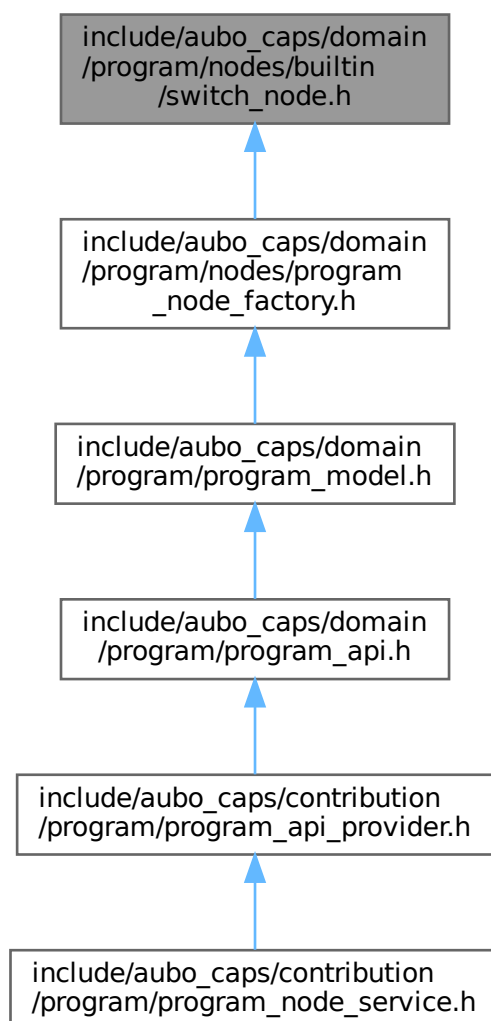

```
00001 #ifndef AUBO_SCOPE_SUB_PROG_NODE_H
00002 #define AUBO_SCOPE_SUB_PROG_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(SubProgNode);
00009
00010 class ARCS_ABI_EXPORT SubProgNode : public ProgramNode
00011 {
00012 public:
00013     SubProgNode(SubProgNode &f);
00014     SubProgNode(SubProgNode &&f);
00015     virtual ~SubProgNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     SubProgNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_SUB_PROG_NODE_H
```

12.186 include/aubo_caps/domain/program/nodes/builtin/↔ switch_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
switch_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::SwitchNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(SwitchNode\)](#)

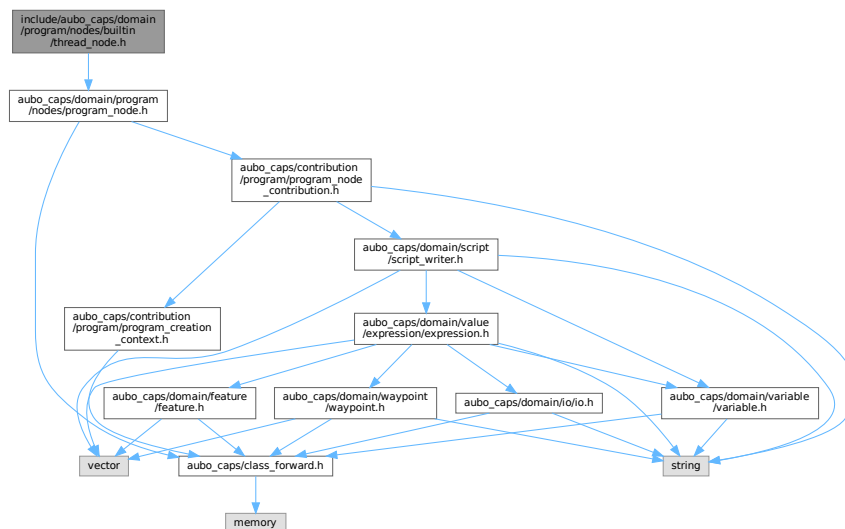
12.187 switch_node.h

[浏览该文件的文档.](#)

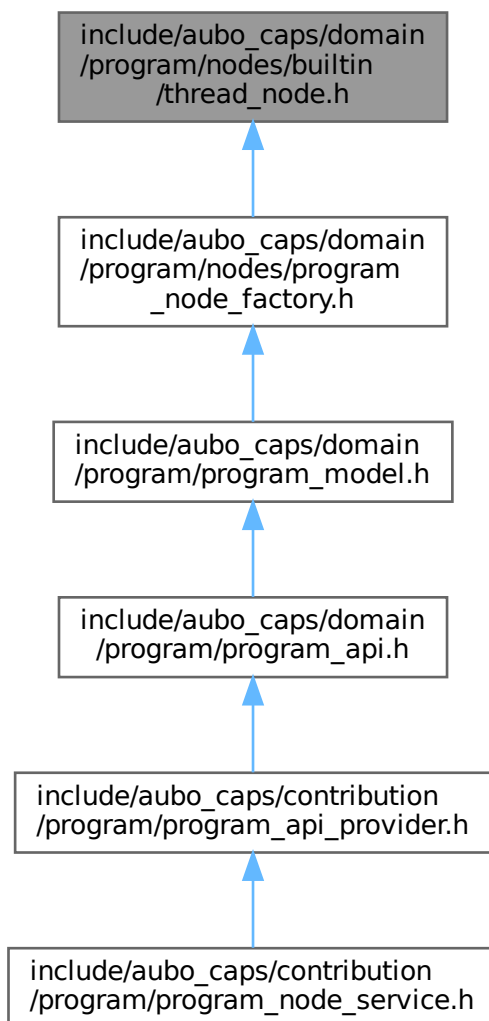
```
00001 #ifndef AUBO_SCOPE_SWITCH_NODE_H
00002 #define AUBO_SCOPE_SWITCH_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(SwitchNode);
00009
00010 class ARCS_ABI_EXPORT SwitchNode : public ProgramNode
00011 {
00012 public:
00013     SwitchNode(SwitchNode &f);
00014     SwitchNode(SwitchNode &&f);
00015     virtual ~SwitchNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     SwitchNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_SWITCH_NODE_H
```

12.188 include/aubo_caps/domain/program/nodes/builtin/↔ thread_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
thread_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ThreadNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ThreadNode\)](#)

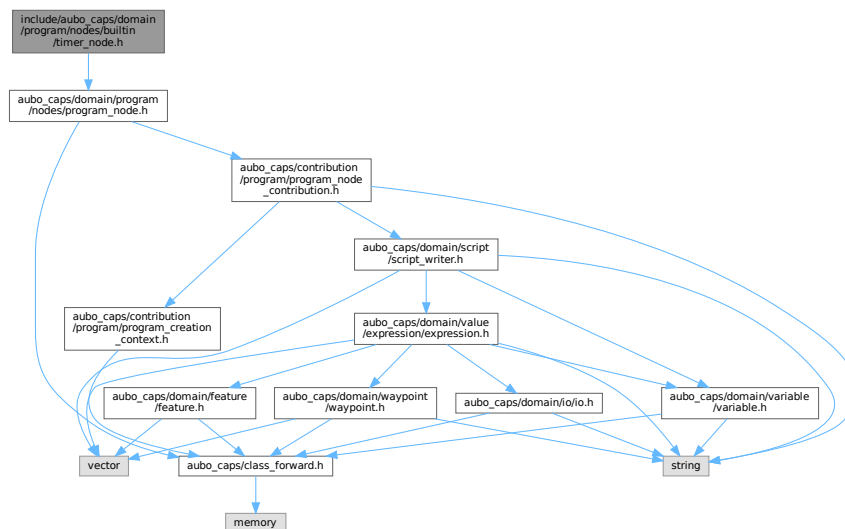
12.189 thread_node.h

[浏览该文件的文档.](#)

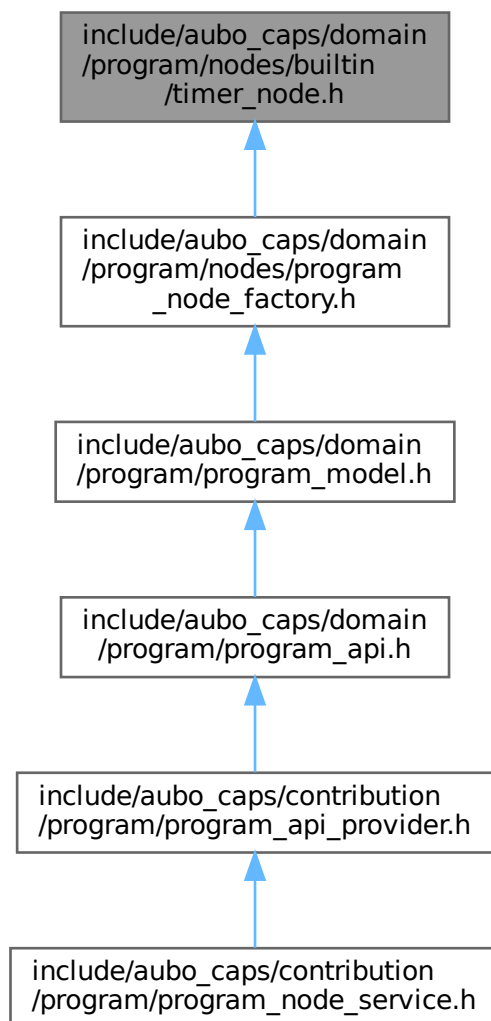
```
00001 #ifndef AUBO_SCOPE_THREAD_NODE_H
00002 #define AUBO_SCOPE_THREAD_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(ThreadNode);
00009
00010 class ARCS_ABI_EXPORT ThreadNode : public ProgramNode
00011 {
00012 public:
00013     ThreadNode(ThreadNode &f);
00014     ThreadNode(ThreadNode &&f);
00015     virtual ~ThreadNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     ThreadNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_THREAD_NODE_H
```

12.190 include/aubo_caps/domain/program/nodes/builtin/↵ timer_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
timer_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::TimerNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(TimerNode\)](#)

12.191 timer_node.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_TIMER_NODE_H
00002 #define AUBO_SCOPE_TIMER_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(TimerNode);
00009
00010         class ARCS_ABI_EXPORT TimerNode : public ProgramNode
00011         {
00012         public:
00013             TimerNode(TimerNode &f);
00014             TimerNode(TimerNode &&f);
00015             virtual ~TimerNode();
00016
00017         private:
00018             friend class DataSwitch;
00019             TimerNode();
00020             void *d_{ nullptr };
00021         };
00022
00023     } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_TIMER_NODE_H

```

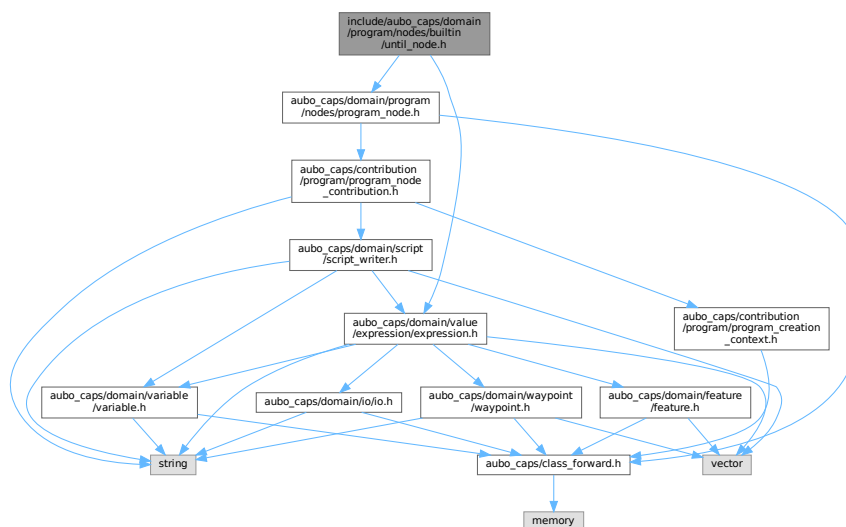
12.192 include/aubo_caps/domain/program/nodes/builtin/↵ until_node.h 文件参考

```

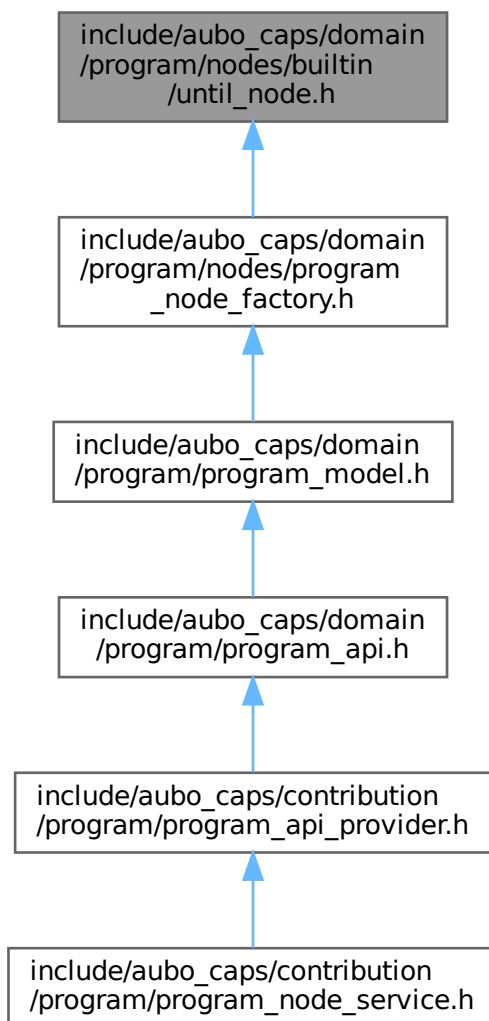
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/value/expression/expression.h>

```

until_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::UntilNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(UntilNode\)](#)

12.193 until_node.h

[浏览该文件的文档.](#)


```

00001 #ifndef AUBO_SCOPE_UNTIL_NODE_H
00002 #define AUBO_SCOPE_UNTIL_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(UntilNode);
00010
00011 /**
00012  * \chinese
00013  * <p>Until 节点是一种必须放在 Waypoint 节点或 Direction 节点下的程序节点类型。</p>
00014  * <p> 当调用 TreeNode#addChild 和 #removeChild 时, Until 节点具有特殊行为。
00015  * 默认情况下 Until 节点是叶子节点, 但可以向其添加动作。当添加动作时, 它们会转换
00016  * 为允许子节点的分支节点。当最后一个子节点被移除时, 它们又变回叶子节点。</p>
00017  * \endchinese
00018  * \english
00019  * <p>The Until node is a type of program node that must be under a Waypoint
00020  * node or Direction node.</p>
00021  * <p>It has special behavior when calling {@link
00022  * TreeNode#addChild(ProgramNode)} and {@link
00023  * TreeNode#removeChild(TreeNode)}. Until nodes by default are leaf nodes,
00024  * however they can have actions added to them. When that happens they are
00025  * transformed into branch nodes that allow child nodes. When the last child
00026  * is removed they become leaf nodes again. This is all automatically
00027  * handled when {@link TreeNode#addChild(ProgramNode)} and {@link
00028  * TreeNode#removeChild(TreeNode)} are called.</p>
00029  * \endenglish
00030  */
00031 class ARCS_ABI_EXPORT UntilNode : public ProgramNode
00032 {
00033 public:
00034     /**
00035      * The configuration type used to determine which type of configuration this
00036      * instance is.
00037      */
00038     enum ConfigType : short
00039     {
00040
00041         /**
00042          * <p>
00043          * An Until node where the until condition is true once the
00044          * evaluation of an expression is true.
00045          * </p>
00046          *
00047          * The config instance can be cast to {@link
00048          * ExpressionUntilNodeConfig}.
00049          */
00050         EXPRESSION,
00051         /**
00052          * <p>
00053          * An Until node where the until condition is true once an impact on
00054          * the Tool is detected.
00055          * </p>
00056          *
00057          * This functionality is not supported by the API.
00058          */
00059         TOOL_CONTACT,
00060         /**
00061          * <p>
00062          * An Until node where the until condition is true once a given I/O
00063          * reaches a certain value.
00064          * </p>
00065          *
00066          * The config can be cast to {@link IOInputUntilNodeConfig}.
00067          */
00068         IO_INPUT,
00069         /**
00070          * <p>
00071          * An Until node where the until condition is true once a given
00072          * distance is reached.
00073          * </p>
00074          *
00075          * The config can be cast to {@link DistanceUntilNodeConfig}.
00076          */
00077         DISTANCE,
00078         /**
00079          * <p>
00080          * An Until node where the until condition is true once its parent
00081          * Waypoint is reached.
00082          * </p>
00083          *
00084          * The config can be cast to {@link ReachedWaypointUntilNodeConfig}.
00085          */
00086         REACHED_WAYPOINT,
00087

```

```

00088
00089     /**
00090      * An undefined Until node with no type set yet.
00091      */
00092     NONE_CONFIG_TYPE
00093 };
00094
00095 /**
00096  * Compare operator types LESS_THAN and GREATER_THAN used when waiting for
00097  * an analog input to go lower or higher, respectively, than a threshold
00098  */
00099 enum CompareOperator : int
00100 {
00101     LESS_THAN,
00102     GREATER_THAN
00103 };
00104
00105 /**
00106  * Blend Parameters can be of different types
00107  * The parameter type is used to determine the explicit blend parameters
00108  * (can be used in place of instanceof operations).
00109  */
00110 enum BlendParameterType
00111 {
00112     /**
00113      * The robot arm will stop without performing a blend.
00114      */
00115     NO_BLEND,
00116     /**
00117      * <p>
00118      * Custom blend has been selected.
00119      * </p>
00120      *
00121      * The blend instance can be cast to {@link CustomBlendParameters}.
00122      */
00123     CUSTOM_BLEND
00124 };
00125
00126 enum CustomDecelerationParameterType
00127 {
00128     /**
00129      * This deceleration was either created specifying values for both
00130      * {@link AngularAcceleration} and {@link Acceleration} and/or is in use
00131      * in an Until node that is not yet in the program tree.
00132      */
00133     Unknown,
00134     /**
00135      * This deceleration is currently used under a MoveJ node.
00136      */
00137     Angular,
00138     /**
00139      * This deceleration is currently used under a MoveL or MoveP node.
00140      */
00141     Cartesian
00142 };
00143
00144 /**
00145  * The deceleration parameter type used to determine which type of
00146  * deceleration parameter this instance is.
00147  */
00148 enum DecelerationParameterType
00149 {
00150     /**
00151      * Shared deceleration has been selected. The acceleration values from
00152      * the parent Move node will be used as deceleration (absolute value).
00153      */
00154     SHARED_DECELERATION,
00155     /**
00156      * <p>
00157      * Custom deceleration has been selected.
00158      * </p>
00159      *
00160      * This instance can be cast to {@link CustomDecelerationParameters}.
00161      */
00162     CUSTOM_DECELERATION
00163 };
00164
00165 /**
00166  * The type of the input selection.
00167  */

```

```

00175     enum IOInputSelectionType
00176     {
00177
00178         /**
00179          * The node is not fully configured.
00180          */
00181         NONE_INPUT_SELECT,
00182
00183         /**
00184          * <p>
00185          * Digital input is selected.
00186          * </p>
00187          *
00188          * The config instance can be cast to {@link
00189          * DigitalInputUntilNodeConfig}.
00190          */
00191         DIGITAL_INPUT,
00192
00193         /**
00194          * <p>
00195          * Analog input is selected.
00196          * </p>
00197          *
00198          * The config instance can be cast to {@link
00199          * AnalogInputUntilNodeConfig}.
00200          */
00201         ANALOG_INPUT,
00202
00203         /**
00204          * <p>
00205          * A float input register is selected.
00206          * </p>
00207          *
00208          * The config instance can be cast to {@link
00209          * FloatRegisterInputUntilNodeConfig}.
00210          */
00211         FLOAT_REGISTER_INPUT
00212     };
00213
00214     UntilNode(UntilNode &f);
00215     UntilNode(UntilNode &&f);
00216     virtual ~UntilNode();
00217
00218     /**
00219      * This method returns the type of configuration. Cast this instance
00220      * appropriately to have access to specific getters.
00221      *
00222      * @return the type of this config.
00223      */
00224     ConfigType getConfigType();
00225     void setConfigType(ConfigType type);
00226
00227     BlendParameterType getBlendParameterType();
00228     void setBlendParameterType(BlendParameterType type);
00229
00230     /**
00231      * Creates blend parameters for an Until node. Can be used in a Distance
00232      * configuration type.
00233      *
00234      * @param suggestedBlend This blend value may be (automatically) corrected
00235      * depending on the value of the distance parameter the Distance
00236      * configuration.
00237      * @return the blend parameters for the Until node.
00238      */
00239     void setWaypointBlendParameter(double suggested_blend);
00240     double getWaypointBlendParameter();
00241
00242     DecelerationParameterType getDecelerationParameterType();
00243     void setDecelerationParameterType(DecelerationParameterType type);
00244
00245     /**
00246      * <p>
00247      * Creates deceleration parameter for an Until node. Can be used in an
00248      * Expression configuration type.
00249      * </p>
00250      *
00251      * An angular deceleration is used when the type of the parent Move node is
00252      * a MoveJ, whereas a Cartesian deceleration is used when the move type is a
00253      * MoveL or MoveP.
00254      *
00255      * @param deceleration the angular deceleration (the value for the Cartesian
00256      * deceleration will be the default value that AuboScope uses), not
00257      * <code>null</code>.
00258      * @param decelerationErrorHandler error handler for handling validation. If
00259      * using {@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00260      * nearest valid angular deceleration value.
00261      * @return the deceleration parameter for the Until node.

```

```

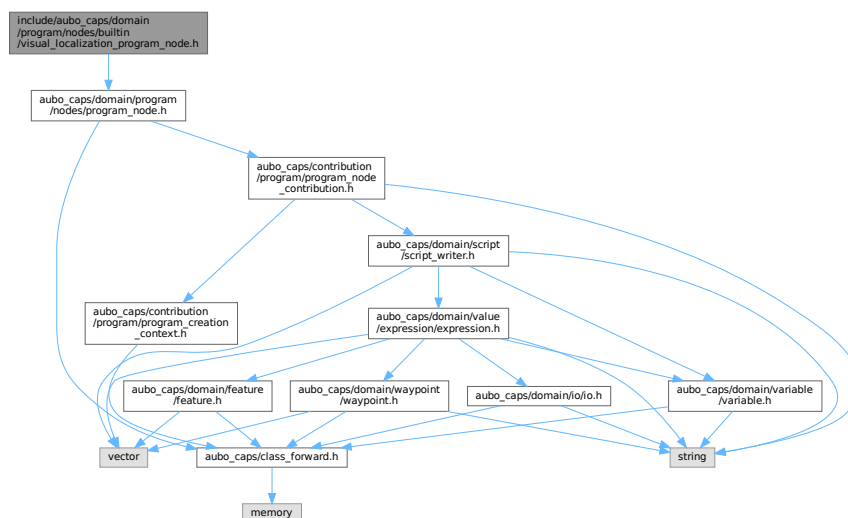
00262     */
00263 void setDecelerationParameters(double deceleration);
00264 double getDecelerationParameters();
00265
00266 /**
00267  *
00268  * @return the deceleration type. This can change depending on the Move node
00269  * type above the Until node.
00270  */
00271 CustomDecelerationParameterType getCustomDecelerationParameterType();
00272
00273 void setCustomDecelerationParameterType(
00274     CustomDecelerationParameterType type);
00275
00276 /**
00277  * The rate at which to decelerate when the until condition is met.
00278  *
00279  * @return the specified angular deceleration of the until node (absolute
00280  * value).
00281  */
00282 double getAngularDeceleration();
00283 void setAngularDeceleration(double dec);
00284
00285 /**
00286  * The rate at which to decelerate when the until condition is met.
00287  *
00288  * @return the specified Cartesian deceleration of the until node (absolute
00289  * value).
00290  */
00291 double getCartesianDeceleration();
00292 void setCartesianDeceleration(double dec);
00293
00294 /**
00295  * <p>
00296  * Creates an Expression configuration for an Until node with an expression
00297  * and default value for the deceleration parameter. The until condition for
00298  * this configuration is true once the evaluation of the expression is true.
00299  * </p>
00300  *
00301  * Note that this default value may change between AuboScope versions.
00302  *
00303  * @param expression the expression whose evaluation will determine whether
00304  * the until condition is met.
00305  * @return the Until node configuration.
00306  */
00307 void setExpressionConfig(ExpressionPtr expression);
00308 ExpressionPtr getExpressionConfig();
00309
00310 /**
00311  * The distance the robot will travel.
00312  *
00313  * @return The distance.
00314  */
00315 double getDistance();
00316 void setDistance(double dis);
00317 IOInputSelectionType getIOInputSelectionType();
00318 void setIOInputSelectionType(IOInputSelectionType type);
00319
00320 /**
00321  * Creates an I/O Input configuration for an Until node. The until condition
00322  * for this configuration is true once the signal value of a given digital
00323  * input goes HIGH or LOW.
00324  *
00325  * @param input the digital input whose signal value will determine whether
00326  * the until condition is met, not <code>null</code>.
00327  * @param valueToProceedUntil <code>true</code>, if the until condition is
00328  * met when the input signal value goes HIGH, <code>false</code> otherwise
00329  * (condition met when signal value is LOW).
00330  * @return the Until node configuration.
00331  * @throws IllegalArgumentException if <code>input</code> is an output (does
00332  * not support reading of values). See {@link IO#isInput()}.
00333  */
00334 void setDigitalInputValueToProceed(bool value_to_proceed_until);
00335 bool getDigitalInputValueToProceed();
00336
00337 /**
00338  * Creates an I/O Input configuration for an Until node. The until condition
00339  * for this configuration is true once the the signal value of a given
00340  * analog input goes past a specified electric Current threshold.
00341  *
00342  * @param input the analog input whose signal value will determine whether
00343  * the until condition is met, not <code>null</code>.
00344  * @param compareOperator the operator to use when comparing the signal
00345  * value of the selected analog input to the threshold value, not
00346  * <code>null</code>. Available options are "less than" ({@literal <}) or
00347  * "greater than"
00348  * ({@literal >}).

```

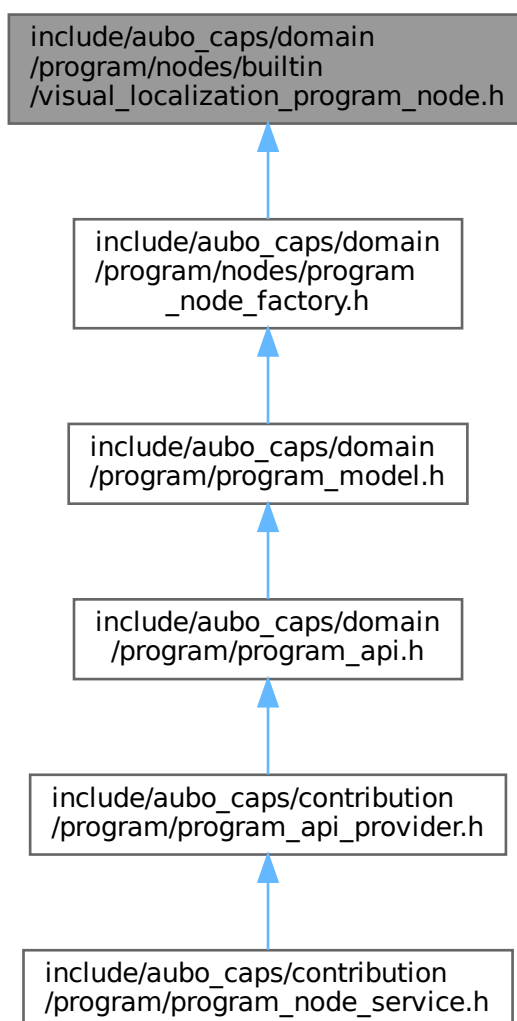
```
00349      * @param currentThreshold the electric Current threshold for the signal
00350      * value of the analog input. Execution will proceed until the signal value
00351      * of the selected analog input goes past this threshold.
00352      * @param errorHandler the error handler for validation errors. If using
00353      * {@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00354      * nearest valid current value.
00355      * @return the Until node configuration.
00356      * @throws IllegalArgumentException if <code>input</code> is an output (does
00357      * not support reading of values). See {@link IO#isInput()}.
00358      */
00359
00360      void setAnalogInputValueToProceed(double value);
00361      double getAnalogInputValueToProceed();
00362
00363      void setCompareOperator(CompareOperator compare_operator);
00364      CompareOperator getCompareOperator();
00365
00366      private:
00367          friend class DataSwitch;
00368          UntilNode();
00369          void *d_{ nullptr };
00370 };
00371
00372 } // namespace aubo_scope
00373 } // namespace arcs
00374 #endif // AUBO_SCOPE_UNTIL_NODE_H
```

12.194 include/aubo_caps/domain/program/nodes/builtin/↵ visual_localization_program_node.h 文件参考

#include <aubo_caps/domain/program/nodes/program_node.h>
visual_localization_program_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- `class arcs::aubo_scope::VisualLocalizationProgramNode`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (VisualLocalizationProgramNode)`

12.195 visual_localization_program_node.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_VISUAL_LOCALIZATION_PROGRAM_NODE_H
00002 #define AUBO_SCOPE_VISUAL_LOCALIZATION_PROGRAM_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(VisualLocalizationProgramNode);
00009
00010 class ARCS_ABI_EXPORT VisualLocalizationProgramNode : public ProgramNode
00011 {
00012 public:
00013     VisualLocalizationProgramNode(VisualLocalizationProgramNode &f);
00014     VisualLocalizationProgramNode(VisualLocalizationProgramNode &&f);
00015     virtual ~VisualLocalizationProgramNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     VisualLocalizationProgramNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025
00026 #endif // AUBO_SCOPE_VISUAL_LOCALIZATION_PROGRAM_NODE_H

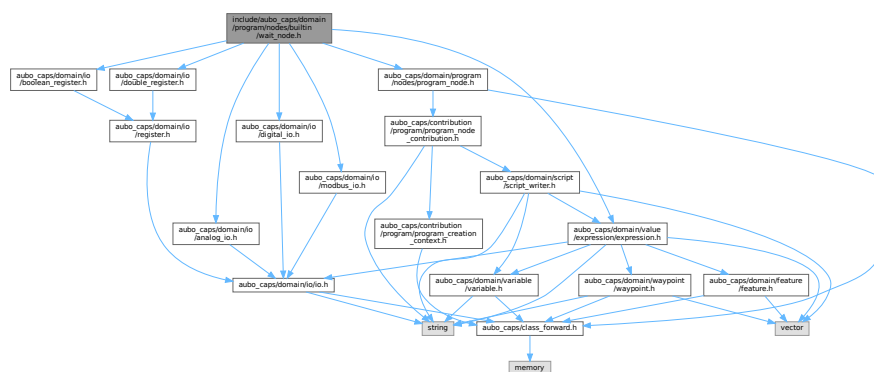
```

12.196 include/aubo_caps/domain/program/nodes/builtin/↵ wait_node.h 文件参考

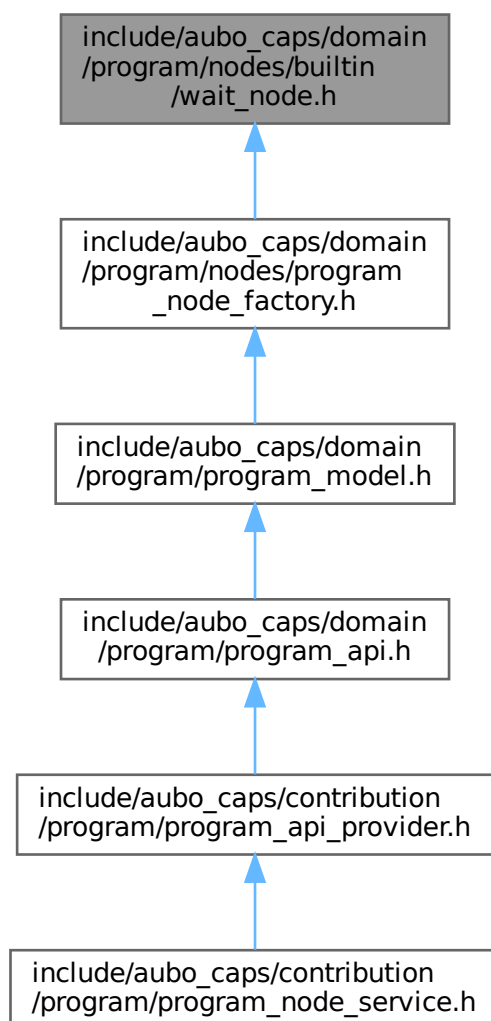
```

#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/io/analog_io.h>
#include <aubo_caps/domain/io/digital_io.h>
#include <aubo_caps/domain/io/modbus_io.h>
#include <aubo_caps/domain/io/boolean_register.h>
#include <aubo_caps/domain/io/double_register.h>
#include <aubo_caps/domain/value/expression/expression.h>
wait_node.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::WaitNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(WaitNode\)](#)

12.197 wait_node.h

[浏览该文件的文档.](#)


```

00001 #ifndef AUBO_SCOPE_WAIT_NODE_H
00002 #define AUBO_SCOPE_WAIT_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/io/analog_io.h>
00006 #include <aubo_caps/domain/io/digital_io.h>
00007 #include <aubo_caps/domain/io/modbus_io.h>
00008 #include <aubo_caps/domain/io/boolean_register.h>
00009 #include <aubo_caps/domain/io/double_register.h>
00010 #include <aubo_caps/domain/value/expression/expression.h>
00011
00012 namespace arcs {
00013 namespace aubo_scope {
00014 ARCS_CLASS_FORWARD(WaitNode);
00015
00016 class ARCS_ABI_EXPORT WaitNode : public ProgramNode
00017 {
00018 public:
00019     /**
00020      * The configuration type used to determine which type of configuration this
00021      * instance is.
00022      */
00023     enum ConfigType : int
00024     {
00025         NO_TYPE,
00026
00027         /**
00028          * <p>
00029          * Nothing to wait.
00030          * </p>
00031          *
00032          */
00033         NO_WAIT,
00034
00035         /**
00036          * <p>
00037          * Wait type is time.
00038          * </p>
00039          * The config instance can be cast to {@link TimeWaitNodeConfig}.
00040          */
00041         TIME,
00042
00043         /**
00044          * <p>
00045          * Wait type is digital_input.
00046          * </p>
00047          * The config instance can be cast to {@link
00048          * DigitalInputWaitNodeConfig}.
00049          */
00050         DIGITAL_INPUT,
00051
00052         /**
00053          * <p>
00054          * Wait type is analog_input_current.
00055          * </p>
00056          * The config instance can be cast to {@link
00057          * AnalogInputCurrentWaitNodeConfig}.
00058          */
00059         ANALOG_INPUT_CURRENT,
00060
00061         /**
00062          * <p>
00063          * Wait type is analog_input_voltage.
00064          * </p>
00065          * The config instance can be cast to {@link
00066          * AnalogInputVoltageWaitNodeConfig}.
00067          */
00068         ANALOG_INPUT_VOLTAGE,
00069
00070         /**
00071          * <p>
00072          * Wait type is float_register_input.
00073          * </p>
00074          * The config instance can be cast to {@link
00075          * FloatRegisterInputWaitNodeConfig}.
00076          */
00077         FLOAT_REGISTER_INPUT,
00078
00079         /**
00080          * <p>

```

```

00088         * Wait type is expression_input.
00089         * </p>
00090         *
00091         * The config instance can be cast to {@link
00092         * ExpressionInputWaitNodeConfig}.
00093         */
00094     EXPRESSION_INPUT
00095 };
00096
00097 /**
00098  * Compare operator types LESS_THAN and GREATER_THAN used when waiting for
00099  * an analog input to go lower or higher, respectively, than a threshold
00100  */
00101 enum CompareOperator : int
00102 {
00103     LESS_THAN,
00104     GREATER_THAN
00105 };
00106
00107 WaitNode(WaitNode &f);
00108 WaitNode(WaitNode &&f);
00109 virtual ~WaitNode();
00110
00111 /**
00112  * This method returns the type of configuration. Cast this instance
00113  * appropriately to have access to specific getters.
00114  *
00115  * @return the type of this config.
00116  */
00117 ConfigType getConfigType();
00118 void setConfigType(ConfigType type);
00119
00120 /**
00121  *
00122  * @return the time to wait for. second
00123  */
00124 double getWaitTime();
00125 void setWaitTime(double time);
00126
00127 /**
00128  * Depending on the type of input this is interpreted in different ways.
00129  * For digital inputs and MODBUS inputs, <code>true</code> means the input
00130  * to wait for must be HIGH, <code>false</code> means the input to wait for
00131  * must be LOW. For boolean registers the value is interpreted literally.
00132  *
00133  * @return the input value to wait for.
00134  */
00135 bool getBoolValueToWaitFor();
00136 void setBoolValueToWaitFor(bool value);
00137
00138 float getFloatValueToWaitFor();
00139 void setFloatValueToWaitFor(float value);
00140
00141 /**
00142  * Creates a configuration for waiting for a digital input to go high or
00143  * low.
00144  *
00145  * @param input the digital input on which to wait, not <code>null</code>.
00146  * @param value_to_wait_for <code>true</code>, if waiting for the input
00147  * signal value/level to go HIGH, <code>false</code> otherwise (waiting for
00148  * LOW signal value/level).
00149  * @return the configuration.
00150  * @throws IllegalArgumentException if <code>input</code> is an output (does
00151  * not support reading of values). See {@link IO#isInput()}.
00152  */
00153 void setDigitalInput(DigitalIoPtr input);
00154 DigitalIoPtr getDigitalInput();
00155
00156 /**
00157  * Creates a configuration for waiting for a digital MODBUS input to go high
00158  * or low.
00159  *
00160  * @param input the digital MODBUS input on which to wait, not
00161  * <code>null</code>.
00162  * @param value_to_wait_for <code>true</code>, if waiting for the input
00163  * signal value/level to go HIGH, <code>false</code> otherwise (waiting for
00164  * LOW signal value/level).
00165  * @return the configuration.
00166  * @throws IllegalArgumentException if <code>input</code> is an output (does
00167  * not support reading of values) or if <code>output</code> is not a digital
00168  * output. See {@link IO#isInput()} and {@link IO#getType()}.
00169  */
00170 void setModbusInput(ModbusIoPtr input);
00171 ModbusIoPtr getModbusInput();
00172
00173 /**
00174  * Creates a configuration for waiting for a boolean register to go

```

```

00175     * <code>true</code> or <code>>false</code>.
00176     *
00177     * @param input the boolean register input on which to wait, not
00178     * <code>null</code>.
00179     * @param value_to_wait_for <code>true</code>, if waiting for the register
00180     * to be <code>true</code>, <code>false</code> otherwise.
00181     * @return the configuration.
00182     * @throws IllegalArgumentException if <code>input</code> is an output (does
00183     * not support reading of values). See {@link IO#isInput()}.
00184     */
00185 void setBoolRegister(BoolRegisterPtr input);
00186 BoolRegisterPtr getBoolRegister();
00187
00188 void setCompareOperator(CompareOperator compareOperator);
00189 CompareOperator getCompareOperator();
00190
00191 /**
00192  * Creates a configuration for waiting for an analog input to go past an
00193  * electric current threshold.
00194  *
00195  * @param input on which to wait, not <code>null</code>.
00196  * @param compareOperator the operator to use when comparing input to value,
00197  * not <code>null</code>. Available options are "less than" ({@literal <})
00198  * or "greater than" ({@literal >}).
00199  * @param current the electric current threshold to wait for, not
00200  * <code>null</code>.
00201  * @param errorHandler the error handler for validation errors. If using
00202  * {@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00203  * nearest valid current value.
00204  * @return the configuration.
00205  * @throws IllegalArgumentException if <code>input</code> is an output (does
00206  * not support reading of values). See {@link IO#isInput()}.
00207  */
00208 void setAnalogInput(AnalogIoPtr input);
00209 AnalogIoPtr getAnalogInput();
00210
00211 /**
00212  * Creates a configuration for waiting for a register to go past a float
00213  * threshold.
00214  *
00215  * @param input the float register input on which to wait, not
00216  * <code>null</code>.
00217  * @param compareOperator the operator to use when comparing register to
00218  * value, not <code>null</code>. Available options are "less than"
00219  * ({@literal <}) or "greater than" ({@literal >}).
00220  * @param value_to_wait_for the float threshold to wait for.
00221  * @return the configuration.
00222  * @throws IllegalArgumentException if <code>input</code> is an output (does
00223  * not support reading of values). See {@link IO#isInput()}.
00224  */
00225 void setDoubleRegisterInput(DoubleRegisterPtr input);
00226 DoubleRegisterPtr getDoubleRegisterInput();
00227
00228 /**
00229  * Create a configuration for waiting for an expression to be evaluated to
00230  * true.
00231  *
00232  * @param expression the expression to evaluate, not <code>null</code>.
00233  * @return the configuration.
00234  */
00235 void setExpressionInputConfig(ExpressionPtr expression);
00236 ExpressionPtr getExpressionInputConfig();
00237
00238 private:
00239     friend class DataSwitch;
00240     WaitNode();
00241     void *d_{ nullptr };
00242 };
00243
00244 } // namespace aubo_scope
00245 } // namespace arcs
00246 #endif // AUBO_SCOPE_WAIT_NODE_H

```

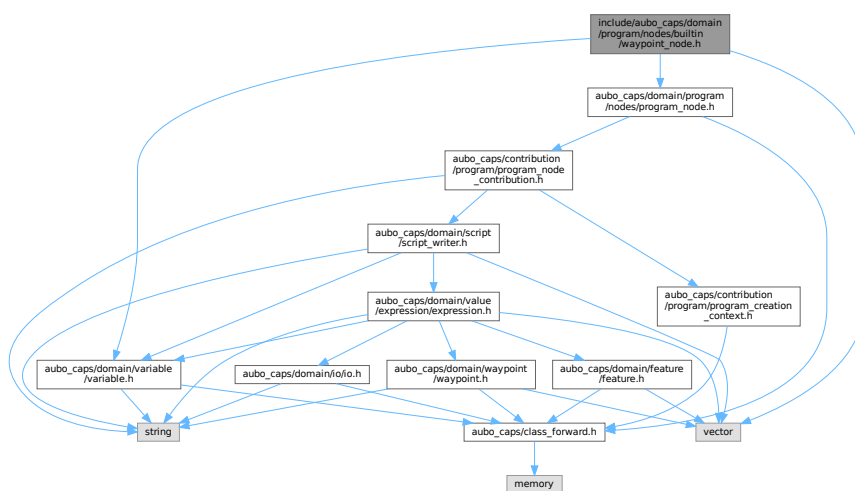
12.198 include/auco_caps/domain/program/nodes/builtin/↵ waypoint_node.h 文件参考

```

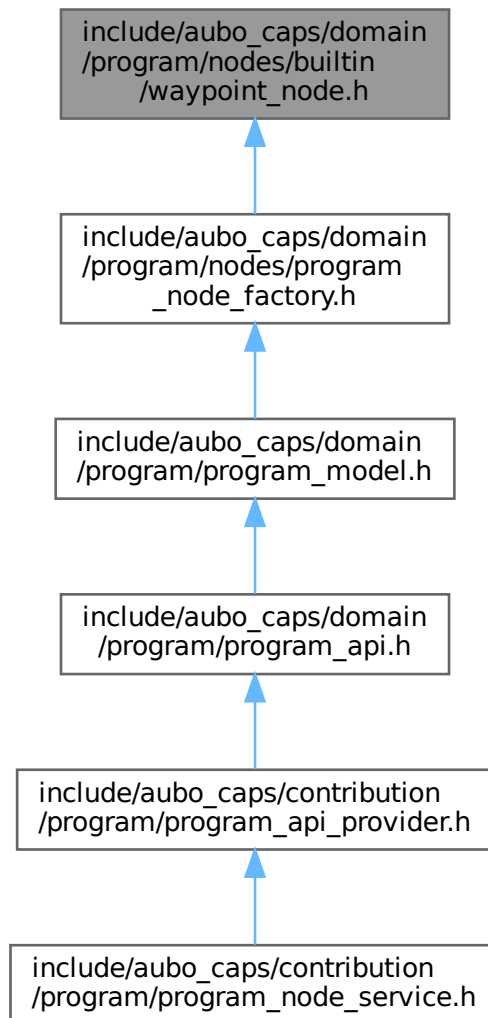
#include <vector>
#include <auco_caps/domain/program/nodes/program_node.h>
#include <auco_caps/domain/variable/variable.h>

```

waypoint_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo__scope::WaypointNode](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo__scope](#)

函数

- [arcs::aubo__scope::ARCS_CLASS_FORWARD \(WaypointNode\)](#)

12.199 waypoint_node.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_WAYPOINT_NODE_H
00002 #define AUBO_SCOPE_WAYPOINT_NODE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006 #include <aubo_caps/domain/variable/variable.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010
00011 ARCS_CLASS_FORWARD(WaypointNode);
00012
00013 class ARCS_ABI_EXPORT WaypointNode : public ProgramNode
00014 {
00015 public:
00016     /**
00017      * The configuration type used to determine which type of configuration this
00018      * instance is.
00019      */
00020     enum ConfigType : int
00021     {
00022         /**
00023          * <p>
00024          * Fixed position has been selected (fixed waypoint).
00025          * </p>
00026          *
00027          * The config instance can be cast to {@link
00028          * FixedPositionWaypointNodeConfig}.
00029          */
00030         FIXED_POSITION = 0,
00031
00032         /**
00033          * <p>
00034          * Variable position has been selected (variable waypoint).
00035          * </p>
00036          *
00037          * The config instance can be cast to {@link
00038          * VariablePositionWaypointNodeConfig}.
00039          */
00040         VARIABLE_POSITION,
00041
00042         /**
00043          * Relative position has been selected (relative waypoint).
00044          */
00045         RELATIVE_POSITION
00046     };
00047
00048     /**
00049      * The blend parameters type used to determine which type of blend
00050      * parameters this instance is.
00051      */
00052     enum BlendType : int
00053     {
00054         /**
00055          * No blend has been selected (the robot arm will stop at this point).
00056          */
00057         NO_BLEND = 0,
00058
00059         /**
00060          * Shared blend has been selected.
00061          */
00062         SHARED_BLEND,
00063
00064         /**
00065          * <p>>
00066          * Waypoint blend has been selected.
00067          * </p>
00068          *
00069          * The blend instance can be cast to {@link WaypointBlendParameters}.
00070          */
00071         WAYPOINT_BLEND
00072     };
00073
00074     /**
00075      * The blend parameters type used to determine which type of blend
00076      * parameters this instance is.
00077      */
00078     enum MotionType : int
00079     {
00080         /**
00081          * <p>
00082          * Time motion parameters has been selected.
00083          * </p>
00084          *
00085          * This instance can be cast to {@link TimeMotionParameters}.
00086          */
00087         TIME_MOTION,

```

```

00088
00089     /**
00090      * <p>
00091      * Joint motion parameters has been selected.
00092      * </p>
00093      *
00094      * This instance can be cast to {@link JointMotionParameters}.
00095      */
00096     JOINT_MOTION,
00097
00098     /**
00099      * <p>
00100      * Tool motion parameters has been selected.
00101      * </p>
00102      *
00103      * This instance can be cast to {@link ToolMotionParameters}.
00104      */
00105     TOOL_MOTION,
00106
00107     /**
00108      * Shared motion parameters has been selected.
00109      */
00110     SHARED_MOTION
00111 };
00112
00113 enum VariableType : int
00114 {
00115     /**
00116      * This variable type pose parameters has been selected.
00117      */
00118     VARIABLE_POSE,
00119
00120     /**
00121      * This variable type joint parameters has been selected.
00122      */
00123     VARIABLE_JOINT
00124 };
00125
00126 enum RelativePositionType : int
00127 {
00128     /**
00129      * The teaching location has been selected.
00130      */
00131     TEACHING,
00132
00133     /**
00134      * The input offset has been selected.
00135      */
00136     INPUT
00137 };
00138
00139 enum RelativeOffsetType : int
00140 {
00141     /**
00142      * Custom offset has been selected.
00143      */
00144     CUSTOM,
00145
00146     /**
00147      * Variable type has been selected.
00148      */
00149     VARIABLE
00150 };
00151
00152 enum class PositionDefinition : int
00153 {
00154     /**
00155      * The position has not been defined.
00156      */
00157     UNDEFINED,
00158
00159     /**
00160      * <p>
00161      * The position of the waypoint has been defined.
00162      * </p>
00163      *
00164      * Cast this instance to {@link FixedPositionDefinedWaypointNodeConfig}.
00165      */
00166     DEFINED
00167 };
00168
00169 enum VariableDefinition : int
00170 {
00171     /**
00172      * A variable has not been selected. This instance has no further
00173      * information.
00174      */

```

```

00175         UNDEFINED,
00176
00177         /**
00178          * <p>
00179          * A variable has been selected making this config defined.
00180          * </p>
00181          *
00182          * Cast this instance to {@link
00183          * VariablePositionDefinedWaypointNodeConfig}.
00184          */
00185         DEFINED
00186     };
00187
00188     WaypointNode(WaypointNode &f);
00189     WaypointNode(WaypointNode &&f);
00190     virtual ~WaypointNode();
00191
00192     /**
00193     * This method returns the type of configuration. Cast this instance
00194     * appropriately to have access to specific getters.
00195     *
00196     * @return the type of this config.
00197     */
00198     ConfigType getConfigType();
00199
00200     /**
00201     * Get blend type of the waypoint
00202     * @return the blend type.
00203     */
00204     BlendType getBlendType();
00205
00206     MotionType getMotionType();
00207
00208     /**
00209     * Cast this instance appropriately to have access to specific getters.
00210     *
00211     * @return the position definition.
00212     */
00213     PositionDefinition getPositionDefinition();
00214
00215     /**
00216     * Cast this instance appropriately to have access to specific getters.
00217     *
00218     * @return the variable definition.
00219     */
00220     VariableDefinition getVariableDefinition();
00221
00222     /**
00223     * <p>
00224     * Get the blend to be used for the movement.
00225     * </p>
00226     *
00227     * Note that, if the configuration has not been applied to a waypoint node
00228     * in the program tree, the blend value returned is a "suggested" value,
00229     * since the blend may be (automatically) corrected depending on surrounding
00230     * waypoints in the program tree (when it is applied to a waypoint node).
00231     *
00232     * @return the blend for the waypoint.
00233     */
00234     double getBlend();
00235
00236     /**
00237     * @return the joint speed to be achieved.
00238     */
00239     double getJointSpeed();
00240
00241     /**
00242     * @return the joint acceleration to use for the movement.
00243     */
00244     double getJointAcceleration();
00245
00246     /**
00247     * @return the time for the waypoint movement.
00248     */
00249     double getTime();
00250
00251     /**
00252     * @return the tool speed to be achieved.
00253     */
00254     double getToolSpeed();
00255
00256     /**
00257     * @return the tool acceleration to use for the movement
00258     */
00259     double getToolAcceleration();
00260
00261     /**

```



```

00262     * @return the waypoint position and orientation.
00263     */
00264     std::vector<double> getPose();
00265
00266     /**
00267     * @return the joint positions corresponding to the pose of the robot (when
00268     * taking the TCP offset into account)
00269     */
00270     std::vector<double> getJointPositions();
00271
00272     /**
00273     * @return the pose for the TCP offset (used when defining the robot
00274     * position)
00275     */
00276     std::vector<double> getTcpOffset();
00277
00278     /**
00279     * @return the variable for the waypoint.
00280     */
00281     VariablePtr getVariable();
00282
00283     /**
00284     * <p>
00285     * Creates shared motion parameters for a waypoint.
00286     * </p>
00287     *
00288     * Can be used for waypoints under a MoveJ, MoveL or a MoveP.
00289     *
00290     * @return the motion parameters for the waypoint.
00291     */
00292     void setSharedMotionParameters();
00293
00294     /**
00295     * <p></p>
00296     * Creates time motion parameters for a waypoint.
00297     * </p>
00298     *
00299     * Can be used for waypoints under a MoveJ or a MoveL.
00300     *
00301     * @param duration the time the movement should take, not <code>null</code>.
00302     * @param errorHandler error handler for handling validation. If using
00303     * {@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00304     * nearest valid time value.
00305     * @return the time motion parameters for the waypoint.
00306     */
00307     void setTimeMotionParameters(unsigned int duration);
00308
00309     /**
00310     * <p>
00311     * Creates joint motion parameters for a waypoint.
00312     * </p>
00313     *
00314     * Can be used for waypoints under a MoveJ.
00315     *
00316     * @param jointSpeed the joint speed to be achieved, not <code>null</code>.
00317     * @param jointSpeedErrorHandler error handler for handling validation. If
00318     * using {@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00319     * nearest valid joint speed value.
00320     * @param joint_acceleration the joint acceleration to be used for the
00321     * movement, not <code>null</code>.
00322     * @param joint_accelerationErrorHandler error handler for handling
00323     * validation. If using {@link ErrorHandler#AUTO_CORRECT} this will clamp
00324     * the value to the nearest valid joint acceleration value.
00325     * @return the joint motion parameters for the waypoint.
00326     */
00327     void setJointMotionParameters(double joint_speed,
00328                                   double joint_acceleration);
00329
00330     /**
00331     * <p>
00332     * Creates tool motion parameters for a waypoint.
00333     * </p>
00334     *
00335     * Can be used for waypoints under a MoveL or a MoveP.
00336     *
00337     * @param tool_speed the tool speed to be achieved, not <code>null</code>.
00338     * @param tool_speedErrorHandler error handler for handling validation. If
00339     * using {@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00340     * nearest valid tool speed value.
00341     * @param tool_acceleration the tool acceleration to be used for the
00342     * movement, not <code>null</code>.
00343     * @param tool_accelerationErrorHandler error handler for handling
00344     * validation. If using {@link ErrorHandler#AUTO_CORRECT} this will clamp
00345     * the value to the nearest valid tool acceleration value.
00346     * @return the tool motion parameters for the waypoint.
00347     */
00348     void setToolMotionParameters(double tool_speed, double tool_acceleration);

```

```

00349
00350 /**
00351  * <p>
00352  * Creates a fixed position configuration with default values for a
00353  * waypoint.
00354  * </p>
00355  *
00356  * Note that these default values may change between AuboScope versions.
00357  *
00358  * @return the waypoint configuration.
00359  */
00360 void setFixedPositionConfig();
00361
00362 /**
00363  * <p>
00364  * Creates a fixed position configuration for a waypoint with a defined
00365  * pose.
00366  * </p>
00367  *
00368  * @param pose_including_tcp the position and orientation for the waypoint,
00369  * not <code>null</code>.
00370  * @param qNear the joint positions that will be used for the inverse
00371  * kinematics calculations as a robot configuration near the waypoint pose,
00372  * not <code>null</code>.
00373  * @param blendParameters the blend parameters for the waypoint, not
00374  * <code>null</code>.
00375  * @param waypoint_motion_parameters the motion parameters for the waypoint,
00376  * not <code>null</code>.
00377  * @return the waypoint configuration.
00378  */
00379 void setFixedPositionConfig(const std::vector<double> &pose_including_tcp,
00380                             const std::vector<double> &qNear);
00381
00382 /**
00383  * <p>
00384  * Creates a variable position configuration with default values for a
00385  * waypoint
00386  * </p>
00387  *
00388  * Note that these default values may change between AuboScope versions.
00389  *
00390  * @return the waypoint configuration.
00391  */
00392 void setVariablePositionConfig();
00393
00394 /**
00395  * Creates a variable position configuration for a waypoint with a defined
00396  * variable
00397  *
00398  * @param variable the variable position and orientation for the waypoint,
00399  * not <code>null</code>.
00400  * @param blendParameters the blend parameters for the waypoint, not
00401  * <code>null</code>.
00402  * @param waypoint_motion_parameters the motion parameters for the waypoint,
00403  * not <code>null</code>.
00404  * @return the waypoint configuration.
00405  */
00406 void setVariablePositionConfig(VariablePtr variable);
00407
00408 private:
00409     friend class DataSwitch;
00410     WaypointNode();
00411     void *d_{ nullptr };
00412 };
00413
00414 } // namespace aubo_scope
00415 } // namespace arcs
00416 #endif // AUBO_SCOPE_WAYPOINT_NODE_H

```

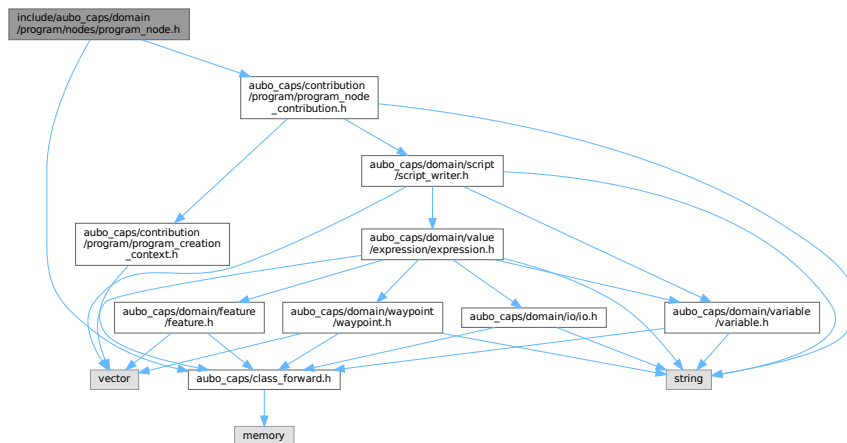
12.200 include/aubo_caps/domain/program/nodes/program_node.h 文件参考

```

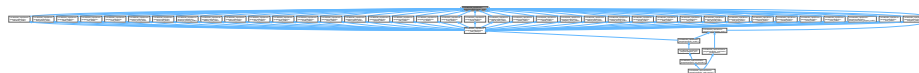
#include <aubo_caps/class_forward.h>
#include <aubo_caps/contribution/program/program_node_contribution.h>

```

program_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- `class arcs::aubo_scope::ProgramNode`
程序节点此接口表示 *AuboScope* 中的程序节点。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ProgramNode)`

12.201 program_node.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_PROGRAM_NODE_H
00002 #define AUBO_SCOPE_PROGRAM_NODE_H
00003
00004 #include <aubo_caps/class_forward.h>
00005 #include <aubo_caps/contribution/program/program_node_contribution.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(ProgramNode);
00010
00011 /**
00012  * \chinese
00013  * 程序节点
00014  * 此接口表示 AuboScope 中的程序节点。
00015  *
00016  * 程序节点可以通过 {@link ProgramNodeFactory} 创建。
00017  * \endchinese
00018  * \english
00019  * ProgramNode
00020  * This interface represents AuboScope program nodes.
00021  *

```

```

00022 * The program nodes can be created using {@link ProgramNodeFactory}.
00023 * \endenglish
00024 */
00025 class ARCS_ABI_EXPORT ProgramNode
00026 {
00027 public:
00028     ProgramNode(ProgramNode &f);
00029     ProgramNode(ProgramNode &&f);
00030     virtual ~ProgramNode();
00031
00032 /**
00033  * \chinese
00034  * 获取程序节点贡献。不推荐用户直接调用 ProgramNodeContribution::generateScript,
00035  * 应使用 ScriptWriter::writeChildren。
00036  * @return 程序节点贡献
00037  * \endchinese
00038  * \english
00039  * Get the program node contribution. Not recommended to call
00040  * ProgramNodeContribution::generateScript directly, use
00041  * ScriptWriter::writeChildren instead.
00042  * @return the ProgramNodeContribution
00043  * \endenglish
00044  */
00045 [[deprecated]] ProgramNodeContributionPtr getProgramNodeContribution();
00046
00047 protected:
00048     ProgramNode();
00049
00050 private:
00051     friend class DataSwitch;
00052     void *d_{ nullptr };
00053 };
00054
00055 } // namespace aubo_scope
00056 } // namespace arcs
00057
00058 #endif // AUBO_SCOPE_PROGRAM_NODE_H

```

12.202 include/auco_caps/domain/program/nodes/program_node_factory.h 文件参考

```

#include <auco_caps/domain/program/nodes/builtin/assignment_node.h>
#include <auco_caps/domain/program/nodes/builtin/circle_move_node.h>
#include <auco_caps/domain/program/nodes/builtin/comment_node.h>
#include <auco_caps/domain/program/nodes/builtin/direction_node.h>
#include <auco_caps/domain/program/nodes/builtin/folder_node.h>
#include <auco_caps/domain/program/nodes/builtin/force_node.h>
#include <auco_caps/domain/program/nodes/builtin/force_option_node.h>
#include <auco_caps/domain/program/nodes/builtin/halt_node.h>
#include <auco_caps/domain/program/nodes/builtin/program_home_node.h>
#include <auco_caps/domain/program/nodes/builtin/if_node.h>
#include <auco_caps/domain/program/nodes/builtin/else_if_node.h>
#include <auco_caps/domain/program/nodes/builtin/else_node.h>
#include <auco_caps/domain/program/nodes/builtin/loop_node.h>
#include <auco_caps/domain/program/nodes/builtin/move_node.h>
#include <auco_caps/domain/program/nodes/builtin/pallet_node.h>
#include <auco_caps/domain/program/nodes/builtin/popup_node.h>
#include <auco_caps/domain/program/nodes/builtin/timer_node.h>
#include <auco_caps/domain/program/nodes/builtin/screw_driving_node.h>
#include <auco_caps/domain/program/nodes/builtin/script_node.h>
#include <auco_caps/domain/program/nodes/builtin/seek_node.h>
#include <auco_caps/domain/program/nodes/builtin/set_node.h>
#include <auco_caps/domain/program/nodes/builtin/set_payload_node.h>
#include <auco_caps/domain/program/nodes/builtin/until_node.h>
#include <auco_caps/domain/program/nodes/builtin/wait_node.h>
#include <auco_caps/domain/program/nodes/builtin/waypoint_node.h>
#include <auco_caps/domain/program/nodes/builtin/switch_node.h>
#include <auco_caps/domain/program/nodes/builtin/case_node.h>
#include <auco_caps/domain/program/nodes/builtin/default_case_node.h>

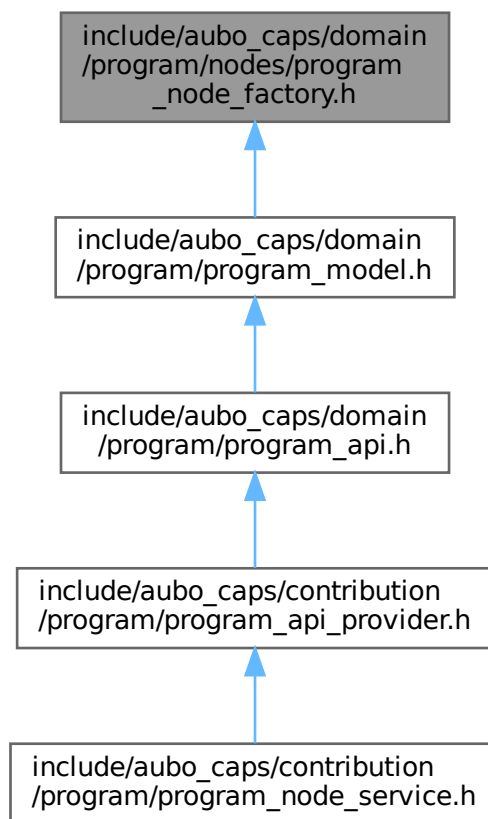
```

```
#include <aubo_caps/domain/program/nodes/builtin/break_node.h>
#include <aubo_caps/domain/program/nodes/builtin/sub_prog_node.h>
#include <aubo_caps/domain/program/nodes/builtin/call_node.h>
#include <aubo_caps/domain/program/nodes/builtin/thread_node.h>
#include <aubo_caps/domain/program/nodes/builtin/offline_track_node.h>
#include <aubo_caps/domain/program/nodes/builtin/record_track_node.h>
#include <aubo_caps/domain/program/nodes/builtin/gripper_node.h>
#include <aubo_caps/domain/program/nodes/builtin/assembly_node.h>
#include <aubo_caps/domain/program/nodes/builtin/visual_localization_program_node.h>
#include <aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h>
```

program_node_factory.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ProgramNodeFactory](#)

程序节点工厂此接口提供创建不同类型程序节点的方法。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`ProgramNodeFactory`)
- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`ProgramNodeService`)

12.203 program_node_factory.h

浏览该文件的文档.

```
00001 #ifndef AUBO_SCOPE_PROGRAM_NODE_FACTORY_H
00002 #define AUBO_SCOPE_PROGRAM_NODE_FACTORY_H
00003
00004 #include <aubo_caps/domain/program/nodes/builtin/assignment_node.h>
00005 #include <aubo_caps/domain/program/nodes/builtin/circle_move_node.h>
00006 #include <aubo_caps/domain/program/nodes/builtin/comment_node.h>
00007 #include <aubo_caps/domain/program/nodes/builtin/direction_node.h>
00008 #include <aubo_caps/domain/program/nodes/builtin/folder_node.h>
00009 #include <aubo_caps/domain/program/nodes/builtin/force_node.h>
00010 #include <aubo_caps/domain/program/nodes/builtin/force_option_node.h>
00011 #include <aubo_caps/domain/program/nodes/builtin/halt_node.h>
00012 #include <aubo_caps/domain/program/nodes/builtin/program_home_node.h>
00013 #include <aubo_caps/domain/program/nodes/builtin/if_node.h>
00014 #include <aubo_caps/domain/program/nodes/builtin/else_if_node.h>
00015 #include <aubo_caps/domain/program/nodes/builtin/else_node.h>
00016 #include <aubo_caps/domain/program/nodes/builtin/loop_node.h>
00017 #include <aubo_caps/domain/program/nodes/builtin/move_node.h>
00018 #include <aubo_caps/domain/program/nodes/builtin/pallet_node.h>
00019 #include <aubo_caps/domain/program/nodes/builtin/popup_node.h>
00020 #include <aubo_caps/domain/program/nodes/builtin/timer_node.h>
00021 #include <aubo_caps/domain/program/nodes/builtin/screw_driving_node.h>
00022 #include <aubo_caps/domain/program/nodes/builtin/script_node.h>
00023 #include <aubo_caps/domain/program/nodes/builtin/seek_node.h>
00024 #include <aubo_caps/domain/program/nodes/builtin/set_node.h>
00025 #include <aubo_caps/domain/program/nodes/builtin/set_payload_node.h>
00026 #include <aubo_caps/domain/program/nodes/builtin/until_node.h>
00027 #include <aubo_caps/domain/program/nodes/builtin/wait_node.h>
00028 #include <aubo_caps/domain/program/nodes/builtin/waypoint_node.h>
00029 #include <aubo_caps/domain/program/nodes/builtin/switch_node.h>
00030 #include <aubo_caps/domain/program/nodes/builtin/case_node.h>
00031 #include <aubo_caps/domain/program/nodes/builtin/default_case_node.h>
00032 #include <aubo_caps/domain/program/nodes/builtin/break_node.h>
00033 #include <aubo_caps/domain/program/nodes/builtin/sub_prog_node.h>
00034 #include <aubo_caps/domain/program/nodes/builtin/call_node.h>
00035 #include <aubo_caps/domain/program/nodes/builtin/thread_node.h>
00036 #include <aubo_caps/domain/program/nodes/builtin/offline_track_node.h>
00037 #include <aubo_caps/domain/program/nodes/builtin/record_track_node.h>
00038 #include <aubo_caps/domain/program/nodes/builtin/gripper_node.h>
00039 #include <aubo_caps/domain/program/nodes/builtin/assembly_node.h>
00040 #include <aubo_caps/domain/program/nodes/builtin/visual_localization_program_node.h>
00041 #include <aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h>
00042 #include <aubo_caps/domain/program/nodes/builtin/gripper_node.h>
00043
00044 namespace arcs {
00045     namespace aubo_scope {
00046         ARCS_CLASS_FORWARD(ProgramNodeFactory);
00047         ARCS_CLASS_FORWARD(ProgramNodeService);
00048     }
00049     /**
00050      * \chinese
00051      * 程序节点工厂
00052      * 此接口提供创建不同类型程序节点的方法。
00053      *
00054      * 可以创建内置程序节点（由 AUBO Robots 提供）和 AuboCap 程序节点。大多数方法
00055      * 将创建与终端用户手动添加节点时具有相同默认配置的节点。
00056      *
00057      * 请注意，此默认配置可能会因 AuboScope 版本而异。
00058      * \endchinese
00059      * \english
00060      * ProgramNodeFactory
00061      * This interfaces supplies methods to create different types of program nodes.
00062      *
00063      * Both built-in program nodes (provided by AUBO Robots) and AuboCap
00064      * program nodes can be created. Most methods will create a node with the same
00065      * default configuration as if the end user added the node manually.
00066      *
00067      * Note that this default configuration may change between AuboScope versions.
00068      * \endenglish
00069     */
00070 }
```

```

00069  */
00070  class ARCS_ABI_EXPORT ProgramNodeFactory
00071  {
00072  public:
00073      ProgramNodeFactory(ProgramNodeFactory &f);
00074      ProgramNodeFactory(ProgramNodeFactory &&f);
00075      virtual ~ProgramNodeFactory();
00076
00077      /**
00078       * \chinese
00079       * 创建一个 AuboCap 程序节点, 该节点是 ProgramNodeContribution 的实例。
00080       *
00081       * @param typeid_service 创建程序节点贡献的服务
00082       * @return 由 ProgramNodeService 创建的表示 ProgramNodeContribution 的
00083       * ProgramNode。如果提供 ProgramNodeService 的 AuboCap 未安装, 则返回
00084       * <code>null</code>。
00085       * @throws IllegalArgumentException 如果参数 ProgramNodeService 为
00086       * <code>null</code> 或未实现 ProgramNodeService
00087       * \endchinese
00088       * \english
00089       * This method creates a AuboCap program node which is an instance of a
00090       * {@link contribution/ProgramNodeContribution}.
00091       *
00092       * @param typeid_service is the service creating the program node
00093       * contribution
00094       * @return the {@link ProgramNode} which represents the {@link
00095       * contribution/ProgramNodeContribution} created by {@link
00096       * ProgramNodeService}. Returns <code>null</code> if AuboCap providing
00097       * ProgramNodeService is not installed.
00098       * @throws IllegalArgumentException if the argument ProgramNodeService
00099       * is <code>null</code> or ProgramNodeService does not implement {@link
00100       * ProgramNodeService}.
00101       * \endenglish
00102       */
00103      ProgramNodePtr createUserProgramNode(const char *typeid_service);
00104
00105      /**
00106       * \chinese
00107       * 创建一个默认的移动节点, 其下包含一个默认路点节点。
00108       * @return 新的移动节点
00109       * \endchinese
00110       * \english
00111       * Creates a default Move node with one default Waypoint node under it.
00112       * @return new Move Node.
00113       * \endenglish
00114       */
00115      MoveNodePtr createMoveNode();
00116
00117      /**
00118       * \chinese
00119       * <p> 创建一个默认的移动节点, 其下不包含路点节点 (例如用于插入命名路点)。</p>
00120       * 注意: 移动节点必须至少包含一个路点节点才有效。
00121       * @return 新的移动节点
00122       * \endchinese
00123       * \english
00124       * <p>Creates a default Move node without a Waypoint node under it (e.g. to be
00125       * used for inserting named waypoints).</p>
00126       * Note that at least one Waypoint node must be under it for a Move node to
00127       * be valid.
00128       * @return new Move Node.
00129       * \endenglish
00130       */
00131      MoveNodePtr createMoveNodeNoTemplate();
00132
00133      /**
00134       * \chinese
00135       * 创建具有默认路点名称的路点节点 (对应终端用户手动添加新节点时生成的名称)。
00136       * @return 新的路点节点
00137       * \endchinese
00138       * \english
00139       * Creates a Waypoint node with default waypoint name (corresponds to the
00140       * name generated if the end user added a new node manually).
00141       * @return new Waypoint Node.
00142       * \endenglish
00143       */
00144      WaypointNodePtr createWaypointNode();
00145
00146      /**
00147       * \chinese
00148       * <p> 创建具有建议名称的路点节点。</p>
00149       * <p> 如果建议名称已被注册, 最终名称可能会不同。路点名称在插入程序树时注册。</p>
00150       * 变量路点将使用变量名称而不是建议名称。
00151       *
00152       * @param suggested_name 建议的路点名称。有效名称必须匹配正则表达式
00153       * [a-zA-Z][a-zA-Z0-9_]{0,14}
00154       * @return 新的路点节点
00155       * @throws IllegalWaypointNameException 如果名称为非法格式

```



```

00156     * \endchinese
00157     * \english
00158     * <p>Creates a Waypoint node with a suggested name.</p>
00159     * <p>The final name can be different if the suggested name is already
00160     * registered. The waypoint name is registered when it is inserted into the
00161     * program tree.</p>
00162     * Variable waypoints will use the name of the variable instead of the
00163     * suggested name.
00164     *
00165     * @param suggested_name the suggested name of the waypoint. Valid names
00166     * must match regex [a-zA-Z][a-zA-Z0-9_]{0,14} for a total of 15 characters.
00167     * @return new Waypoint Node.
00168     * @throws IllegalWaypointNameException In case of an illegal name.
00169     * \endenglish
00170     */
00171     WaypointNodePtr createWaypointNode(const std::string &suggested_name);
00172
00173     /**
00174     * \chinese
00175     * 创建一个未指定 until 类型的 Until 节点。
00176     * @return 新的 Until 节点
00177     * \endchinese
00178     * \english
00179     * Creates an Until Node with no until type specified.
00180     * @return a new Until Node.
00181     * \endenglish
00182     */
00183     UntilNodePtr createUntilNode();
00184     DirectionNodePtr createDirectionNode();
00185     WaitNodePtr createWaitNode();
00186     SetNodePtr createSetNode();
00187     SetPayloadNodePtr createSetPayloadNode();
00188     PopupNodePtr createPopupNode();
00189     HaltNodePtr createHaltNode();
00190     CommentNodePtr createCommentNode();
00191     FolderNodePtr createFolderNode();
00192     LoopNodePtr createLoopNode();
00193     AssignmentNodePtr createAssignmentNode();
00194     IfNodePtr createIfNode();
00195     ElseIfNodePtr createElseIfNode();
00196     ElseNodePtr createElseNode();
00197     CircleMoveNodePtr createCircleMoveNode();
00198     PalletNodePtr createPalletNode();
00199     SeekNodePtr createSeekNode();
00200     ForceNodePtr createForceNode();
00201     ForceOptionNodePtr createForceOptionNode();
00202     AssemblyNodePtr createAssemblyNode();
00203     ScrewdrivingNodePtr createScrewdrivingNode();
00204     ScriptNodePtr createScriptNode();
00205     TimerNodePtr createTimerNode();
00206     SwitchNodePtr createSwitchNode();
00207     CaseNodePtr createCaseNode();
00208     DefaultCaseNodePtr createDefaultCaseNode();
00209     BreakNodePtr createBreakNode();
00210     SubProgNodePtr createSubProgNode();
00211     CallNodePtr createCallNode();
00212     OfflineTrackNodePtr createOfflineTrackNode();
00213     RecordTrackNodePtr createRecordTrackNode();
00214     ThreadNodePtr createThreadNode();
00215     ProgramHomeNodePtr createProgramHomeNode();
00216     VisualLocalizationProgramNodePtr createVisualLocalizationProgramNode();
00217     ConveyorTrackingNodePtr createConveyorTrackingNode();
00218
00219     /**
00220     * \chinese
00221     * <p> 创建新的夹爪程序节点。此节点类型可用于编程所选夹爪设备的夹取和释放动作。</p>
00222     * 程序节点将使用与终端用户在 AuboScope 中手动添加节点时相同的默认配置创建，
00223     * 即未选择夹爪动作的未定义配置。
00224     *
00225     * @param gripperDevice 要为其创建夹爪节点的夹爪设备，不能为 <code>null</code>。
00226     * 创建的夹爪节点将为此设备编程夹爪动作。
00227     * @return 新的夹爪程序节点（可使用 TreeNode 接口插入到程序树中）
00228     * @throws UnresolvedDeviceException 当指定的 gripperDevice 设备不可解析时
00229     * \endchinese
00230     * \english
00231     * <p>
00232     * This method creates a new Gripper program node. This node type can be
00233     * used for programming grip and release actions with a selected gripper
00234     * device.
00235     * </p>
00236     * The program node will be created with the same default configuration as
00237     * if the end user added the node manually in AuboScope, i.e. an undefined
00238     * configuration where the gripper action is unselected.
00239     *
00240     * @param gripperDevice the gripper device to create a Gripper node for, not
00241     * <code>null</code>. The created Gripper node will program gripper actions
00242     * for this device.

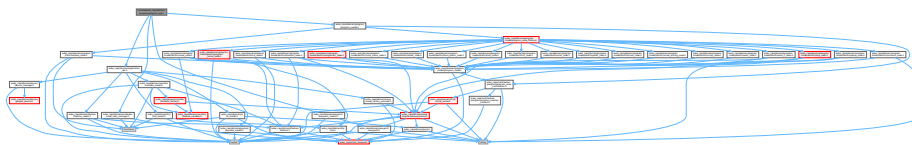
```



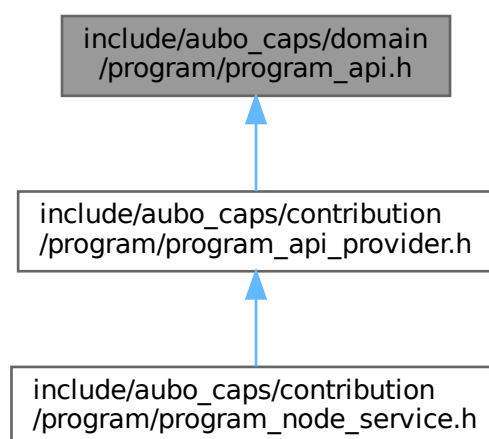
```
00243     * @return a new Gripper program node (which can be inserted into the
00244     * program tree using the {@link TreeNode} interface).
00245     * @throws UnresolvedDeviceException when the specified
00246     * <code>gripperDevice</code> device is unresolved. See
00247     *                                     {@link GripperDevice#isResolvable()}.
00248     * \endenglish
00249     */
00250     GripperNodePtr createGripperNode(GripperDevicePtr gripperDevice);
00251
00252 private:
00253     friend class DataSwitch;
00254     ProgramNodeFactory();
00255     void *d_{ nullptr };
00256 };
00257
00258 } // namespace aubo_scope
00259 } // namespace arcs
00260
00261 #endif
```

12.204 include/aubo_caps/domain/program/program_api.h 文件参考

```
#include <aubo_caps/domain/application_api.h>
#include <aubo_caps/domain/program/program_model.h>
#include <aubo_caps/domain/undoredo/undo_redo_manager.h>
#include <aubo_caps/domain/waypoint/waypoint_model.h>
program_api.h 的引用 (Include) 关系图:
```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::ProgramApi](#)

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`ProgramApi`)

12.205 program_api.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_PROGRAM_API_H
00002 #define AUBO_SCOPE_PROGRAM_API_H
00003
00004 #include <aubo_caps/domain/application_api.h>
00005 #include <aubo_caps/domain/program/program_model.h>
00006 #include <aubo_caps/domain/undoredo/undo_redo_manager.h>
00007 #include <aubo_caps/domain/waypoint/waypoint_model.h>
00008
00009 namespace arcs {
00010     namespace aubo_scope {
00011         ARCS_CLASS_FORWARD(ProgramApi);
00012
00013         /**
00014          * \chinese
00015          * @defgroup ProgramApi ProgramApi(程序)
00016          * @ingroup ProgramApi
00017          * 程序接口
00018          * 提供与程序相关的功能和服务访问。这包括 AuboScope 内部已有的功能，以及在
00019          * AuboScope 中创建要使用的额外模型元素。
00020          * \endchinese
00021          * \english
00022          * @defgroup ProgramApi ProgramApi
00023          * @ingroup ProgramApi
00024          * ProgramApi
00025          * Provides access to functionality and services related to programs. This
00026          * includes what is available from within AuboScope, as well as creating
00027          * additional model elements to be used within AuboScope.
00028          * \endenglish
00029          */
00030         class ARCS_ABI_EXPORT ProgramApi : public ApplicationApi
00031         {
00032         public:
00033             ProgramApi(ProgramApi &f);
00034             ProgramApi(ProgramApi &&f);
00035             virtual ~ProgramApi();
00036
00037             /**
00038              * @ingroup ProgramApi
00039              * @ref ProgramModel
00040              * \chinese
00041              * @return 用于操作程序节点子树的接口
00042              * \endchinese
00043              * \english
00044              * @return An interface for working with the sub-trees of a program node
00045              * \endenglish
00046              */
00047             ProgramModelPtr getProgramModel();
00048
00049             /**
00050              * @ingroup ProgramApi
00051              * @ref WaypointModel
00052              * \chinese
00053              * @return 用于操作路点的接口
00054              * \endchinese
00055              * \english
00056              * @return An interface for working with waypoints
00057              * \endenglish
00058              */
00059             WaypointModelPtr getWaypointModel();
00060
00061             /**
00062              * @ingroup ProgramApi
00063              * @ref UndoRedoManager
00064              * \chinese
00065              * @return 用于记录程序节点贡献可撤销更改的接口
00066              * \endchinese
00067              * \english
00068              * @return An interface for recording undoable changes for program node
00069              * contributions

```

```

00070      * \endenglish
00071      */
00072      UndoRedoManagerPtr getUndoRedoManager();
00073
00074 private:
00075     friend class DataSwitch;
00076     ProgramApi();
00077     void *d_{ nullptr };
00078 };
00079
00080 } // namespace aubo_scope
00081 } // namespace arcs
00082 #endif // AUBO_SCOPE_PROGRAM_API_H

```

12.206 include/aubo_caps/domain/program/program_model.h 文件参考

```

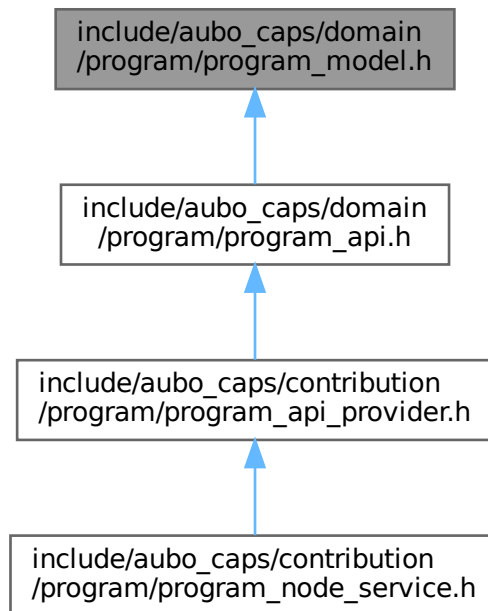
#include <aubo_caps/domain/program/structure/tree_node.h>
#include <aubo_caps/contribution/program/program_node_contribution.h>
#include <aubo_caps/domain/program/nodes/program_node_factory.h>

```

program_model.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class arcs::aubo_scope::ProgramModel
程序模型

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ProgramModel)`

12.207 program_model.h

浏览该文件的文档.

```
00001 #ifndef AUBO_SCOPE_PROGRAM_MODEL_H
00002 #define AUBO_SCOPE_PROGRAM_MODEL_H
00003
00004 #include <aubo_caps/domain/program/structure/tree_node.h>
00005 #include <aubo_caps/contribution/program/program_node_contribution.h>
00006 #include <aubo_caps/domain/program/nodes/program_node_factory.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(ProgramModel);
00011
00012 /**
00013  * @ingroup ProgramApi
00014  * \chinese
00015  * 程序模型
00016  * <h1> 概述 </h1>
00017  * 此接口使得修改 AuboStudio 插件的子树成为可能。树的结构与节点的内容相分离。
00018  * {@link TreeNode} 接口关注树的结构，而 {@link ProgramNode} 接口处理树节点的具体内容。
00019  *
00020  * <h1> 程序修改入门 </h1>
00021  * ProgramModel 包含：
00022  * <ul>
00023  *   <li>getProgramNodeFactory() - 可以创建新的程序节点（包括 AuboScope
00024  *   内置节点和 AuboCap 程序节点）。</li>
00025  *   <li>getRootTreeNode(ProgramNodeContribution root) - 返回给定 AuboCap
00026  *   ProgramNodeContribution 的子树 </li>
00027  * </ul>
00028  *
00029  * TreeNode 接口支持节点的插入和删除。使用它可以获取特定节点的子节点列表。对树节点
00030  * 调用 lockChildSequence 会锁定该节点下的直接子节点。即终端用户无法重新排列、删除
00031  * 子节点或将其他节点插入到子节点序列中。
00032  * \endchinese
00033  * \english
00034  * ProgramModel
00035  * <h1>Summary</h1>
00036  * This interface makes it possible to modify sub-trees of aubo_studio plugins.
00037  * The structure of trees is separated from the contents of the nodes.
00038  * The {@link TreeNode} interface is concerned with the structure of trees
00039  * whereas the {@link ProgramNode} interface deals with the concrete content of
00040  * the nodes of the tree.
00041  *
00042  * <h1>Getting started with program modification</h1>
00043  * A ProgramModel contains a:
00044  * <ul>
00045  *   <li>getProgramNodeFactory() - that makes it possible to create new
00046  *   program nodes (both built-in AuboScope nodes and AuboCap program nodes).
00047  * </li> <li>getRootTreeNode(ProgramNodeContribution root)} - returns the
00048  *   sub-tree of a given AuboCap ProgramNodeContribution</li>
00049  * </ul>
00050  *
00051  * The TreeNode interface supports insertion and removal of nodes. With it you
00052  * can retrieve the list of children of particular nodes. Calling
00053  * lockChildSequence on a tree node, locks the immediate children under the
00054  * node. I.e. children can not be rearranged, deleted or have other nodes
00055  * inserted into the child sequence by the end user.
00056  * \endenglish
00057  */
00058 class ARCS_ABI_EXPORT ProgramModel
00059 {
00060 public:
00061     ProgramModel(ProgramModel &f);
00062     ProgramModel(ProgramModel &&f);
00063     virtual ~ProgramModel();
00064
00065     /**
00066      * \chinese
00067      * 返回一个 ProgramNodeFactory 用于创建程序节点。
00068      *
00069      * @return 用于创建程序节点的工厂
00070      */
00071     ProgramNodeFactory * getProgramNodeFactory();
00072
00073     /**
00074      * \chinese
00075      * 返回给定 AuboCap ProgramNodeContribution 的子树
00076      *
00077      * @param root 给定的 AuboCap ProgramNodeContribution
00078      * @return 子树
00079      */
00080     ProgramNode * getRootTreeNode(ProgramNodeContribution root);
00081
00082     /**
00083      * \chinese
00084      * 锁定子节点序列
00085      *
00086      * @param node 要锁定的节点
00087      */
00088     void lockChildSequence(ProgramNode * node);
00089
00090     /**
00091      * \chinese
00092      * 插入子节点
00093      *
00094      * @param parent 父节点
00095      * @param child 要插入的子节点
00096      */
00097     void insertChild(ProgramNode * parent, ProgramNode * child);
00098
00099     /**
00100      * \chinese
00101      * 删除子节点
00102      *
00103      * @param parent 父节点
00104      * @param child 要删除的子节点
00105      */
00106     void removeChild(ProgramNode * parent, ProgramNode * child);
00107
00108     /**
00109      * \chinese
00110      * 获取子节点列表
00111      *
00112      * @param node 要获取子节点列表的节点
00113      * @return 子节点列表
00114      */
00115     ProgramNode * * getChildList(ProgramNode * node);
00116
00117     /**
00118      * \chinese
00119      * 获取子节点数量
00120      *
00121      * @param node 要获取子节点数量的节点
00122      * @return 子节点数量
00123      */
00124     int getChildCount(ProgramNode * node);
00125
00126     /**
00127      * \chinese
00128      * 获取子节点名称
00129      *
00130      * @param node 要获取子节点名称的节点
00131      * @return 子节点名称
00132      */
00133     const char * getChildName(ProgramNode * node);
00134
00135     /**
00136      * \chinese
00137      * 获取子节点类型
00138      *
00139      * @param node 要获取子节点类型的节点
00140      * @return 子节点类型
00141      */
00142     ProgramNodeType getChildType(ProgramNode * node);
00143
00144     /**
00145      * \chinese
00146      * 获取子节点内容
00147      *
00148      * @param node 要获取子节点内容的节点
00149      * @return 子节点内容
00150      */
00151     ProgramNodeContent * getChildContent(ProgramNode * node);
00152
00153     /**
00154      * \chinese
00155      * 设置子节点内容
00156      *
00157      * @param node 要设置子节点内容的节点
00158      * @param content 子节点内容
00159      */
00160     void setChildContent(ProgramNode * node, ProgramNodeContent * content);
00161
00162     /**
00163      * \chinese
00164      * 获取子节点工厂
00165      *
00166      * @return 子节点工厂
00167      */
00168     ProgramNodeFactory * getProgramNodeFactory();
00169
00170     /**
00171      * \chinese
00172      * 获取子节点名称
00173      *
00174      * @param node 要获取子节点名称的节点
00175      * @return 子节点名称
00176      */
00177     const char * getChildName(ProgramNode * node);
00178
00179     /**
00180      * \chinese
00181      * 获取子节点类型
00182      *
00183      * @param node 要获取子节点类型的节点
00184      * @return 子节点类型
00185      */
00186     ProgramNodeType getChildType(ProgramNode * node);
00187
00188     /**
00189      * \chinese
00190      * 获取子节点数量
00191      *
00192      * @param node 要获取子节点数量的节点
00193      * @return 子节点数量
00194      */
00195     int getChildCount(ProgramNode * node);
00196
00197     /**
00198      * \chinese
00199      * 获取子节点列表
00200      *
00201      * @param node 要获取子节点列表的节点
00202      * @return 子节点列表
00203      */
00204     ProgramNode * * getChildList(ProgramNode * node);
00205
00206     /**
00207      * \chinese
00208      * 插入子节点
00209      *
00210      * @param parent 父节点
00211      * @param child 要插入的子节点
00212      */
00213     void insertChild(ProgramNode * parent, ProgramNode * child);
00214
00215     /**
00216      * \chinese
00217      * 删除子节点
00218      *
00219      * @param parent 父节点
00220      * @param child 要删除的子节点
00221      */
00222     void removeChild(ProgramNode * parent, ProgramNode * child);
00223
00224     /**
00225      * \chinese
00226      * 锁定子节点序列
00227      *
00228      * @param node 要锁定的节点
00229      */
00230     void lockChildSequence(ProgramNode * node);
00231
00232     /**
00233      * \chinese
00234      * 获取子节点名称
00235      *
00236      * @param node 要获取子节点名称的节点
00237      * @return 子节点名称
00238      */
00239     const char * getChildName(ProgramNode * node);
00240
00241     /**
00242      * \chinese
00243      * 获取子节点类型
00244      *
00245      * @param node 要获取子节点类型的节点
00246      * @return 子节点类型
00247      */
00248     ProgramNodeType getChildType(ProgramNode * node);
00249
00250     /**
00251      * \chinese
00252      * 获取子节点数量
00253      *
00254      * @param node 要获取子节点数量的节点
00255      * @return 子节点数量
00256      */
00257     int getChildCount(ProgramNode * node);
00258
00259     /**
00260      * \chinese
00261      * 获取子节点列表
00262      *
00263      * @param node 要获取子节点列表的节点
00264      * @return 子节点列表
00265      */
00266     ProgramNode * * getChildList(ProgramNode * node);
00267
00268     /**
00269      * \chinese
00270      * 插入子节点
00271      *
00272      * @param parent 父节点
00273      * @param child 要插入的子节点
00274      */
00275     void insertChild(ProgramNode * parent, ProgramNode * child);
00276
00277     /**
00278      * \chinese
00279      * 删除子节点
00280      *
00281      * @param parent 父节点
00282      * @param child 要删除的子节点
00283      */
00284     void removeChild(ProgramNode * parent, ProgramNode * child);
00285
00286     /**
00287      * \chinese
00288      * 锁定子节点序列
00289      *
00290      * @param node 要锁定的节点
00291      */
00292     void lockChildSequence(ProgramNode * node);
00293
00294     /**
00295      * \chinese
00296      * 获取子节点名称
00297      *
00298      * @param node 要获取子节点名称的节点
00299      * @return 子节点名称
00300      */
00301     const char * getChildName(ProgramNode * node);
00302
00303     /**
00304      * \chinese
00305      * 获取子节点类型
00306      *
00307      * @param node 要获取子节点类型的节点
00308      * @return 子节点类型
00309      */
00310     ProgramNodeType getChildType(ProgramNode * node);
00311
00312     /**
00313      * \chinese
00314      * 获取子节点数量
00315      *
00316      * @param node 要获取子节点数量的节点
00317      * @return 子节点数量
00318      */
00319     int getChildCount(ProgramNode * node);
00320
00321     /**
00322      * \chinese
00323      * 获取子节点列表
00324      *
00325      * @param node 要获取子节点列表的节点
00326      * @return 子节点列表
00327      */
00328     ProgramNode * * getChildList(ProgramNode * node);
00329
00330     /**
00331      * \chinese
00332      * 插入子节点
00333      *
00334      * @param parent 父节点
00335      * @param child 要插入的子节点
00336      */
00337     void insertChild(ProgramNode * parent, ProgramNode * child);
00338
00339     /**
00340      * \chinese
00341      * 删除子节点
00342      *
00343      * @param parent 父节点
00344      * @param child 要删除的子节点
00345      */
00346     void removeChild(ProgramNode * parent, ProgramNode * child);
00347
00348     /**
00349      * \chinese
00350      * 锁定子节点序列
00351      *
00352      * @param node 要锁定的节点
00353      */
00354     void lockChildSequence(ProgramNode * node);
00355
00356     /**
00357      * \chinese
00358      * 获取子节点名称
00359      *
00360      * @param node 要获取子节点名称的节点
00361      * @return 子节点名称
00362      */
00363     const char * getChildName(ProgramNode * node);
00364
00365     /**
00366      * \chinese
00367      * 获取子节点类型
00368      *
00369      * @param node 要获取子节点类型的节点
00370      * @return 子节点类型
00371      */
00372     ProgramNodeType getChildType(ProgramNode * node);
00373
00374     /**
00375      * \chinese
00376      * 获取子节点数量
00377      *
00378      * @param node 要获取子节点数量的节点
00379      * @return 子节点数量
00380      */
00381     int getChildCount(ProgramNode * node);
00382
00383     /**
00384      * \chinese
00385      * 获取子节点列表
00386      *
00387      * @param node 要获取子节点列表的节点
00388      * @return 子节点列表
00389      */
00390     ProgramNode * * getChildList(ProgramNode * node);
00391
00392     /**
00393      * \chinese
00394      * 插入子节点
00395      *
00396      * @param parent 父节点
00397      * @param child 要插入的子节点
00398      */
00399     void insertChild(ProgramNode * parent, ProgramNode * child);
00400
00401     /**
00402      * \chinese
00403      * 删除子节点
00404      *
00405      * @param parent 父节点
00406      * @param child 要删除的子节点
00407      */
00408     void removeChild(ProgramNode * parent, ProgramNode * child);
00409
00410     /**
00411      * \chinese
00412      * 锁定子节点序列
00413      *
00414      * @param node 要锁定的节点
00415      */
00416     void lockChildSequence(ProgramNode * node);
00417
00418     /**
00419      * \chinese
00420      * 获取子节点名称
00421      *
00422      * @param node 要获取子节点名称的节点
00423      * @return 子节点名称
00424      */
00425     const char * getChildName(ProgramNode * node);
00426
00427     /**
00428      * \chinese
00429      * 获取子节点类型
00430      *
00431      * @param node 要获取子节点类型的节点
00432      * @return 子节点类型
00433      */
00434     ProgramNodeType getChildType(ProgramNode * node);
00435
00436     /**
00437      * \chinese
00438      * 获取子节点数量
00439      *
00440      * @param node 要获取子节点数量的节点
00441      * @return 子节点数量
00442      */
00443     int getChildCount(ProgramNode * node);
00444
00445     /**
00446      * \chinese
00447      * 获取子节点列表
00448      *
00449      * @param node 要获取子节点列表的节点
00450      * @return 子节点列表
00451      */
00452     ProgramNode * * getChildList(ProgramNode * node);
00453
00454     /**
00455      * \chinese
00456      * 插入子节点
00457      *
00458      * @param parent 父节点
00459      * @param child 要插入的子节点
00460      */
00461     void insertChild(ProgramNode * parent, ProgramNode * child);
00462
00463     /**
00464      * \chinese
00465      * 删除子节点
00466      *
00467      * @param parent 父节点
00468      * @param child 要删除的子节点
00469      */
00470     void removeChild(ProgramNode * parent, ProgramNode * child);
00471
00472     /**
00473      * \chinese
00474      * 锁定子节点序列
00475      *
00476      * @param node 要锁定的节点
00477      */
00478     void lockChildSequence(ProgramNode * node);
00479
00480     /**
00481      * \chinese
00482      * 获取子节点名称
00483      *
00484      * @param node 要获取子节点名称的节点
00485      * @return 子节点名称
00486      */
00487     const char * getChildName(ProgramNode * node);
00488
00489     /**
00490      * \chinese
00491      * 获取子节点类型
00492      *
00493      * @param node 要获取子节点类型的节点
00494      * @return 子节点类型
00495      */
00496     ProgramNodeType getChildType(ProgramNode * node);
00497
00498     /**
00499      * \chinese
00500      * 获取子节点数量
00501      *
00502      * @param node 要获取子节点数量的节点
00503      * @return 子节点数量
00504      */
00505     int getChildCount(ProgramNode * node);
00506
00507     /**
00508      * \chinese
00509      * 获取子节点列表
00510      *
00511      * @param node 要获取子节点列表的节点
00512      * @return 子节点列表
00513      */
00514     ProgramNode * * getChildList(ProgramNode * node);
00515
00516     /**
00517      * \chinese
00518      * 插入子节点
00519      *
00520      * @param parent 父节点
00521      * @param child 要插入的子节点
00522      */
00523     void insertChild(ProgramNode * parent, ProgramNode * child);
00524
00525     /**
00526      * \chinese
00527      * 删除子节点
00528      *
00529      * @param parent 父节点
00530      * @param child 要删除的子节点
00531      */
00532     void removeChild(ProgramNode * parent, ProgramNode * child);
00533
00534     /**
00535      * \chinese
00536      * 锁定子节点序列
00537      *
00538      * @param node 要锁定的节点
00539      */
00540     void lockChildSequence(ProgramNode * node);
00541
00542     /**
00543      * \chinese
00544      * 获取子节点名称
00545      *
00546      * @param node 要获取子节点名称的节点
00547      * @return 子节点名称
00548      */
00549     const char * getChildName(ProgramNode * node);
00550
00551     /**
00552      * \chinese
00553      * 获取子节点类型
00554      *
00555      * @param node 要获取子节点类型的节点
00556      * @return 子节点类型
00557      */
00558     ProgramNodeType getChildType(ProgramNode * node);
00559
00560     /**
00561      * \chinese
00562      * 获取子节点数量
00563      *
00564      * @param node 要获取子节点数量的节点
00565      * @return 子节点数量
00566      */
00567     int getChildCount(ProgramNode * node);
00568
00569     /**
00570      * \chinese
00571      * 获取子节点列表
00572      *
00573      * @param node 要获取子节点列表的节点
00574      * @return 子节点列表
00575      */
00576     ProgramNode * * getChildList(ProgramNode * node);
00577
00578     /**
00579      * \chinese
00580      * 插入子节点
00581      *
00582      * @param parent 父节点
00583      * @param child 要插入的子节点
00584      */
00585     void insertChild(ProgramNode * parent, ProgramNode * child);
00586
00587     /**
00588      * \chinese
00589      * 删除子节点
00590      *
00591      * @param parent 父节点
00592      * @param child 要删除的子节点
00593      */
00594     void removeChild(ProgramNode * parent, ProgramNode * child);
00595
00596     /**
00597      * \chinese
00598      * 锁定子节点序列
00599      *
00600      * @param node 要锁定的节点
00601      */
00602     void lockChildSequence(ProgramNode * node);
00603
00604     /**
00605      * \chinese
00606      * 获取子节点名称
00607      *
00608      * @param node 要获取子节点名称的节点
00609      * @return 子节点名称
00610      */
00611     const char * getChildName(ProgramNode * node);
00612
00613     /**
00614      * \chinese
00615      * 获取子节点类型
00616      *
00617      * @param node 要获取子节点类型的节点
00618      * @return 子节点类型
00619      */
00620     ProgramNodeType getChildType(ProgramNode * node);
00621
00622     /**
00623      * \chinese
00624      * 获取子节点数量
00625      *
00626      * @param node 要获取子节点数量的节点
00627      * @return 子节点数量
00628      */
00629     int getChildCount(ProgramNode * node);
00630
00631     /**
00632      * \chinese
00633      * 获取子节点列表
00634      *
00635      * @param node 要获取子节点列表的节点
00636      * @return 子节点列表
00637      */
00638     ProgramNode * * getChildList(ProgramNode * node);
00639
00640     /**
00641      * \chinese
00642      * 插入子节点
00643      *
00644      * @param parent 父节点
00645      * @param child 要插入的子节点
00646      */
00647     void insertChild(ProgramNode * parent, ProgramNode * child);
00648
00649     /**
00650      * \chinese
00651      * 删除子节点
00652      *
00653      * @param parent 父节点
00654      * @param child 要删除的子节点
00655      */
00656     void removeChild(ProgramNode * parent, ProgramNode * child);
00657
00658     /**
00659      * \chinese
00660      * 锁定子节点序列
00661      *
00662      * @param node 要锁定的节点
00663      */
00664     void lockChildSequence(ProgramNode * node);
00665
00666     /**
00667      * \chinese
00668      * 获取子节点名称
00669      *
00670      * @param node 要获取子节点名称的节点
00671      * @return 子节点名称
00672      */
00673     const char * getChildName(ProgramNode * node);
00674
00675     /**
00676      * \chinese
00677      * 获取子节点类型
00678      *
00679      * @param node 要获取子节点类型的节点
00680      * @return 子节点类型
00681      */
00682     ProgramNodeType getChildType(ProgramNode * node);
00683
00684     /**
00685      * \chinese
00686      * 获取子节点数量
00687      *
00688      * @param node 要获取子节点数量的节点
00689      * @return 子节点数量
00690      */
00691     int getChildCount(ProgramNode * node);
00692
00693     /**
00694      * \chinese
00695      * 获取子节点列表
00696      *
00697      * @param node 要获取子节点列表的节点
00698      * @return 子节点列表
00699      */
00700     ProgramNode * * getChildList(ProgramNode * node);
00701
00702     /**
00703      * \chinese
00704      * 插入子节点
00705      *
00706      * @param parent 父节点
00707      * @param child 要插入的子节点
00708      */
00709     void insertChild(ProgramNode * parent, ProgramNode * child);
00710
00711     /**
00712      * \chinese
00713      * 删除子节点
00714      *
00715      * @param parent 父节点
00716      * @param child 要删除的子节点
00717      */
00718     void removeChild(ProgramNode * parent, ProgramNode * child);
00719
00720     /**
00721      * \chinese
00722      * 锁定子节点序列
00723      *
00724      * @param node 要锁定的节点
00725      */
00726     void lockChildSequence(ProgramNode * node);
00727
00728     /**
00729      * \chinese
00730      * 获取子节点名称
00731      *
00732      * @param node 要获取子节点名称的节点
00733      * @return 子节点名称
00734      */
00735     const char * getChildName(ProgramNode * node);
00736
00737     /**
00738      * \chinese
00739      * 获取子节点类型
00740      *
00741      * @param node 要获取子节点类型的节点
00742      * @return 子节点类型
00743      */
00744     ProgramNodeType getChildType(ProgramNode * node);
00745
00746     /**
00747      * \chinese
00748      * 获取子节点数量
00749      *
00750      * @param node 要获取子节点数量的节点
00751      * @return 子节点数量
00752      */
00753     int getChildCount(ProgramNode * node);
00754
00755     /**
00756      * \chinese
00757      * 获取子节点列表
00758      *
00759      * @param node 要获取子节点列表的节点
00760      * @return 子节点列表
00761      */
00762     ProgramNode * * getChildList(ProgramNode * node);
00763
00764     /**
00765      * \chinese
00766      * 插入子节点
00767      *
00768      * @param parent 父节点
00769      * @param child 要插入的子节点
00770      */
00771     void insertChild(ProgramNode * parent, ProgramNode * child);
00772
00773     /**
00774      * \chinese
00775      * 删除子节点
00776      *
00777      * @param parent 父节点
00778      * @param child 要删除的子节点
00779      */
00780     void removeChild(ProgramNode * parent, ProgramNode * child);
00781
00782     /**
00783      * \chinese
00784      * 锁定子节点序列
00785      *
00786      * @param node 要锁定的节点
00787      */
00788     void lockChildSequence(ProgramNode * node);
00789
00790     /**
00791      * \chinese
00792      * 获取子节点名称
00793      *
00794      * @param node 要获取子节点名称的节点
00795      * @return 子节点名称
00796      */
00797     const char * getChildName(ProgramNode * node);
00798
00799     /**
00800      * \chinese
00801      * 获取子节点类型
00802      *
00803      * @param node 要获取子节点类型的节点
00804      * @return 子节点类型
00805      */
00806     ProgramNodeType getChildType(ProgramNode * node);
00807
00808     /**
00809      * \chinese
00810      * 获取子节点数量
00811      *
00812      * @param node 要获取子节点数量的节点
00813      * @return 子节点数量
00814      */
00815     int getChildCount(ProgramNode * node);
00816
00817     /**
00818      * \chinese
00819      * 获取子节点列表
00820      *
00821      * @param node 要获取子节点列表的节点
00822      * @return 子节点列表
00823      */
00824     ProgramNode * * getChildList(ProgramNode * node);
00825
00826     /**
00827      * \chinese
00828      * 插入子节点
00829      *
00830      * @param parent 父节点
00831      * @param child 要插入的子节点
00832      */
00833     void insertChild(ProgramNode * parent, ProgramNode * child);
00834
00835     /**
00836      * \chinese
00837      * 删除子节点
00838      *
00839      * @param parent 父节点
00840      * @param child 要删除的子节点
00841      */
00842     void removeChild(ProgramNode * parent, ProgramNode * child);
00843
00844     /**
00845      * \chinese
00846      * 锁定子节点序列
00847      *
00848      * @param node 要锁定的节点
00849      */
00850     void lockChildSequence(ProgramNode * node);
00851
00852     /**
00853      * \chinese
00854      * 获取子节点名称
00855      *
00856      * @param node 要获取子节点名称的节点
00857      * @return 子节点名称
00858      */
00859     const char * getChildName(ProgramNode * node);
00860
00861     /**
00862      * \chinese
00863      * 获取子节点类型
00864      *
00865      * @param node 要获取子节点类型的节点
00866      * @return 子节点类型
00867      */
00868     ProgramNodeType getChildType(ProgramNode * node);
00869
00870     /**
00871      * \chinese
00872      * 获取子节点数量
00873      *
00874      * @param node 要获取子节点数量的节点
00875      * @return 子节点数量
00876      */
00877     int getChildCount(ProgramNode * node);
00878
00879     /**
00880      * \chinese
00881      * 获取子节点列表
00882      *
00883      * @param node 要获取子节点列表的节点
00884      * @return 子节点列表
00885      */
00886     ProgramNode * * getChildList(ProgramNode * node);
00887
00888     /**
00889      * \chinese
00890      * 插入子节点
00891      *
00892      * @param parent 父节点
00893      * @param child 要插入的子节点
00894      */
00895     void insertChild(ProgramNode * parent, ProgramNode * child);
00896
00897     /**
00898      * \chinese
00899      * 删除子节点
00900      *
00901      * @param parent 父节点
00902      * @param child 要删除的子节点
00903      */
00904     void removeChild(ProgramNode * parent, ProgramNode * child);
00905
00906     /**
00907      * \chinese
00908      * 锁定子节点序列
00909      *
00910      * @param node 要锁定的节点
00911      */
00912     void lockChildSequence(ProgramNode * node);
00913
00914     /**
00915      * \chinese
00916      * 获取子节点名称
00917      *
00918      * @param node 要获取子节点名称的节点
00919      * @return 子节点名称
00920      */
00921     const char * getChildName(ProgramNode * node);
00922
00923     /**
00924      * \chinese
00925      * 获取子节点类型
00926      *
00927      * @param node 要获取子节点类型的节点
00928      * @return 子节点类型
00929      */
00930     ProgramNodeType getChildType(ProgramNode * node);
00931
00932     /**
00933      * \chinese
00934      * 获取子节点数量
00935      *
00936      * @param node 要获取子节点数量的节点
00937      * @return 子节点数量
00938      */
00939     int getChildCount(ProgramNode * node);
00940
00941     /**
00942      * \chinese
00943      * 获取子节点列表
00944      *
00945      * @param node 要获取子节点列表的节点
00946      * @return 子节点列表
00947      */
00948     ProgramNode * * getChildList(ProgramNode * node);
00949
00950     /**
00951      * \chinese
00952      * 插入子节点
00953      *
00954      * @param parent 父节点
00955      * @param child 要插入的子节点
00956      */
00957     void insertChild(ProgramNode * parent, ProgramNode * child);
00958
00959     /**
00960      * \chinese
00961      * 删除子节点
00962      *
00963      * @param parent 父节点
00964      * @param child 要删除的子节点
00965      */
00966     void removeChild(ProgramNode * parent, ProgramNode * child);
00967
00968     /**
00969      * \chinese
00970      * 锁定子节点序列
00971      *
00972      * @param node 要锁定的节点
00973      */
00974     void lockChildSequence(ProgramNode * node);
00975
00976     /**
00977      * \chinese
00978      * 获取子节点名称
00979      *
00980      * @param node 要获取子节点名称的节点
00981      * @return 子节点名称
00982      */
00983     const char * getChildName(ProgramNode * node);
00984
00985     /**
00986      * \chinese
00987      * 获取子节点类型
00988      *
00989      * @param node 要获取子节点类型的节点
00990      * @return 子节点类型
00991      */
00992     ProgramNodeType getChildType(ProgramNode * node);
00993
00994     /**
00995      * \chinese
00996      * 获取子节点数量
00997      *
00998      * @param node 要获取子节点数量的节点
00999      * @return 子节点数量
01000     */
01001     int getChildCount(ProgramNode * node);
01002
01003     /**
01004      * \chinese
01005      * 获取子节点列表
01006      *
01007      * @param node 要获取子节点列表的节点
01008      * @return 子节点列表
01009     */
01010     ProgramNode * * getChildList(ProgramNode * node);
01011
01012     /**
01013      * \chinese
01014      * 插入子节点
01015      *
01016      * @param parent 父节点
01017      * @param child 要插入的子节点
01018     */
01019     void insert
```

```

00070      * \endchinese
00071      * \english
00072      * This method returns a {@link ProgramNodeFactory} to create program nodes.
00073      *
00074      * @return the factory to create program nodes.
00075      * \endenglish
00076      */
00077      ProgramNodeFactoryPtr getProgramNodeFactory();
00078
00079      /**
00080      * \chinese
00081      * 从 ProgramNodeContribution 获取 TreeNode 根节点。从这里可以添加子节点
00082      * 以形成子树。
00083      *
00084      * @param root 要作为子树根的 AuboCap 程序节点
00085      * @return TreeNode 根节点
00086      * \endchinese
00087      * \english
00088      * Gets the {@link TreeNode} root from {@link ProgramNodeContribution}. From
00089      * here, children can be added to form a sub-tree.
00090      *
00091      * @param root The AuboCap program node where a sub-tree is to be rooted.
00092      * @return Returns a {@link TreeNode} root.
00093      * \endenglish
00094      */
00095      TreeNodePtr getTreeNode(ProgramNodeContribution *root);
00096
00097      private:
00098      friend class DataSwitch;
00099      ProgramModel();
00100      void *d_{ nullptr };
00101  };
00102
00103  } // namespace aubo_scope
00104  } // namespace arcs
00105
00106  #endif // AUBO_SCOPE_PROGRAM_MODEL_H

```

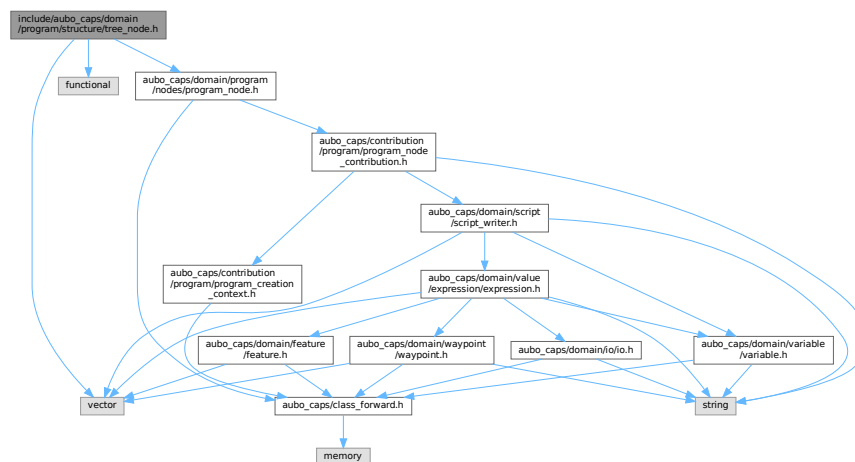
12.208 include/aubo_caps/domain/program/structure/tree_node.h 文件参考

```

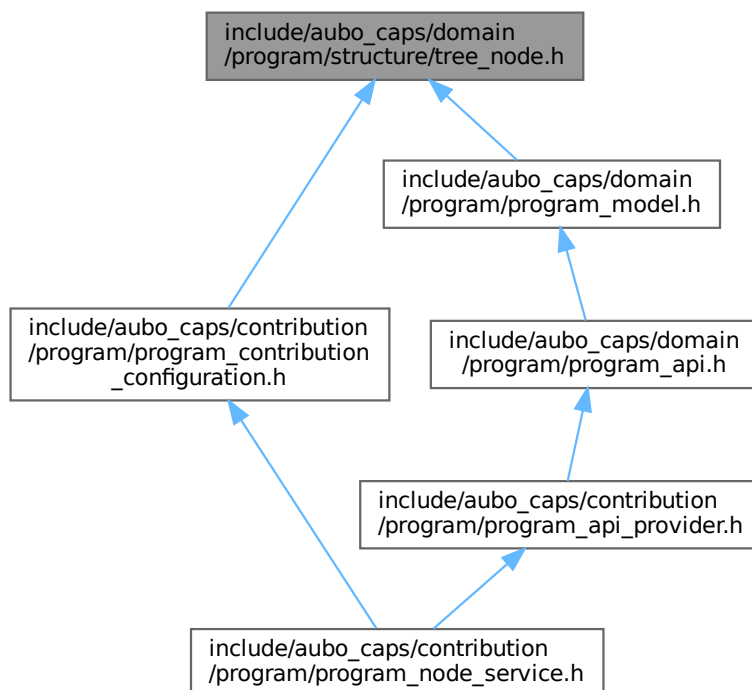
#include <vector>
#include <functional>
#include <aubo_caps/domain/program/nodes/program_node.h>

```

tree_node.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- `class arcs::aubo_scope::TreeNode`

树节点此接口表示程序树中的一个节点，可用于构建以 *AuboCap* 程序节点为根的子树。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (TreeNode)`

12.209 tree_node.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_TREE_NODE_H
00002 #define AUBO_SCOPE_TREE_NODE_H
00003
00004 #include <vector>
00005 #include <functional>
00006 #include <aubo_caps/domain/program/nodes/program_node.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(TreeNode);
00011
00012 /**
00013  * \chinese
00014  * 树节点
00015  * 此接口表示程序树中的一个节点，可用于构建以 AuboCap 程序节点为根的子树。

```

```

00016 *
00017 * 使用 {@link ProgramModel#getRootTreeNode(ProgramNodeContribution)} 获取子树
00018 * 的根节点后, 即可添加子节点。每次调用 {@link TreeNode#addChild(ProgramNode)}
00019 * 都会返回一个新的 {@link TreeNode}, 该节点又可以作为另一个子树的根节点。
00020 * \endchinese
00021 * \english
00022 * TreeNode
00023 * This interface represents a node in the program tree that can be used to
00024 * construct a sub-tree rooted in a AudoCap program node.
00025 *
00026 * Using the {@link ProgramModel#getRootTreeNode(ProgramNodeContribution)} to
00027 * obtain a root for the sub-tree, it is possible to add children. For each call
00028 * to {@link TreeNode#addChild(ProgramNode)}, a new {@link TreeNode} is
00029 * returned, that can, in turn, act as a root for yet another sub-tree.
00030 * \endenglish
00031 */
00032 class ARCS_ABI_EXPORT TreeNode
00033 {
00034 public:
00035     TreeNode(TreeNode &f);
00036     TreeNode(TreeNode &&f);
00037     virtual ~TreeNode();
00038
00039 /**
00040  * \chinese
00041  * 向子树添加一个子程序节点。
00042  *
00043  * @param program_node 使用 ProgramNodeFactory 构建的 ProgramNode
00044  * @return 返回一个 TreeNode, 可用于向新添加的子节点添加子节点。
00045  * @throws TreeStructureException 如果不允许在此位置插入 ProgramNode
00046  * @throws IllegalStateException 如果在 UndoableChanges 范围外从基于 Qt 的
00047  * AudoCap 调用
00048  * \endchinese
00049  * \english
00050  * Add a child program node to the sub-tree.
00051  *
00052  * @param program_node the {@link ProgramNode} constructed using the {@link
00053  * ProgramNodeFactory}
00054  * @return Returns a TreeNode that can be used to add children to the newly
00055  * added child.
00056  * @throws TreeStructureException If it is not allowed to insert the {@link
00057  * ProgramNode} at this position a
00058  * {@link TreeStructureException} will be thrown.
00059  * @throws IllegalStateException if called from a Qt-based AudoCap
00060  * outside of an {@link UndoableChanges} scope (see also {@link
00061  * UndoRedoManager}).
00062  * \endenglish
00063  */
00064     TreeNodePtr addChild(ProgramNodePtr program_node);
00065
00066 /**
00067  * \chinese
00068  * 在子树中, 在现有选中子节点之前插入一个子程序节点。将选中子节点及其后的节点
00069  * 移到新添加子节点之后的位置。
00070  *
00071  * @param existingChildNode 此 TreeNode 的现有 TreeNode 子节点
00072  * @param program_node 使用 ProgramNodeFactory 构建的 ProgramNode
00073  * @return 返回一个 TreeNode, 可用于向新添加的子节点添加子节点。
00074  * @throws TreeStructureException 如果不允许在此位置插入或选中子节点不是
00075  * 此 TreeNode 的子节点
00076  * @throws IllegalStateException 如果在 UndoableChanges 范围外调用
00077  * \endchinese
00078  * \english
00079  * Inserts a child program node in the sub-tree directly before the existing
00080  * selected child node. Shifts the selected child node and any subsequent
00081  * nodes to positions after the newly added child.
00082  *
00083  * @param existingChildNode existing TreeNode child of this TreeNode.
00084  * @param program_node the {@link ProgramNode} constructed using the {@link
00085  * ProgramNodeFactory}.
00086  * @return Returns a TreeNode that can be used to add children to the newly
00087  * added child.
00088  * @throws TreeStructureException If it is not allowed to insert the {@link
00089  * ProgramNode} at this position or if the selected child node is not a
00090  * child of this TreeNode a {@link TreeStructureException} will be thrown.
00091  * @throws IllegalStateException if called from a Qt-based AudoCap
00092  * outside of an {@link UndoableChanges} scope (see also {@link
00093  * UndoRedoManager}).
00094  * \endenglish
00095  */
00096     TreeNodePtr insertChildBefore(TreeNodePtr existingChildNode,
00097                                   ProgramNodePtr program_node);
00098
00099 /**
00100  * \chinese
00101  * 在子树中, 在现有选中子节点之后插入一个子程序节点。将后续节点移到新添加
00102  * 子节点之后的位置。

```

```

00103      *
00104      * @param existingChildNode 此 TreeNode 的现有 TreeNode 子节点
00105      * @param program_node 使用 ProgramNodeFactory 构建的 ProgramNode
00106      * @return 返回一个 TreeNode, 可用于向新添加的子节点添加子节点。
00107      * @throws TreeStructureException 如果不允许在此位置插入或选中子节点不是
00108      * 此 TreeNode 的子节点
00109      * @throws IllegalStateException 如果在 UndoableChanges 范围外调用
00110      * \endchinese
00111      * \english
00112      * Inserts a child program node under in the sub-tree directly after the
00113      * existing selected child node. Shifts any subsequent nodes to positions
00114      * after the newly added child.
00115      *
00116      * @param existingChildNode existing TreeNode child of this TreeNode.
00117      * @param program_node the {@link ProgramNode} constructed using the {@link
00118      * ProgramNodeFactory}.
00119      * @return Returns a TreeNode that can be used to add children to the newly
00120      * added child.
00121      * @throws TreeStructureException If it is not allowed to insert the {@link
00122      * ProgramNode} at this position or if the selected child node is not a
00123      * child of this TreeNode a {@link TreeStructureException} will be thrown.
00124      * @throws IllegalStateException if called from a Qt-based AudoCap
00125      * outside of an {@link UndoableChanges} scope (see also {@link
00126      * UndoRedoManager}).
00127      * \endenglish
00128      */
00129      TreeNodePtr insertChildAfter(TreeNodePtr existingChildNode,
00130                                   ProgramNodePtr program_node);
00131
00132      /**
00133      * \chinese
00134      * 剪切子树中的一个子程序节点。
00135      *
00136      * @param child 此 TreeNode 的现有 TreeNode 子节点
00137      * @return 返回被剪切掉的 TreeNode
00138      * @throws TreeStructureException 如果不允许剪切此 ProgramNode 或子节点不是
00139      * 此 TreeNode 的子节点
00140      * @throws IllegalStateException 如果在 UndoableChanges 范围外调用
00141      * \endchinese
00142      * \english
00143      * Cut a child program node under in the sub-tree directly after the
00144      * existing selected child node. Shifts any subsequent nodes to positions
00145      * after the newly added child.
00146      *
00147      * @param child existing TreeNode child of this TreeNode.
00148      * @return Returns the TreeNode that has been cut off.
00149      * @throws TreeStructureException If it is not allowed to cut the {@link
00150      * ProgramNode} or if the selected child node is not a child of this
00151      * TreeNode a {@link TreeStructureException} will be thrown.
00152      * @throws IllegalStateException if called from a Qt-based AudoCap
00153      * outside of an {@link UndoableChanges} scope (see also {@link
00154      * UndoRedoManager}).
00155      * \endenglish
00156      */
00157      TreeNodePtr cutChildNode(TreeNodePtr child);
00158
00159      /**
00160      * \chinese
00161      * 从子树中移除子节点。注意: 移除最后一个子节点将触发插入一个
00162      * {@literal <empty>} 子节点。
00163      *
00164      * @param program_node 要移除的 TreeNode 子节点
00165      * @return 成功移除返回 <code>true</code>, 否则返回 <code>false</code>
00166      * @throws TreeStructureException 如果移除子节点会使树处于非法状态
00167      * @throws IllegalStateException 如果在 UndoableChanges 范围外调用
00168      * \endchinese
00169      * \english
00170      * Removes a child node from the sub-tree. Be aware that removing the last
00171      * child will trigger the insertion of an
00172      * {@literal <empty>} child node.
00173      *
00174      * @param program_node The TreeNode child to be removed.
00175      * @return Returns <code>true</code> if removed successfully.
00176      * <code>false</code> otherwise.
00177      * @throws TreeStructureException If the removed child would leave the tree
00178      * in an illegal state a
00179      * {@link TreeStructureException} will be thrown.
00180      * @throws IllegalStateException if called from a Qt-based AudoCap
00181      * outside of an {@link UndoableChanges} scope (see also {@link
00182      * UndoRedoManager}).
00183      * \endenglish
00184      */
00185      bool removeChild(TreeNodePtr program_node);
00186
00187      /**
00188      * \chinese
00189      * @return TreeNode 对象列表, 表示此 TreeNode 的所有子节点 (包括以编程方式

```



```

00190     * 添加的和终端用户插入的)
00191     * \endchinese
00192     * \english
00193     * @return a list of <code>TreeNode</code> objects that represents all the
00194     * children of this <code>TreeNode</code> (both programmatically added as
00195     * well as inserted by the end user).
00196     * \endenglish
00197     */
00198     std::vector<TreeNodePtr> getChildren();
00199
00200     std::vector<TreeNodePtr> getChildrenNotSuppressed();
00201
00202     /**
00203     * \chinese
00204     * @return 节点的父 TreeNode
00205     * \endchinese
00206     * \english
00207     * @return Parent treeNode of the node
00208     * \endenglish
00209     */
00210     TreeNodePtr getParent();
00211
00212     /**
00213     * \chinese
00214     * @return 返回子树中此位置的 ProgramNode。可以是内置 AuboScope 程序节点
00215     * 或 AuboCap 程序节点。
00216     * \endchinese
00217     * \english
00218     * @return Returns the {@link ProgramNode} at this position in the sub-tree.
00219     * Can either be a built-in AuboScope program node (provided by AUBO
00220     * Robots) or a AuboCap program node.
00221     * \endenglish
00222     */
00223     ProgramNodePtr getProgramNode();
00224     bool isSuppressed();
00225
00226     /**
00227     * \chinese
00228     * 节点是否在程序树中；不在程序树中的节点可能已被剪切或删除。
00229     *
00230     * @return <code>true</code> 节点在程序树中，
00231     * <code>false</code> 节点不在程序树中（可能已被剪切或删除）
00232     * \endchinese
00233     * \english
00234     * Whether the node is in the program tree; nodes not in the program tree
00235     * may have been cut or deleted.
00236     *
00237     * @return Returns <code>true</code> The node is in the program tree.
00238     * <code>false</code> The node is not in the program tree; such nodes may
00239     * have been cut or deleted.
00240     * \endenglish
00241     */
00242     bool isInProgramTree();
00243
00244     /**
00245     * \chinese
00246     * <p> 获取此 TreeNode 子树中子程序节点（ProgramNode 实例）对应的 TreeNode
00247     * 实例。</p>
00248     * 此方法可用于在遍历父节点子树时，获取遇到子程序节点下的子树访问权限。
00249     *
00250     * @param program_node 需要获取对应 TreeNode 表示的程序节点，不能为
00251     * <code>null</code>
00252     * @return 指定程序节点的 TreeNode 实例
00253     * @throws ProgramNodeNotInSubTreeException 当程序节点不在子树中时
00254     * \endchinese
00255     * \english
00256     * <p>
00257     * Gets a corresponding {@link TreeNode} instance for a child program node
00258     * (a {@link ProgramNode} instance) in the sub-tree under this {@link
00259     * TreeNode}.
00260     * </p>
00261     * This method can for instance be used to gain access to the sub-tree under
00262     * a child program node encountered when iterating the sub-tree of the
00263     * parent node using the {@link ProgramNodeVisitor} (or any derived sub
00264     * class thereof).
00265     *
00266     * @param program_node program node to get a corresponding {@link TreeNode}
00267     * representation for, not <code>null</code>. Can either be a built-in
00268     * AuboScope program node (provided by AUBO Robots) or a AuboCap
00269     * program node.
00270     * @return the <code>TreeNode</code> instance for the specified program
00271     * node.
00272     * @throws ProgramNodeNotInSubTreeException when the program node cannot be
00273     * found because it is not in the sub-tree.
00274     * \endenglish
00275     */
00276     TreeNodePtr locateDescendantTreeNode(ProgramNodePtr program_node);

```

```

00277
00278 /**
00279  * \chinese
00280  * 配置子节点是否可被终端用户重新排列、删除或将其他节点插入到子节点序列中。
00281  *
00282  * @param isChildSequenceLocked 如果为 <code>true</code>, 则此
00283  * TreeNode 下的直接子节点将被锁定
00284  * \endchinese
00285  * \english
00286  * Configures whether or not child nodes can be rearranged, deleted or have
00287  * other nodes inserted into the child sequence by the end user.
00288  *
00289  * @param isChildSequenceLocked If <code>true</code> then the immediate
00290  * children under this <code>TreeNode</code> will be locked.
00291  * \endenglish
00292  */
00293 void setChildSequenceLocked(bool isChildSequenceLocked);
00294
00295 /**
00296  * \chinese
00297  * <p> 此方法以深度优先方式遍历此树节点下的整个子树（对应于程序树中的自顶向下方式）。</p>
00298  * <p> 使用节点访问器进行回调，回调的重载取决于遇到的节点类型。</p>
00299  * <p> 所有 visit 方法都具有程序节点、兄弟索引和深度作为参数，以便在需要时过滤节点。</p>
00300  *
00301  * @param nodeVisitor 回调实例
00302  * @return 0 表示正常退出，非零表示异常或错误导致提前退出
00303  * \endchinese
00304  * \english
00305  * <p>
00306  * This method traverses the entire sub-tree under this tree node in a
00307  * depth-first fashion (this corresponds to a top-down approach in the
00308  * program tree).
00309  * </p>
00310  * <p>
00311  * A node visitor is used for callbacks to the visit-overloads you choose to
00312  * override. The overload called depends on the node type encountered.
00313  * Override the overloads for the node types you are concerned with. All
00314  * visit-methods have the program node, sibling index and depth as arguments
00315  * to help filter nodes if needed.
00316  * </p>
00317  * <p>
00318  * The node visitor can be either a {@link ProgramNodeVisitor}
00319  * implementation with optional overrides or a
00320  * {@link ProgramNodeInterfaceVisitor} implementation. In the latter
00321  * case, the
00322  * {@link ProgramNodeInterfaceVisitor#visitAs(Object, int, int)}
00323  * method must be implemented.
00324  * </p>
00325  * <p>
00326  * The {@link ProgramNodeInterfaceVisitor} can be used when targeting
00327  * AuboCap program nodes implementing the (generic) type parameter specified
00328  * in {@link ProgramNodeInterfaceVisitor} (see also
00329  * {@link ProgramNode#getAs(Class)}).
00330  * </p>
00331  * Note that this method is sometimes called <code>accept()</code> in the
00332  * Visitor software design pattern.
00333  *
00334  * @param nodeVisitor the instance callbacks are made to.
00335  * @return 0 for normal exit.
00336  *         non-zero for exception or error causing premature exit.
00337  * \endenglish
00338  */
00339 int traverse(std::function<int(ProgramNodePtr, int, int)> nodeVisitor);
00340
00341 private:
00342     friend class DataSwitch;
00343     TreeNode();
00344     void *d_{ nullptr };
00345 };
00346
00347 } // namespace aubo_scope
00348 } // namespace arcs
00349 #endif // AUBO_SCOPE_TREE_NODE_H

```

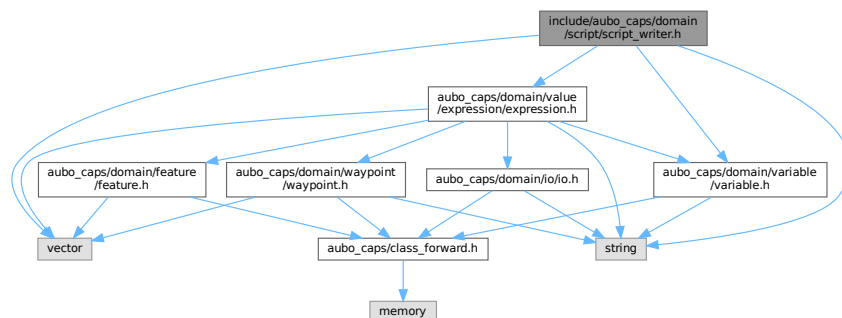
12.210 include/aubo_caps/domain/script/script_writer.h 文件参考

```

#include <vector>
#include <string>
#include <aubo_caps/domain/variable/variable.h>
#include <aubo_caps/domain/value/expression/expression.h>

```

script_writer.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::ScriptWriter`
脚本写入器此接口提供生成脚本代码的支持。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ScriptWriter)`

12.211 script_writer.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_SCRIPT_WRITER_H
00002 #define AUBO_SCOPE_SCRIPT_WRITER_H
00003
00004 #include <vector>
00005 #include <string>
00006 #include <aubo_caps/domain/variable/variable.h>
00007 #include <aubo_caps/domain/value/expression/expression.h>
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011 ARCS_CLASS_FORWARD(ScriptWriter);
00012
00013 /**
00014  * \chinese
00015  * 脚本写入器
00016  * 此接口提供生成脚本代码的支持。
00017  * \endchinese
00018  * \english
00019  * ScriptWriter
00020  * This interface provides support for generating Script code.
00021  * \endenglish
00022  */
00023 class ARCS_ABI_EXPORT ScriptWriter
00024 {
00025 public:
00026     ScriptWriter(ScriptWriter &f);
00027     ScriptWriter(ScriptWriter &&f);
00028     virtual ~ScriptWriter();
  
```

```

00029
00030 void setLabel(int lineno, const std::string &comment);
00031
00032 /**
00033  * \chinese
00034  * 使用自动缩进添加一行脚本代码。
00035  * @param script_line 要追加的脚本代码行。
00036  * \endchinese
00037  * \english
00038  * Adds a single line of script code using auto-indentation.
00039  *
00040  * @param script_line single line of script code to append.
00041  * \endenglish
00042  */
00043 void appendLine(const std::string &script_line);
00044 void appendVectorDouble(const std::vector<double> &value);
00045
00046 /**
00047  * \chinese
00048  * 按原样追加脚本代码，不使用自动缩进。
00049  * @param script 要追加的脚本代码。
00050  * \endchinese
00051  * \english
00052  * Adds script code appending the script code as is without using
00053  * auto-indentation.
00054  *
00055  * @param script script code to append.
00056  * \endenglish
00057  */
00058 void appendRaw(const std::string &script);
00059
00060 /**
00061  * \chinese
00062  * 生成包含完整脚本代码的字符串。
00063  * @return 生成的脚本字符串。
00064  * \endchinese
00065  * \english
00066  * Generate a string with the full script code.
00067  *
00068  * @return the resulting script as a string.
00069  * \endenglish
00070  */
00071 std::string generateScript();
00072
00073 /**
00074  * \chinese
00075  * 当为具有子节点的节点生成脚本代码时使用此方法。让子节点生成代码。
00076  * \endchinese
00077  * \english
00078  * Use this method when generating script code for a node that has children
00079  * nodes. Let children nodes generate code.
00080  * \endenglish
00081  */
00082 void writeChildren();
00083
00084 void assign(VariablePtr variable, const std::string &expression,
00085            bool sync = false);
00086
00087 /**
00088  * \chinese
00089  * <p> 变量赋值。在无 local 或 global 限定符的情况下赋值。</p>
00090  * @param variableName 变量名称。
00091  * @param expression 赋给变量的表达式。
00092  * \endchinese
00093  * \english
00094  * <p>Variable assignment.
00095  * Assigns the variable without a local or global qualifier.</p>
00096  * @param variableName name of the variable, not null.
00097  * @param expression expression that is assigned to the variable, not null.
00098  * \endenglish
00099  */
00100 void assign(const std::string &variableName,
00101            const ExpressionPtr &expression, bool sync = false);
00102
00103 /**
00104  * \chinese
00105  * <p> 变量赋值。</p>
00106  * @param variable 变量。
00107  * @param expression 表达式。
00108  * \endchinese
00109  * \english
00110  * <p>Variable assignment.</p>
00111  * @param variable the variable to assign an expression to, not null.
00112  * @param expression expression that is assigned to the variable, not null.
00113  * \endenglish
00114  */
00115 void assign(VariablePtr variable, const ExpressionPtr &expression,

```

```

00116         bool sync = false);
00117
00118     /**
00119     * \chinese
00120     * 变量值加 1。
00121     * @param variable_name 要递增的变量名称。
00122     * \endchinese
00123     * \english
00124     * Add 1 to the variable value.
00125     * @param variable_name the variable to increment.
00126     * \endenglish
00127     */
00128     void incrementVariable(const std::string &variable_name);
00129
00130     /**
00131     * \chinese 添加注释。 \endchinese
00132     * \english Add a note. @param expression the note expression. \endenglish
00133     */
00134     void note(const std::string &expression);
00135
00136     /**
00137     * \chinese 休眠指定秒数。 \endchinese
00138     * \english Sleep for a number of seconds. @param seconds amount of time to sleep in seconds. \endenglish
00139     */
00140     void sleep(double seconds);
00141
00142     /** \chinese 用完当前帧剩余时间。 \endchinese \english Sync. \endenglish */
00143     void sync();
00144
00145     /** \chinese 定义函数。 \endchinese \english Define a function. @param func_name function name. \endenglish */
00146     void defineFunction(const std::string &func_name);
00147     void anonyFunction(const std::string &func_name);
00148
00149     void setRobotIndex(int robot_index);
00150
00151     /** \chinese 返回。 \endchinese \english Return from method. \endenglish */
00152     void returnMethod();
00153
00154     /** \chinese 插入 end。 \endchinese \english Insert an end. \endenglish */
00155     void end();
00156
00157     /** \chinese 插入空行。 \endchinese \english Insert an empty line. \endenglish */
00158     void lineFeed();
00159
00160     /**
00161     * \chinese 设置负载质量和重心。 \endchinese
00162     * \english Sets the mass and Center of Gravity of the payload. \endenglish
00163     */
00164     void setPayload(double mass, double x, double y, double z);
00165
00166     /** \chinese 设置 TCP。 \endchinese \english Set the Tool Center Point. \endenglish */
00167     void setTcp(const std::vector<double> &pose);
00168
00169     /** \chinese 开始 if 条件。 \endchinese \english Start an if-conditional. @param expression the expression. \endenglish */
00170     void ifCondition(const ExpressionPtr &expression);
00171
00172     /** \chinese 开始取反的 if 条件。 \endchinese \english Start a negated if-conditional. @param expression the expression. \endenglish */
00173     void ifNotCondition(const ExpressionPtr &expression);
00174
00175     /** \chinese 添加 else-if 分支。 \endchinese \english Adds an else-if branch. @param expression the expression. \endenglish */
00176     void elseifCondition(const ExpressionPtr &expression);
00177
00178     /** \chinese 添加 else 分支。 \endchinese \english Adds an else branch. \endenglish */
00179     void elseCondition();
00180
00181     /** \chinese 开始 for 循环。 \endchinese \english Starts a for-loop. @param count loop count, @param step loop step. \endenglish */
00182     void forCondition(int count, int step);
00183
00184     /** \chinese 开始 while true 循环。 \endchinese \english Starts a while true loop. \endenglish */
00185     void whileTrue();
00186
00187     /** \chinese 开始 while 循环。 \endchinese \english Starts a while-loop. @param expression the loop invariant. \endenglish */
00188     void whileCondition(const ExpressionPtr &expression);
00189
00190     /** \chinese 开始取反的 while 循环。 \endchinese \english Starts a negated while-loop. @param expression the loop invariant. \endenglish */
00191     void whileNot(const ExpressionPtr &expression);
00192
00193     /**
00194     * \chinese 定义线程。 \endchinese
00195     * \english Start a thread definition. @param thread_name the thread name, @param loop_or_not whether loops. \endenglish

```

```

00196     */
00197 void defineThread(const std::string &thread_name, bool loop_or_not);
00198
00199 /** \chinese 启动线程。 \endchinese \english Start a thread. @param thread_name the thread name. \endenglish */
00200 void runThread(const std::string &thread_name);
00201
00202 /** \chinese 终止线程。 \endchinese \english Kill a thread. @param thread_name the thread name. \endenglish */
00203 void killThread(const std::string &thread_name);
00204
00205 /**
00206  * \chinese
00207  * 获取可在脚本中使用的已注册变量名。
00208  * @param variable_name 变量名称。
00209  * @return 可在脚本中使用的变量名。
00210  * \endchinese
00211  * \english
00212  * Returns a registered variable name that can be used in a script.
00213  * @param variable_name a variable name.
00214  * @return variable name that can be used in a script.
00215  * \endenglish
00216  */
00217 std::string getResolvedVariable(const std::string &variable_name);
00218
00219 void increaseIndent();
00220 void decreaseIndent();
00221
00222 private:
00223     friend class DataSwitch;
00224     ScriptWriter();
00225     void *d_{ nullptr };
00226 };
00227
00228 } // namespace aubo_scope
00229 } // namespace arcs
00230
00231 #endif // AUBO_SCOPE_SCRIPT_WRITER_H

```

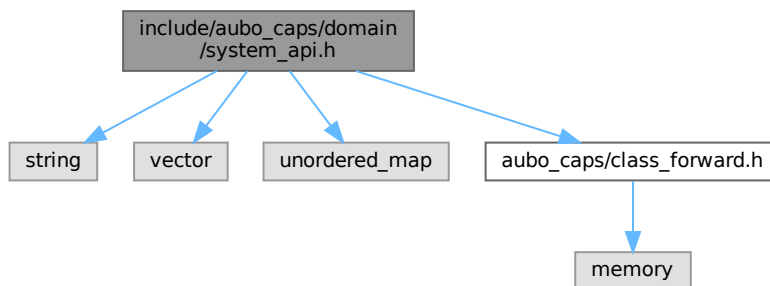
12.212 include/auco_caps/domain/system_api.h 文件参考

```

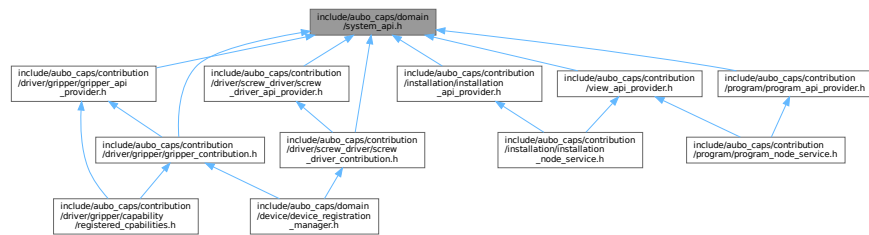
#include <string>
#include <vector>
#include <unordered_map>
#include <auco_caps/class_forward.h>

```

system_api.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- struct `arcs::aubo_scope::IoConfigure`
- class `arcs::aubo_scope::SystemApi`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

枚举

- enum class `arcs::aubo_scope::UnitType` { `arcs::aubo_scope::METRIC` , `arcs::aubo_scope::IMPERIAL` }

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`SystemApi`)

12.213 system_api.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_SYSTEM_API_H
00002 #define AUBO_SCOPE_SYSTEM_API_H
00003
00004 #include <string>
00005 #include <vector>
00006 #include <unordered_map>
00007 #include <aubo_caps/class_forward.h>
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011
00012 enum class UnitType
00013 {
00014     METRIC, ///< 公制 mm Kg
00015     IMPERIAL ///< 英制
00016 };
00017
00018 struct IoConfigure
00019 {
00020     std::string label;
00021     int num;
00022     std::vector<std::string> names;
00023     std::vector<int> sequences;
00024     std::vector<int> functions;
00025     std::vector<int> safety_functions;
00026     std::vector<int> safety_functions_fixed;
00027     bool function_lock{ false };
00028     std::vector<int> types;
00029     std::vector<double> mins;
00030     std::vector<double> maxs;
00031 };
00032
00033 ARCS_CLASS_FORWARD(SystemApi);
00034
00035 /**

```

```

00036 * \chinese
00037 * @defgroup SystemApi SystemApi(系统信息)
00038 * @ingroup SystemApi
00039 * 系统信息接口
00040 * 提供对 AuboScope 软件系统信息的访问, 包括软件版本号、系统语言区域、单位制、机器人
00041 * 序列号, 以及机器人的关节位置范围、速度、加速度和负载等参数信息。
00042 * \endchinese
00043 * \english
00044 * @defgroup SystemApi SystemApi
00045 * @ingroup SystemApi
00046 * SystemApi
00047 * Provides access to system information of the AuboScope software, including
00048 * software version numbers, system locale, unit types, the robot serial number,
00049 * as well as robot joint position limits, velocity, acceleration, and payload
00050 * parameters.
00051 * \endenglish
00052 */
00053
00054 class ARCS_ABI_EXPORT SystemApi
00055 {
00056 public:
00057     SystemApi(SystemApi &f);
00058     SystemApi(SystemApi &&f);
00059     virtual ~SystemApi();
00060
00061     /**
00062     * @ingroup SystemApi
00063     * \chinese
00064     * 获取软件版本号
00065     * @return 版本号。
00066     * \endchinese
00067     * \english
00068     * Get the version of Software.
00069     * @return the major version of Software.
00070     * \endenglish
00071     */
00072     int getMajorVersion();
00073
00074     /**
00075     * @ingroup SystemApi
00076     * \chinese
00077     * 获取软件次版本号
00078     * @return 次版本号。
00079     * \endchinese
00080     * \english
00081     * Get the minor version of Software.
00082     * @return the minor version of Software.
00083     * \endenglish
00084     */
00085     int getMinorVersion();
00086
00087     /**
00088     * @ingroup SystemApi
00089     * \chinese
00090     * 获取软件 Bug 修复版本号
00091     * @return Bug 修复版本号。
00092     * \endchinese
00093     * \english
00094     * Get the bug fix version of Software.
00095     * @return the bug fix version of Software.
00096     * \endenglish
00097     */
00098     int getBugfixVersion();
00099
00100     /**
00101     * @ingroup SystemApi
00102     * \chinese
00103     * 获取软件构建号
00104     * @return 构建号。
00105     * \endchinese
00106     * \english
00107     * Get the build number of Software.
00108     * @return the build number of Software.
00109     * \endenglish
00110     */
00111     int getBuildNumber();
00112
00113     /**
00114     * @ingroup SystemApi
00115     * \chinese
00116     * 返回系统的 {@link Locale} 语言区域设置。
00117     *
00118     * @return 语言区域。
00119     * \endchinese
00120     * \english
00121     * Returns the {@link Locale} for the system.
00122     */

```



```

00123     *
00124     * @return Locale.
00125     * \endenglish
00126     */
00127     std::string getLocale();
00128
00129     /**
00130     * @ingroup SystemApi
00131     * \chinese
00132     * 返回系统中编程语言使用的 {@link Locale} 语言区域设置。
00133     *
00134     * @return 语言区域。
00135     * \endchinese
00136     * \english
00137     * Returns the {@link Locale} for the programming language used in the
00138     * system.
00139     *
00140     * @return Locale.
00141     * \endenglish
00142     */
00143     std::string getLocaleForProgrammingLanguage();
00144
00145     /**
00146     * @ingroup SystemApi
00147     * \chinese
00148     * 以枚举类型返回系统的单位制。
00149     *
00150     * @return 单位制枚举。
00151     * \endchinese
00152     * \english
00153     * Returns the UnitType of the system as an enum.
00154     *
00155     * @return UnitType.
00156     * \endenglish
00157     */
00158     UnitType getUnitType();
00159
00160     /**
00161     * @ingroup SystemApi
00162     * \chinese
00163     * 获取机器人的序列号
00164     * @return 序列号。
00165     * \endchinese
00166     * \english
00167     * Returns the serial number of the robot.
00168     * @return the serial number of the robot.
00169     * \endenglish
00170     */
00171     std::string getSerialNumber();
00172
00173     /**
00174     * @ingroup SystemApi
00175     * \chinese
00176     * 检查是否控制真实机器人
00177     * @return <code>true</code> 表示控制真实物理机器人, <code>false</code> 表示处于仿真模式。
00178     * \endchinese
00179     * \english
00180     * Returns <code>true</code> if controlling the real, physical robot and
00181     * <code>false</code> if in simulation mode.
00182     * @return <code>true</code> if controlling the real, physical robot and
00183     * <code>false</code> if in simulation mode.
00184     * \endenglish
00185     */
00186     bool isRealRobot();
00187
00188     /**
00189     * @ingroup SystemApi
00190     * \chinese
00191     * 获取机器人各关节的最大位置
00192     * @return 各关节最大位置, 单位为弧度。
00193     * \endchinese
00194     * \english
00195     * Returns the max positions of robot joints.
00196     * @return Max Positions of robot joints in radian.
00197     * \endenglish
00198     */
00199     std::vector<double> getJointMaxPositions();
00200
00201     /**
00202     * @ingroup SystemApi
00203     * \chinese
00204     * 获取机器人各关节的最小位置
00205     * @return 各关节最小位置, 单位为弧度。
00206     * \endchinese
00207     * \english
00208     * Returns the min positions of robot joints.
00209     * @return Min Positions of robot joints in radian.

```

```

00210     * \endenglish
00211     */
00212     std::vector<double> getJointMinPositions();
00213
00214     /**
00215     * @ingroup SystemApi
00216     * \chinese
00217     * 获取关节默认速度
00218     * @return 关节默认速度。
00219     * \endchinese
00220     * \english
00221     * Returns the default velocity of robot joints.
00222     * @return the default velocity of robot joints.
00223     * \endenglish
00224     */
00225     double getJointDefaultVelocity();
00226
00227     /**
00228     * @ingroup SystemApi
00229     * \chinese
00230     * 获取关节默认加速度
00231     * @return 关节默认加速度。
00232     * \endchinese
00233     * \english
00234     * Returns the default acceleration of robot joints.
00235     * @return the default acceleration of robot joints.
00236     * \endenglish
00237     */
00238     double getJointDefaultAcceleration();
00239
00240     /**
00241     * @ingroup SystemApi
00242     * \chinese
00243     * 获取各关节最大速度
00244     * @return 各关节最大速度。
00245     * \endchinese
00246     * \english
00247     * Returns the max velocity of robot joints.
00248     * @return the max velocity of robot joints.
00249     * \endenglish
00250     */
00251     std::vector<double> getJointMaxVelocity();
00252
00253     /**
00254     * @ingroup SystemApi
00255     * \chinese
00256     * 获取各关节最大加速度
00257     * @return 各关节最大加速度。
00258     * \endchinese
00259     * \english
00260     * Returns the max acceleration of robot joints.
00261     * @return the max acceleration of robot joints.
00262     * \endenglish
00263     */
00264     std::vector<double> getJointMaxAcceleration();
00265
00266     /**
00267     * @ingroup SystemApi
00268     * \chinese
00269     * 获取 TCP 默认速度
00270     * @return TCP 默认速度。
00271     * \endchinese
00272     * \english
00273     * Returns the default velocity of TCP.
00274     * @return the default velocity of TCP.
00275     * \endenglish
00276     */
00277     double getTcpDefaultVelocity();
00278
00279     /**
00280     * @ingroup SystemApi
00281     * \chinese
00282     * 获取 TCP 默认加速度
00283     * @return TCP 默认加速度。
00284     * \endchinese
00285     * \english
00286     * Returns the default acceleration of TCP.
00287     * @return the default acceleration of TCP.
00288     * \endenglish
00289     */
00290     double getTcpDefaultAcceleration();
00291
00292     /**
00293     * @ingroup SystemApi
00294     * \chinese
00295     * 获取 TCP 最大速度
00296     * @return TCP 最大速度。

```

```

00297     * \endchinese
00298     * \english
00299     * Returns the max velocity of TCP.
00300     * @return the max velocity of TCP.
00301     * \endenglish
00302     */
00303 double getTcpMaxVelocity();
00304
00305 /**
00306  * @ingroup SystemApi
00307  * \chinese
00308  * 获取 TCP 最大加速度
00309  * @return TCP 最大加速度。
00310  * \endchinese
00311  * \english
00312  * Returns the max acceleration of TCP.
00313  * @return the max acceleration of TCP.
00314  * \endenglish
00315  */
00316 double getTcpMaxAcceleration();
00317
00318 /**
00319  * @ingroup SystemApi
00320  * \chinese
00321  * 获取 TCP 最大移动距离
00322  * @return TCP 最大距离。
00323  * \endchinese
00324  * \english
00325  * Returns the max distance of TCP.
00326  * @return the max distance of TCP.
00327  * \endenglish
00328  */
00329 double getTcpMaxDistance();
00330
00331 /**
00332  * @ingroup SystemApi
00333  * \chinese
00334  * 获取最大负载质量
00335  * @return 最大负载质量。
00336  * \endchinese
00337  * \english
00338  * Returns the max payload mass.
00339  * @return the max payload mass.
00340  * \endenglish
00341  */
00342 double getMaxPayloadMass();
00343
00344 /**
00345  * @ingroup SystemApi
00346  * \chinese
00347  * 获取最大负载距离
00348  * @return 最大负载距离。
00349  * \endchinese
00350  * \english
00351  * Returns the max payload distance.
00352  * @return the max payload distance.
00353  * \endenglish
00354  */
00355 double getMaxPayloadDistance();
00356
00357 /**
00358  * @ingroup SystemApi
00359  * \chinese
00360  * 获取默认 Home 位置
00361  * @return 默认 Home 位置关节角度。
00362  * \endchinese
00363  * \english
00364  * Returns the default home position.
00365  * @return the default home position.
00366  * \endenglish
00367  */
00368 std::vector<double> getHomeDefaultPosition();
00369
00370 /**
00371  * @ingroup SystemApi
00372  * \chinese
00373  * 获取包装 Home 位置
00374  * @return 包装 Home 位置关节角度。
00375  * \endchinese
00376  * \english
00377  * Returns the package home position.
00378  * @return the package home position.
00379  * \endenglish
00380  */
00381 std::vector<double> getHomePackagePosition();
00382
00383 /**

```

```

00384     * @ingroup SystemApi
00385     * \chinese
00386     * 获取机器人 DH 参数
00387     * @param is_real 是否获取真实参数。
00388     * @return DH 参数表。
00389     * \endchinese
00390     * \english
00391     * Returns the robot DH parameters.
00392     * @param is_real whether to get the real parameters.
00393     * @return the DH parameters.
00394     * \endenglish
00395     */
00396     std::unordered_map<std::string, std::vector<double>> getRobotDhParams(
00397         bool is_real);
00398
00399     /**
00400     * @ingroup SystemApi
00401     * \chinese
00402     * 获取工具端数字 IO 配置
00403     * @return 工具端数字 IO 配置。
00404     * \endchinese
00405     * \english
00406     * Returns the tool digital I/O configuration.
00407     * @return the tool digital I/O configuration.
00408     * \endenglish
00409     */
00410     IoConfigure getToolDigitalIOConfigure();
00411
00412     /**
00413     * @ingroup SystemApi
00414     * \chinese
00415     * 获取标准数字输入配置
00416     * @return 标准数字输入配置。
00417     * \endchinese
00418     * \english
00419     * Returns the standard digital input configuration.
00420     * @return the standard digital input configuration.
00421     * \endenglish
00422     */
00423     IoConfigure getStandardDigitalInputConfigure();
00424
00425     /**
00426     * @ingroup SystemApi
00427     * \chinese
00428     * 获取可配置数字输入配置
00429     * @return 可配置数字输入配置。
00430     * \endchinese
00431     * \english
00432     * Returns the configurable digital input configuration.
00433     * @return the configurable digital input configuration.
00434     * \endenglish
00435     */
00436     IoConfigure getConfigurableDigitalInputConfigure();
00437
00438     /**
00439     * @ingroup SystemApi
00440     * \chinese
00441     * 获取标准数字输出配置
00442     * @return 标准数字输出配置。
00443     * \endchinese
00444     * \english
00445     * Returns the standard digital output configuration.
00446     * @return the standard digital output configuration.
00447     * \endenglish
00448     */
00449     IoConfigure getStandardDigitalOutputConfigure();
00450
00451     /**
00452     * @ingroup SystemApi
00453     * \chinese
00454     * 获取可配置数字输出配置
00455     * @return 可配置数字输出配置。
00456     * \endchinese
00457     * \english
00458     * Returns the configurable digital output configuration.
00459     * @return the configurable digital output configuration.
00460     * \endenglish
00461     */
00462     IoConfigure getConfigurableDigitalOutputConfigure();
00463
00464     /**
00465     * @ingroup SystemApi
00466     * \chinese
00467     * 获取标准模拟输入配置
00468     * @return 标准模拟输入配置。
00469     * \endchinese
00470     * \english

```

```

00471     * Returns the standard analog input configuration.
00472     * @return the standard analog input configuration.
00473     * \endenglish
00474     */
00475 IoConfigure getStandardAnalogInputConfigure();
00476
00477 /**
00478  * @ingroup SystemApi
00479  * \chinese
00480  * 获取工具端模拟输入配置
00481  * @return 工具端模拟输入配置。
00482  * \endchinese
00483  * \english
00484  * Returns the tool analog input configuration.
00485  * @return the tool analog input configuration.
00486  * \endenglish
00487  */
00488 IoConfigure getToolAnalogInputConfigure();
00489
00490 /**
00491  * @ingroup SystemApi
00492  * \chinese
00493  * 获取标准模拟输出配置
00494  * @return 标准模拟输出配置。
00495  * \endchinese
00496  * \english
00497  * Returns the standard analog output configuration.
00498  * @return the standard analog output configuration.
00499  * \endenglish
00500  */
00501 IoConfigure getStandardAnalogOutputConfigure();
00502
00503 /**
00504  * @ingroup SystemApi
00505  * \chinese
00506  * 获取工具端模拟输出配置
00507  * @return 工具端模拟输出配置。
00508  * \endchinese
00509  * \english
00510  * Returns the tool analog output configuration.
00511  * @return the tool analog output configuration.
00512  * \endenglish
00513  */
00514 IoConfigure getToolAnalogOutputConfigure();
00515
00516 /**
00517  * @ingroup SystemApi
00518  * \chinese
00519  * 获取静态链路输入配置
00520  * @return 静态链路输入配置。
00521  * \endchinese
00522  * \english
00523  * Returns the static link input configuration.
00524  * @return the static link input configuration.
00525  * \endenglish
00526  */
00527 IoConfigure getStaticLinkInputConfigure();
00528
00529 /**
00530  * @ingroup SystemApi
00531  * \chinese
00532  * 获取静态链路输出配置
00533  * @return 静态链路输出配置。
00534  * \endchinese
00535  * \english
00536  * Returns the static link output configuration.
00537  * @return the static link output configuration.
00538  * \endenglish
00539  */
00540 IoConfigure getStaticLinkOutputConfigure();
00541
00542 /**
00543  * @ingroup SystemApi
00544  * \chinese
00545  * 获取编码器配置
00546  * @return 编码器配置。
00547  * \endchinese
00548  * \english
00549  * Returns the encoder configuration.
00550  * @return the encoder configuration.
00551  * \endenglish
00552  */
00553 IoConfigure getENCCConfigure();
00554
00555 private:
00556     friend class DataSwitch;
00557     SystemApi();

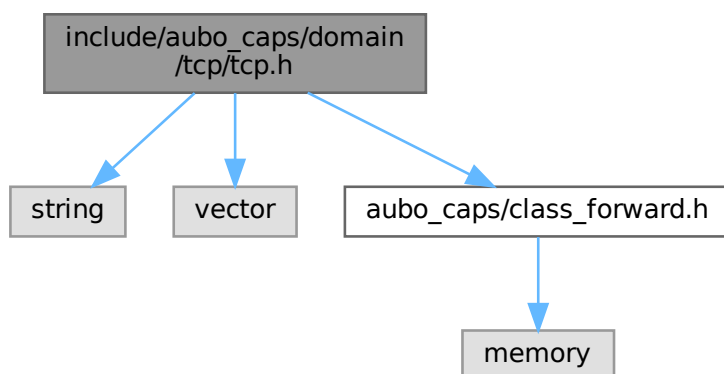
```

```
00558     void *d_{ nullptr };
00559 };
00560
00561 } // namespace aubo_scope
00562 } // namespace arcs
00563
00564 #endif // SYSTEMAPI_H
```

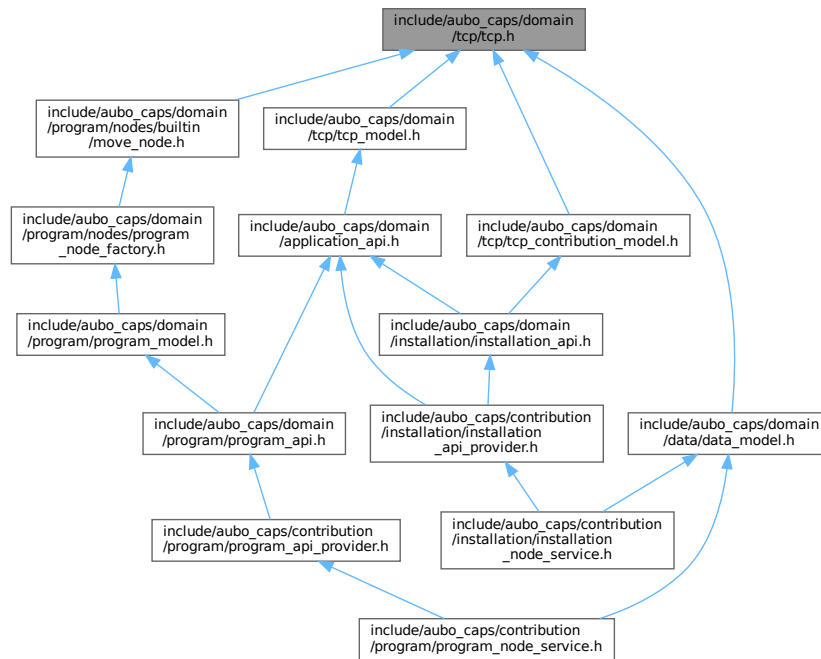
12.214 include/aubo_caps/domain/tcp/tcp.h 文件参考

```
#include <string>
#include <vector>
#include <aubo_caps/class_forward.h>
```

tcp.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class [arcs::aubo_scope::TCP](#)

[TCP](#) 接口此接口表示 *AuboScope* 中的一个 *TCP*（工具中心点）。可以是用户定义的 *TCP*，也可以是由 *AuboCap* 添加的 *TCP*。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(TCP\)](#)

12.215 tcp.h

[浏览该文件的文档.](#)

```
00001 #ifndef AUBO_SCOPE_TCP_H
00002 #define AUBO_SCOPE_TCP_H
00003
00004 #include <string>
00005 #include <vector>
00006 #include <aubo_caps/class_forward.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(TCP);
00011
00012 /**
00013  * @ingroup InstallationApi
00014  * \chinese
00015  * TCP 接口
00016  * 此接口表示 AuboScope 中的一个 TCP（工具中心点）。可以是用户定义的 TCP，也可以
00017  * 是由 AuboCap 添加的 TCP。
00018  * \endchinese
00019  * \english
```

```

00020 * TCP
00021 * This interface represents a TCP in AuboScope. This can either be a
00022 * user-defined TCP or a TCP added by a AuboCap.
00023 * \endenglish
00024 */
00025 class ARCS_ABI_EXPORT TCP
00026 {
00027 public:
00028     TCP(TCP &f);
00029     TCP(TCP &&f);
00030     ~TCP();
00031
00032     /**
00033      * @ingroup TCP
00034      * \chinese
00035      * 获取 TCP 相对于工具输出法兰的偏移量
00036      * @return TCP 偏移量。
00037      * \endchinese
00038      * \english
00039      * Returns the offset of the TCP with respect to the tool output flange.
00040      * @return The offset of the TCP.
00041      * \endenglish
00042      */
00043     std::vector<double> getOffset();
00044
00045     /**
00046      * @ingroup TCP
00047      * \chinese
00048      * 获取 TCP 显示名称。注意: TCP 可时被重命名。
00049      * @return TCP 名称。
00050      * \endchinese
00051      * \english
00052      * Note: The TCP can be renamed at any time.
00053      * @return The name of the TCP.
00054      * \endenglish
00055      */
00056     std::string getDisplayName();
00057
00058     /**
00059      * @ingroup TCP
00060      * \chinese
00061      * TCP 无法保证始终存在。终端用户可加载不包含该 TCP 的其他安装, 或该 TCP 可被终端用户
00062      * 或添加该 TCP 的 AuboCap 删除。此方法用于判断 TCP 是否存在。
00063      * @return 如果 TCP 存在返回 <code>true</code>, 否则返回 <code>false</code>。
00064      * \endchinese
00065      * \english
00066      * A TCP cannot be guaranteed to be present in AuboScope. The end user can
00067      * load a different installation (not containing the TCP) or the TCP was
00068      * removed by the end user or the AuboCap that added the tcp. This method
00069      * can be used to determine, if the TCP is present.
00070      *
00071      * @return <code>true</code> if this TCP is present in AuboScope, otherwise
00072      * <code>false</code>.
00073      * \endenglish
00074      */
00075     bool isResolvable();
00076
00077     /**
00078      * @ingroup TCP
00079      * \chinese
00080      * 获取哈希码
00081      * @return 哈希码。
00082      * \endchinese
00083      * \english
00084      * Returns the hash code.
00085      * @return the hash code.
00086      * \endenglish
00087      */
00088     size_t hashCode();
00089
00090 private:
00091     friend class DataSwitch;
00092     TCP();
00093     void *d_{ nullptr };
00094 };
00095
00096 } // namespace aubo_scope
00097 } // namespace arcs
00098
00099 #endif // AUBO_SCOPE_TCP_H

```

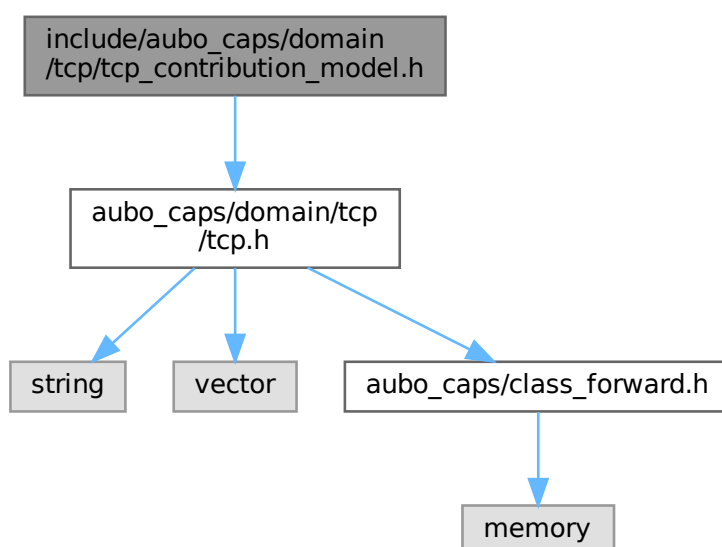

12.216.

INCLUDE/AUBO_CAPS/DOMAIN/TCP/TCP_CONTRIBUTION_MODEL.H 文件参
考 857

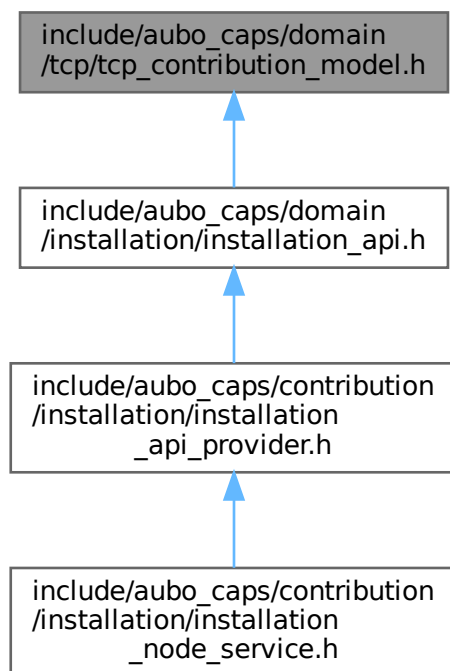
12.216 include/aubo_caps/domain/tcp/tcp_contribution_
model.h 文件参考

```
#include <aubo_caps/domain/tcp/tcp.h>
```

tcp_contribution_model.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::TcpContributionModel`
TCP 贡献模型此接口提供在 *AuboScope* 中添加、更新和删除 *TCP* 的功能。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (TcpContributionModel)`

12.217 tcp_contribution_model.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_TCP_CONTRIBUTION_MODEL_H
00002 #define AUBO_SCOPE_TCP_CONTRIBUTION_MODEL_H
00003
00004 #include <aubo_caps/domain/tcp/tcp.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(TcpContributionModel);
00009
00010 /**
00011 * @ingroup InstallationApi
00012 * \chinese
00013 * TCP 贡献模型
00014 * 此接口提供在 AuboScope 中添加、更新和删除 TCP 的功能。
00015 *

```

```

00016 * <b> 注意: </b> 此接口只能在安装贡献中使用。
00017 * \endchinese
00018 * \english
00019 * TcpContributionModel
00020 * This interface provides functionality for adding, updating and removing TCPs
00021 * in AuboScope.
00022 *
00023 * <b>Note: </b> This interface must only be used in an installation
00024 * contribution.
00025 * \endenglish
00026 */
00027 class ARCS_ABI_EXPORT TcpContributionModel
00028 {
00029 public:
00030     TcpContributionModel(TcpContributionModel &f);
00031     TcpContributionModel(TcpContributionModel &&f);
00032     virtual ~TcpContributionModel();
00033
00034     /**
00035      * @ingroup TcpContributionModel
00036      * @ref TCP
00037      * \chinese
00038      * 向当前 AuboScope 安装中添加一个 TCP。添加后终端用户可选择该 TCP，但不可修改。
00039      *
00040      * @param idKey 用于标识此 TCP 的键。该键仅对此 AuboCap 有效。
00041      * @param suggested_name 建议的 TCP 名称。
00042      * @param offset TCP 相对于工具输出法兰的偏移量。
00043      * @return 在 AuboScope 中创建并注册的 TCP。
00044      * @throws TCPAlreadyAddedException 如果已使用相同 idKey 添加过 TCP。
00045      * @throws IllegalTCPNameException 如果建议的名称不匹配所需正则表达式。
00046      * @throws IllegalArgumentException 如果偏移量超出有效范围。
00047      * \endchinese
00048      * \english
00049      * Add a TCP to the current AuboScope installation. This makes it selectable
00050      * by the end user. The TCP is not modifiable by the end user.
00051      *
00052      * @param idKey The key to identify this TCP by.
00053      * @param suggested_name Suggested name for the TCP.
00054      * @param offset The offset of the TCP with respect to the tool output flange.
00055      * @return The TCP created and registered in AuboScope.
00056      * @throws TCPAlreadyAddedException If called twice using the same idKey.
00057      * @throws IllegalTCPNameException If the suggested name does not match regex.
00058      * @throws IllegalArgumentException If the offset is invalid.
00059      * \endenglish
00060      */
00061     TCPPtr addTcp(const std::string &idKey, const std::string &suggested_name,
00062                  const std::vector<double> &offset);
00063
00064     /**
00065      * @ingroup TcpContributionModel
00066      * @ref TCP
00067      * \chinese
00068      * 返回此前由此 AuboCap 使用相同 idKey 添加的 TCP。
00069      * @param idKey TCP 标识符键。
00070      * @return 此前添加的 TCP，如果不存在则返回 <code>null</code>。
00071      * \endchinese
00072      * \english
00073      * Returns the TCP previously added by this AuboCap using the same idKey.
00074      * @param idKey The key to identify this TCP by.
00075      * @return The TCP previously added by this AuboCap.
00076      * \endenglish
00077      */
00078     TCPPtr getTcp(const std::string &idKey);
00079
00080     /**
00081      * @ingroup TcpContributionModel
00082      * \chinese
00083      * 更新此 AuboCap 已添加 TCP 的偏移量。
00084      * @param idKey TCP 标识符键。
00085      * @param newOffset TCP 的新偏移量。
00086      * @throws TCPNotFoundExpection 如果不存在对应 idKey 的 TCP。
00087      * @throws IllegalArgumentException 如果偏移量超出有效范围。
00088      * \endchinese
00089      * \english
00090      * Update the offset of a TCP already added by this AuboCap.
00091      * @param idKey The key to identify the TCP.
00092      * @param newOffset The new offset to set for the TCP.
00093      * @throws TCPNotFoundExpection If no TCP exists with the provided idKey.
00094      * @throws IllegalArgumentException If the offset is invalid.
00095      * \endenglish
00096      */
00097     void updateTcp(const std::string &idKey,
00098                  const std::vector<double> &newOffset);
00099
00100     /**
00101      * @ingroup TcpContributionModel
00102      * \chinese

```

```

00103      * 从此 AuboCap 中移除已添加的 TCP。使用该 TCP 的程序节点将变为未定义。
00104      * @param idKey 添加 TCP 时使用的键。
00105      * @throws TCPNotFoundException 如果不存在对应 idKey 的 TCP。
00106      * \endchinese
00107      * \english
00108      * Remove a TCP added by this AuboCap from AuboScope.
00109      * @param idKey The key used to add the TCP with.
00110      * @throws TCPNotFoundException If no TCP exists with the provided idKey.
00111      * \endenglish
00112      */
00113      void removeTcp(const std::string &idKey);
00114
00115  private:
00116      friend class DataSwitch;
00117      TcpContributionModel();
00118      void *d_{ nullptr };
00119 };
00120
00121 } // namespace aubo_scope
00122 } // namespace arcs
00123
00124 #endif // TCPCONTRIBUTIONMODEL_H

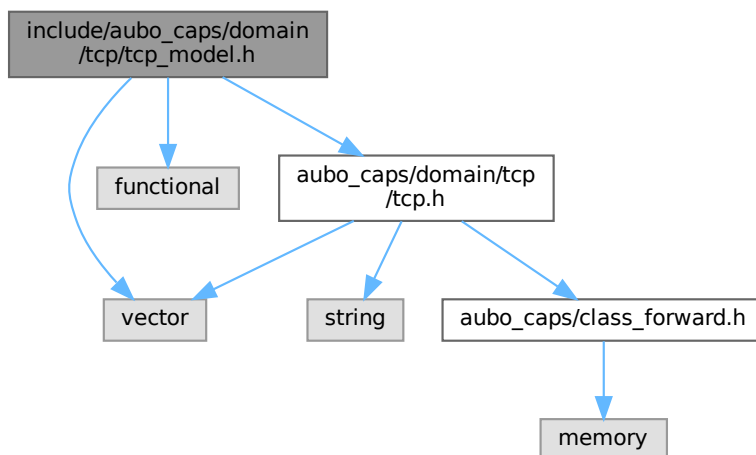
```

12.218 include/aubo_caps/domain/tcp/tcp_model.h 文件参考

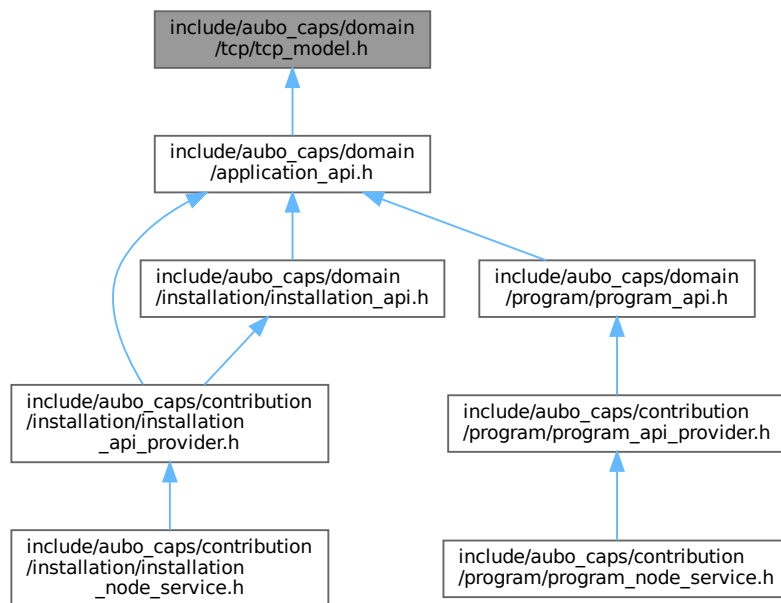
```

#include <vector>
#include <functional>
#include <aubo_caps/domain/tcp/tcp.h>
tcp_model.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::TcpModel`
TCP 模型此接口提供对 *AuboScope* 中当前可用的 *TCP* 的访问。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (TcpModel)`

12.219 tcp_model.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOP_TCP_MODEL_H
00002 #define AUBO_SCOP_TCP_MODEL_H
00003
00004 #include <vector>
00005 #include <functional>
00006 #include <aubo_caps/domain/tcp/tcp.h>
00007
00008 namespace arcs {
00009     namespace aubo_scope {
00010         ARCS_CLASS_FORWARD(TcpModel);
00011     }
00012 /**
00013  * @ingroup ApplicationApi
00014  * \chinese
00015  * TCP 模型
00016  * 此接口提供对 AuboScope 中当前可用的 TCP 的访问。
00017  * \endchinese
00018  * \english
00019  * TcpModel
00020  * This interface provides access to the TCPs that are currently available in
00021  * AuboScope.
  
```

```

00022 * \endenglish
00023 */
00024 class ARCS_ABI_EXPORT TcpModel
00025 {
00026 public:
00027     TcpModel(TcpModel &f);
00028     TcpModel(TcpModel &&f);
00029     virtual ~TcpModel();
00030
00031     /**
00032      * @ingroup TcpModel
00033      * \chinese
00034      * 获取 AboScope 中所有 TCP 的集合
00035      * @return 所有 TCP 的集合。
00036      * \endchinese
00037      * \english
00038      * Returns a collection of all the TCPs in AboScope.
00039      * @return A collection of all the TCPs in AboScope.
00040      * \endenglish
00041      */
00042     std::vector<TCPPtr> getAll();
00043
00044     /**
00045      * @ingroup TcpModel
00046      * @ref TCP
00047      * \chinese
00048      * 根据名称获取 TCP
00049      * @param name TCP 名称。
00050      * @return 对应名称的 TCP。
00051      * \endchinese
00052      * \english
00053      * Get a TCP by name.
00054      * @param name the name of the TCP.
00055      * @return the TCP with the given name.
00056      * \endenglish
00057      */
00058     TCPPtr getTcp(const std::string &name);
00059
00060     /**
00061      * @ingroup TcpModel
00062      * @ref TCP
00063      * \chinese
00064      * 获取默认 TCP
00065      * @return 默认 TCP。
00066      * \endchinese
00067      * \english
00068      * Returns the default TCP.
00069      * @return the default TCP.
00070      * \endenglish
00071      */
00072     TCPPtr getDefaultTcp();
00073
00074     /**
00075      * @ingroup TcpModel
00076      * \chinese
00077      * 注册变更回调
00078      * @param receiver 接收者标识。
00079      * @param slot 回调函数。
00080      * @return 连接标识符。
00081      * \endchinese
00082      * \english
00083      * Register callback for changed.
00084      * @param receiver the receiver identifier.
00085      * @param slot the callback function.
00086      * @return the connection identifier.
00087      * \endenglish
00088      */
00089     std::string connectedTo(const std::string &receiver,
00090                             const std::function<void(TcpModel *)> &slot);
00091
00092     /**
00093      * @ingroup TcpModel
00094      * \chinese
00095      * 断开回调连接
00096      * @param receiver 接收者标识。
00097      * \endchinese
00098      * \english
00099      * Disconnect a callback.
00100      * @param receiver the receiver identifier.
00101      * \endenglish
00102      */
00102     void disconnectFrom(const std::string &receiver);
00103
00104 private:
00105     friend class DataSwitch;
00106     TcpModel();
00107     void *d_{ nullptr };
00108 };

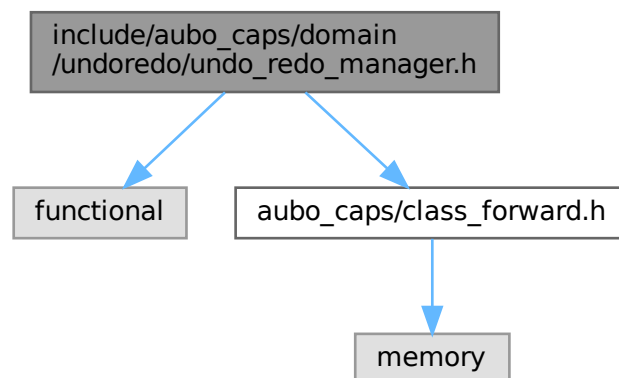
```

```
00109
00110 } // namespace aubo_scope
00111 } // namespace arcs
00112
00113 #endif // AUBO_SCOP_TCP_MODEL_H
```

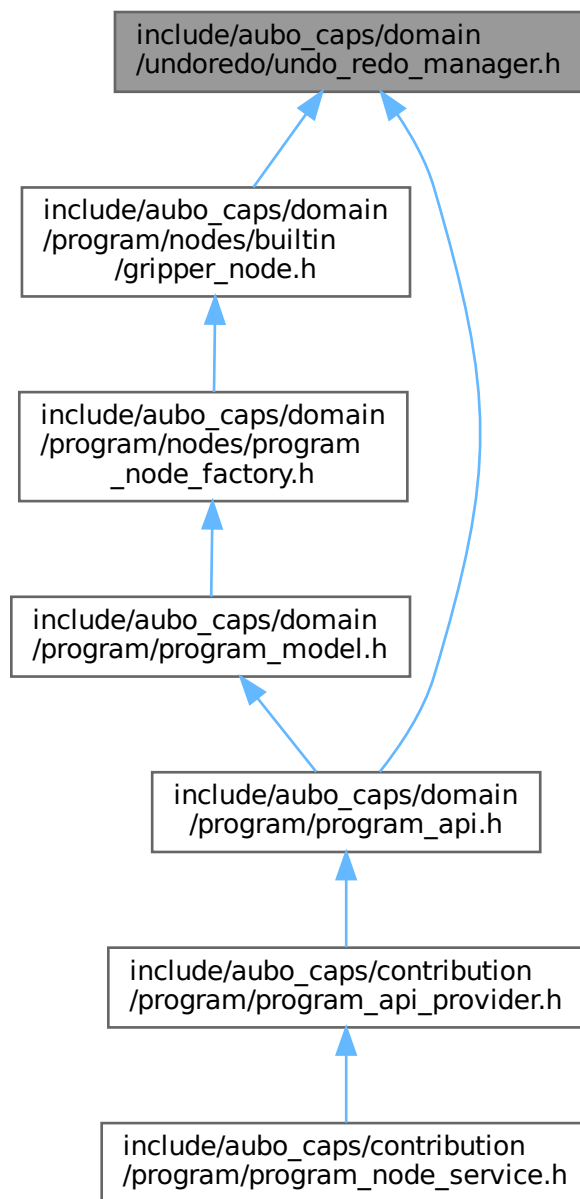
12.220 include/aubo_caps/domain/undoredo/undo_redo_manager.h 文件参考

```
#include <functional>
#include <aubo_caps/class_forward.h>
```

undo_redo_manager.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class `arcs::aubo_scope::UndoRedoManager`

撤销/重做管理器此接口用于记录终端用户在 *AuboCap* 程序节点贡献画面中触发的更改（例如，点击按钮时）。应在 *AuboCap* 程序节点中使用，以支持 *AuboScope* 中的撤销/重做功能。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (UndoRedoManager)`

12.221 undo_redo_manager.h

[浏览该文件的文档.](#)

```

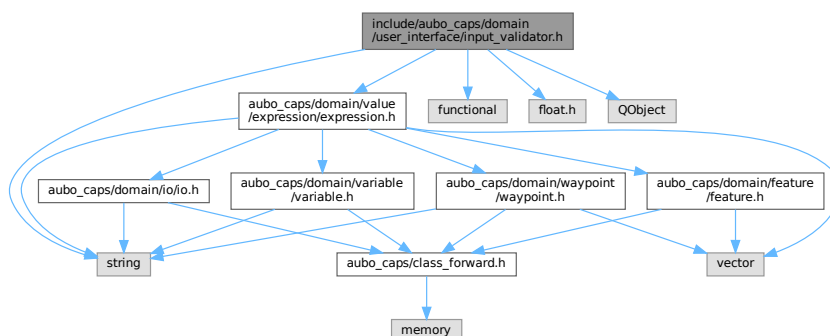
00001 #ifndef AUBO_SCOPE_UNDO_REDO_MANAGER_H
00002 #define AUBO_SCOPE_UNDO_REDO_MANAGER_H
00003
00004 #include <functional>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008     namespace aubo_scope {
00009         ARCS_CLASS_FORWARD(UndoRedoManager);
00010
00011         /**
00012          * @ingroup ProgramApi
00013          * \chinese
00014          * 撤销/重做管理器
00015          * 此接口用于记录终端用户在 AuboCap 程序节点贡献画面中触发的更改（例如，点击按钮时）。
00016          * 应在 AuboCap 程序节点中使用，以支持 AuboScope 中的撤销/重做功能。
00017          *
00018          * 注意：请勿对 AuboCap 安装节点使用此功能，因为它不受支持（且无效）。<br>
00019          * 注意：此功能不适用于具有基于 HTML 的用户界面的 AuboCap 程序节点，因为这些节点
00020          * 已支持自动撤销。
00021          * \endchinese
00022          * \english
00023          * UndoRedoManager
00024          * This interface is used to record changes triggered by an end user in the
00025          * screen of a AuboCap program node contribution (e.g. when a button is
00026          * clicked). It should be used in a AuboCap program node to support the
00027          * Undo/Redo functionality in AuboScope.
00028          *
00029          * NOTE: Do not use this functionality for AuboCap installation nodes, since it
00030          * is not supported (and has no effect). <br>
00031          * NOTE: This functionality is not relevant for AuboCap program nodes with a
00032          * HTML-based user interface, since automatic undo is supported for these.
00033          * \endenglish
00034          */
00035         class ARCS_ABI_EXPORT UndoRedoManager
00036         {
00037         public:
00038             UndoRedoManager(UndoRedoManager &f);
00039             UndoRedoManager(UndoRedoManager &&f);
00040             virtual ~UndoRedoManager();
00041
00042             /**
00043              * \chinese
00044              * <p>
00045              * 将一组变更归并为一个撤销/重做步骤。
00046              * </p>
00047              * 目前撤销/重做支持程序树和数据模型的更改。
00048              *
00049              * @param undoableChanges 被记录为一个撤销/重做步骤的代码块。在 AuboScope 中
00050              * 按下" 撤销" 将撤销此代码块内对程序树和数据模型所做的所有更改。
00051              * \endchinese
00052              * \english
00053              * <p>
00054              * Group changes into one Undo/Redo step.
00055              * </p>
00056              *
00057              * Currently Undo/Redo supports changes to the program tree and the data
00058              * model.
00059              *
00060              * @param undoableChanges Code to be recorded as one Undo/Redo step.
00061              * Pressing Undo in AuboScope will undo all changes
00062              * done to the program tree and the data model inside this block of code.
00063              * \endenglish
00064              */
00065             void recordChanges(const std::string &text,
00066                             std::function<void()> undoableChanges);
00067
00068         private:
00069             friend class DataSwitch;
00070             UndoRedoManager();
00071             void *d_{ nullptr };
00072         };
00073     } // namespace aubo_scope
00074 } // namespace arcs
00075
00076 #endif

```

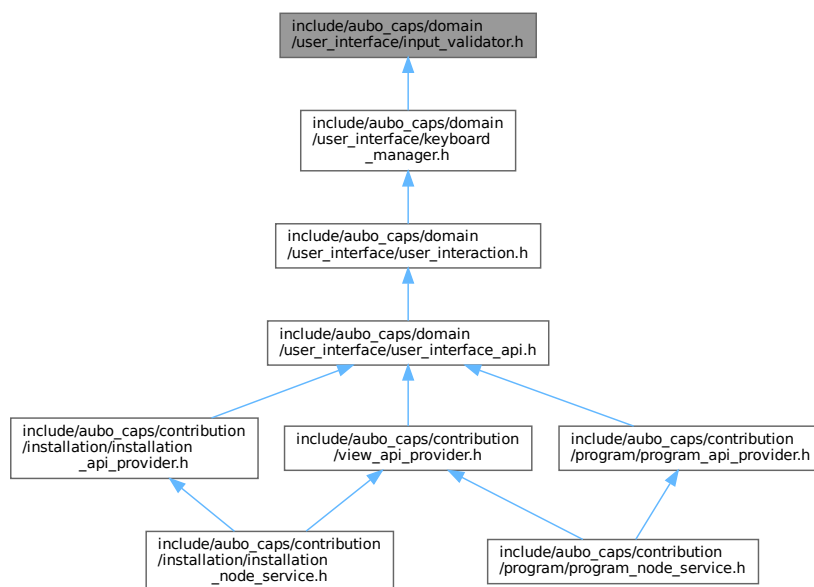
12.222 include/aubo_caps/domain/user_interface/input_validator.h 文件参考

```
#include <string>
#include <functional>
#include <float.h>
#include <QObject>
#include <aubo_caps/domain/value/expression/expression.h>
```

input_validator.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- struct [arcs::aubo_scope::InputValidator](#)

输入验证器表示由 *InputValidationFactory* 创建的输入验证器接口。该工厂提供一组标准输入验证器，可用于验证用户使用虚拟键盘/小键盘输入的内容。

- struct [arcs::aubo_scope::IntegerRangeValidator](#)
- struct [arcs::aubo_scope::UIntegerRangeValidator](#)

- struct `arcs::aubo_scope::DoubleRangeValidator`
- struct `arcs::aubo_scope::StringLengthValidator`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

12.223 input_validator.h

[浏览该文件的文档.](#)

```
00001 #ifndef AUBO_SCOPE_KEYBOARD_INPUT_VALIDATOR_H
00002 #define AUBO_SCOPE_KEYBOARD_INPUT_VALIDATOR_H
00003
00004 #include <string>
00005 #include <functional>
00006 #include <float.h>
00007
00008 #include <QObject>
00009
00010 #include <aubo_caps/domain/value/expression/expression.h>
00011
00012 namespace arcs {
00013 namespace aubo_scope {
00014
00015 /**
00016  * \chinese
00017  * 输入验证器
00018  * 表示由 {@link InputValidationFactory} 创建的输入验证器接口。该工厂提供一组标准
00019  * 输入验证器，可用于验证用户使用虚拟键盘/小键盘输入的内容。
00020  *
00021  * 如果 {@link InputValidationFactory} 中可用的标准验证器不能满足您的需求，您可以
00022  * 自由实现自己的自定义验证器。
00023  *
00024  * @param <T> 表示正在验证的输入数据类型的接口（通用）类型参数（例如 Integer 或 Double）。
00025  * \endchinese
00026  * \english
00027  * InputValidator
00028  * Interface representing the input validators created by {@link
00029  * InputValidationFactory}. This factory provides a set of standard input
00030  * validators that can be used for validating the input entered by the user
00031  * using the virtual keyboard/keypad.
00032  *
00033  * If the standard validators available in the {@link InputValidationFactory}
00034  * does not satisfy your needs, you are free to implement your own custom
00035  * validator.
00036  *
00037  * @param <T> the (generic) type parameter for the interface representing the
00038  * type of input data being validated (e.g. Integer or Double).
00039  * \endenglish
00040  */
00041 struct InputValidator
00042 {
00043     /**
00044      * \chinese
00045      * 验证输入值是否有效。
00046      *
00047      * @param value 需要验证的值
00048      * @return 如果值有效则返回 <code>true</code>
00049      * \endchinese
00050      * \english
00051      * Validates whether the input value is valid.
00052      *
00053      * @param value to be validated.
00054      * @return <code>true</code> if value is valid.
00055      * \endenglish
00056      */
00057     virtual bool isValid(std::string value) = 0;
00058
00059     /**
00060      * \chinese
00061      * 当值无效时返回有意义的提示信息。
00062      *
00063      * @param value 无效的值，如果合理可包含在消息中
00064      * @return 提示信息
00065      * \endchinese
00066      * \english
00067      * Returns a meaningful message in case the value is not valid.
00068      *
00069      * @param value the invalid value. Can be included in the message if it
00070      * makes sense.
```

```

00071     * @return message.
00072     * \endenglish
00073     */
00074     virtual std::string getMessage(std::string value) = 0;
00075 };
00076
00077 struct IntegerRangeValidator : InputValidator
00078 {
00079     IntegerRangeValidator(int min_value, int max_value);
00080
00081     IntegerRangeValidator(std::function<std::pair<int, int>()> range_func);
00082
00083     bool isValid(std::string value) override;
00084
00085     std::string getMessage(std::string value) override;
00086
00087 private:
00088     void updateValue();
00089
00090 private:
00091     int min_value_{ INT_MIN };
00092     int max_value_{ INT_MAX };
00093
00094     std::function<std::pair<int, int>()> range_func_{ nullptr };
00095 };
00096
00097 struct UIntegerRangeValidator : InputValidator
00098 {
00099     UIntegerRangeValidator(uint32_t min_value, uint32_t max_value);
00100
00101     UIntegerRangeValidator(
00102         std::function<std::pair<uint32_t, uint32_t>()> range_func);
00103
00104     bool isValid(std::string value) override;
00105
00106     std::string getMessage(std::string value) override;
00107
00108 private:
00109     void updateValue();
00110
00111 private:
00112     uint32_t min_value_{ 0 };
00113     uint32_t max_value_{ UINT_MAX };
00114
00115     std::function<std::pair<uint32_t, uint32_t>()> range_func_{ nullptr };
00116 };
00117
00118 struct DoubleRangeValidator : InputValidator
00119 {
00120     DoubleRangeValidator(double min_value, double max_value,
00121         bool is_min_inclusive = true,
00122         bool is_max_inclusive = true);
00123
00124     DoubleRangeValidator(std::function<std::pair<double, double>()> range_func);
00125
00126     DoubleRangeValidator(
00127         std::function<std::tuple<double, double, bool, bool>()> range_func);
00128
00129     bool isValid(std::string value) override;
00130
00131     std::string getMessage(std::string value) override;
00132
00133 private:
00134     void updateValue();
00135
00136 private:
00137     double min_value_{ DBL_MIN };
00138     double max_value_{ DBL_MAX };
00139     std::function<std::pair<double, double>()> range_func_{ nullptr };
00140     std::function<std::tuple<double, double, bool, bool>()>
00141         range_func_with_flags_{ nullptr };
00142     bool is_min_inclusive_{ true }; // 最小值是否闭区间
00143     bool is_max_inclusive_{ true }; // 最大值是否闭区间
00144 };
00145
00146 struct StringLengthValidator : InputValidator
00147 {
00148     StringLengthValidator(size_t min_length, size_t max_length);
00149
00150     StringLengthValidator(
00151         std::function<std::pair<size_t, size_t>()> range_func);
00152
00153     bool isValid(std::string value) override;
00154
00155     std::string getMessage(std::string value) override;
00156
00157 private:

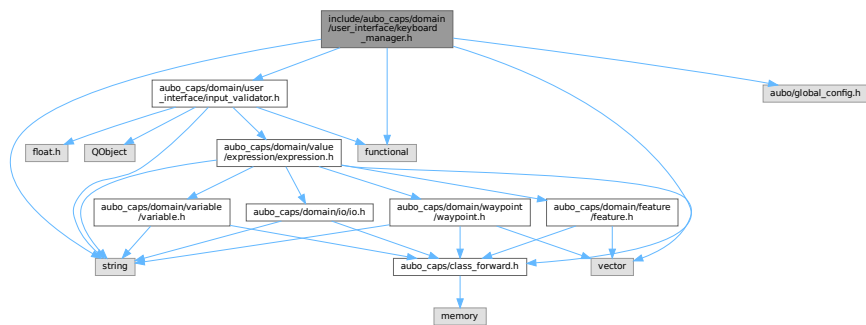
```

```
00158     void updateValue();
00159
00160 private:
00161     size_t min_length_{ 0 };
00162     size_t max_length_{ ULONG_MAX };
00163     std::function<std::pair<size_t, size_t>()> range_func_{ nullptr };
00164 };
00165
00166 } // namespace aubo_scope
00167 } // namespace arcs
00168
00169 #endif
```

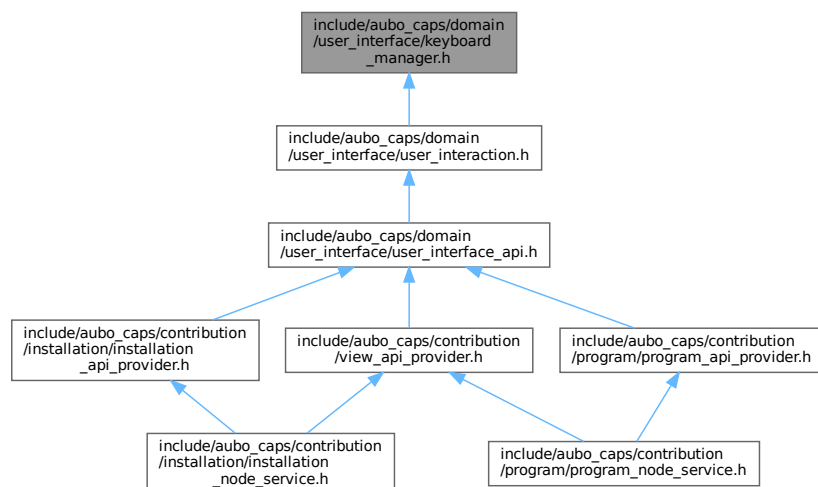
12.224 include/auco_caps/domain/user_interface/keyboard_↵ __manager.h 文件参考

```
#include <string>
#include <functional>
#include <auco_caps/class_forward.h>
#include <auco_caps/domain/user_interface/input_validator.h>
#include <auco/global_config.h>
```

keyboard_manager.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- `class arcs::aubo_scope::KeyboardManager`

命名空间

- `namespace arcs`
- `namespace arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (KeyboardManager)`
- `arcs::aubo_scope::ARCS_CLASS_FORWARD (Expression)`

12.225 keyboard_manager.h

浏览该文件的文档.

```

00001 #ifndef AUBO_CAPS_INTERFACE_KEYBOARD_MANAGER_H
00002 #define AUBO_CAPS_INTERFACE_KEYBOARD_MANAGER_H
00003
00004 #include <string>
00005 #include <functional>
00006
00007 #include <aubo_caps/class_forward.h>
00008 #include <aubo_caps/domain/user_interface/input_validator.h>
00009 #include <aubo/global_config.h>
00010 class QWidget;
00011
00012 namespace arcs {
00013 namespace aubo_scope {
00014
00015 ARCS_CLASS_FORWARD(KeyboardManager);
00016 ARCS_CLASS_FORWARD(Expression);
00017
00018 class ARCS_ABI_EXPORT KeyboardManager
00019 {
00020 public:
00021     KeyboardManager(KeyboardManager &f);
00022     KeyboardManager(KeyboardManager &&f);
00023     virtual ~KeyboardManager();
00024
00025     /**
00026      * @brief registerStandardKeyboard 注册标准键盘
00027      * @param wd: 当前要书写的控件, 比如 ui->lineedit
00028      * @param init_cb: 初始 edit 自带的字符串输出到键盘
00029      * @param validator: 键盘的过滤器
00030      * @param result_cb: 把当前键盘中的值发送到 edit
00031      */
00032     ARCS_DEPRECATED void registerStandardKeyboard(
00033         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00034         const std::shared_ptr<InputValidator> &validator = nullptr,
00035         std::function<void(std::string)> result_cb = nullptr);
00036     /**
00037      * @brief registerStandardKeyboard
00038      * @param wd: 当前要书写的控件, 比如 ui->lineedit
00039      * @param init_cb:
00040      * 当前控件存储的内容以表达式指针的形式表示 将指针指向的内容发送到键盘
00041      * @param result_cb 将当前键盘的内容的 Expression 存储 然后已 string
00042      * 的格式显示到控件
00043      * @param validator: 键盘的过滤器
00044      */
00045     void registerStandardKeyboard(
00046         QWidget *wd, std::function<ExpressionPtr(void)> init_cb,
00047         std::function<void(ExpressionPtr)> result_cb,
00048         const std::shared_ptr<InputValidator> &validator = nullptr);
00049
00050     /// 注册字符串键盘
00051     ARCS_DEPRECATED void registerStringKeyboard(
00052         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00053         const std::shared_ptr<InputValidator> &validator = nullptr,
00054         std::function<void(std::string)> result_cb = nullptr);
00055     void registerStringKeyboard(
00056         QWidget *wd, std::function<ExpressionPtr(void)> init_cb,
00057         std::function<void(ExpressionPtr)> result_cb,
00058         const std::shared_ptr<InputValidator> &validator = nullptr);
00059
00060     /// 注册密码键盘
00061     ARCS_DEPRECATED void registerPasswordKeyboard(
00062         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,

```

```

00063     const std::shared_ptr<InputValidator> &validator = nullptr,
00064     std::function<void(std::string)> result_cb = nullptr);
00065 void registerPasswordKeyboard(
00066     QWidget *wd, std::function<ExpressionPtr(void)> init_cb,
00067     std::function<void(ExpressionPtr)> result_cb,
00068     const std::shared_ptr<InputValidator> &validator = nullptr);
00069
00070 /// 注册多行代码编辑键盘
00071 ARCS_DEPRECATED void registerCodeEditorKeyboard(
00072     QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00073     const std::shared_ptr<InputValidator> &validator = nullptr,
00074     std::function<void(std::string)> result_cb = nullptr);
00075 void registerCodeEditorKeyboard(
00076     QWidget *wd, std::function<ExpressionPtr(void)> init_cb,
00077     std::function<void(ExpressionPtr)> result_cb,
00078     const std::shared_ptr<InputValidator> &validator = nullptr);
00079
00080 void registerNumberKeyboard(
00081     QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00082     const std::shared_ptr<InputValidator> &validator = nullptr,
00083     std::function<void(std::string)> result_cb = nullptr);
00084 void registerDoubleKeyboard(
00085     QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00086     const std::shared_ptr<InputValidator> &validator = nullptr,
00087     std::function<void(std::string)> result_cb = nullptr);
00088 void registerPositiveDoubleKeyboard(
00089     QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00090     const std::shared_ptr<InputValidator> &validator = nullptr,
00091     std::function<void(std::string)> result_cb = nullptr);
00092 void registerIntegerKeyboard(
00093     QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00094     const std::shared_ptr<InputValidator> &validator = nullptr,
00095     std::function<void(std::string)> result_cb = nullptr);
00096 void registerPositiveIntegerKeyboard(
00097     QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00098     const std::shared_ptr<InputValidator> &validator = nullptr,
00099     std::function<void(std::string)> result_cb = nullptr);
00100 void registerIPAddressKeyboard(
00101     QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00102     std::function<void(std::string)> result_cb = nullptr);
00103 void registerExpressionKeyboard(
00104     QWidget *wd, std::function<ExpressionPtr(void)> init_cb = nullptr,
00105     std::function<void(ExpressionPtr result_value)> result_cb = nullptr,
00106     std::shared_ptr<InputValidator> validator = nullptr);
00107
00108 private:
00109     friend class DataSwitch;
00110     KeyboardManager();
00111     void *d_{ nullptr };
00112 };
00113
00114 } // namespace aubo_scope
00115 } // namespace arcs
00116
00117 #endif // AUBO_CAPS_INTERFACE_KEYBOARD_MANAGER_H

```

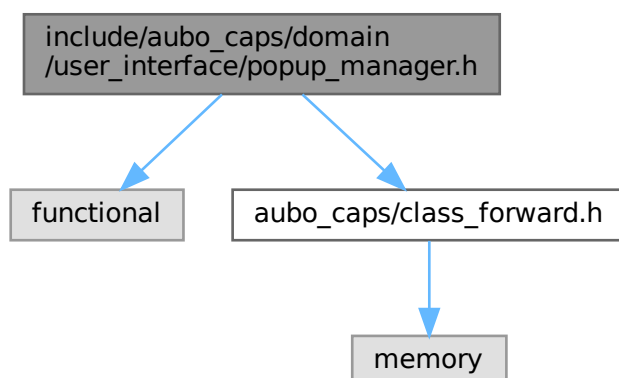
12.226 include/aubo_caps/domain/user_interface/popup_manager.h 文件参考

```

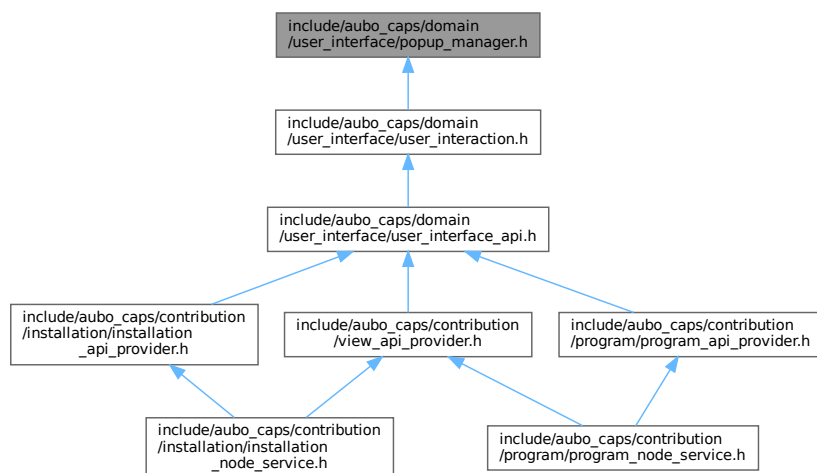
#include <functional>
#include <aubo_caps/class_forward.h>

```

popup_manager.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::PopupManager`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (PopupManager)`

12.227 popup_manager.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_POPUP_MANAGER_H
00002 #define AUBO_SCOPE_POPUP_MANAGER_H
00003
00004 #include <functional>
00005
00006 #include <aubo_caps/class_forward.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010
00011     template <typename Type>
00012     struct DefaultSingletonTraits;
00013
00014     ARCS_CLASS_FORWARD(PopupManager);
00015
00016     class ARCS_ABI_EXPORT PopupManager
00017     {
00018     public:
00019         PopupManager(PopupManager &f);
00020         PopupManager(PopupManager &&f);
00021         virtual ~PopupManager();
00022
00023         // 增加弹窗计数。应在受管理的弹窗显示时调用。
00024         void notifyPopupShown();
00025
00026         // 减少弹窗计数。应在受管理的弹窗隐藏或关闭时调用。
00027         void notifyPopupHidden();
00028
00029         // 返回当前跟踪的弹窗数量。
00030         int popupCount() const;
00031
00032         // 返回当前弹窗数量是否为 0。
00033         bool isPopupCountZero() const;
00034
00035         // 设置弹窗数量变化时的回调。参数 is_zero==true 表示当前数量为 0。
00036         void setPopupCountChangedCallback(std::function<void(bool)> callback);
00037
00038     private:
00039         friend struct DefaultSingletonTraits<PopupManager>;
00040         PopupManager();
00041         void *d_{ nullptr };
00042     };
00043
00044 } // namespace aubo_scope
00045 } // namespace arcs
00046
00047 #endif // AUBO_SCOPE_POPUP_MANAGER_H

```

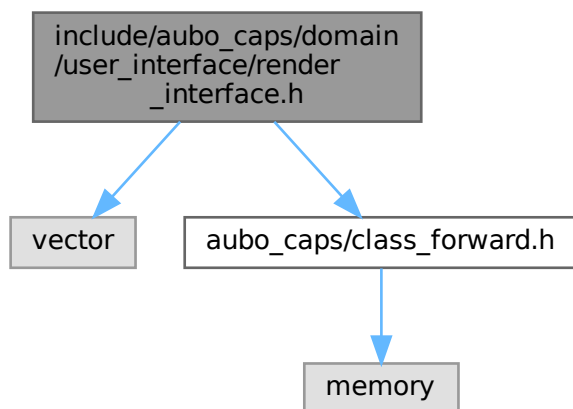
12.228 include/aubo_caps/domain/user__interface/render__↵ interface.h 文件参考

```

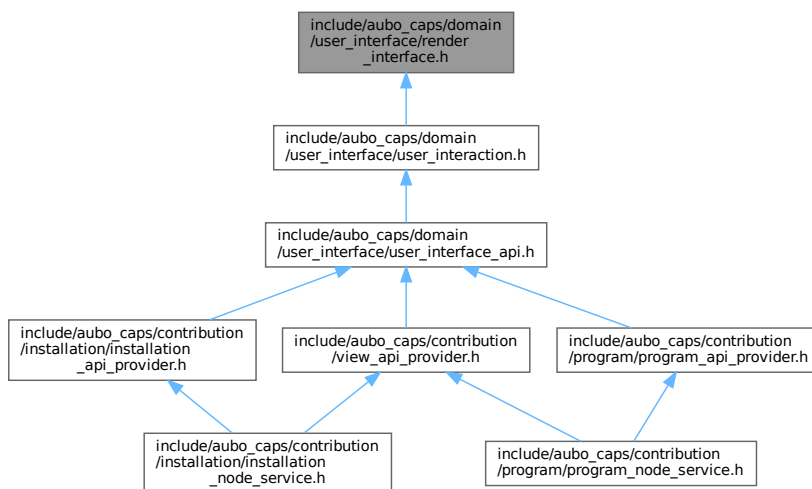
#include <vector>
#include <aubo_caps/class_forward.h>

```

render_interface.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::RenderInterface`

渲染接口提供 3D 渲染相关的显示选项控制, 包括地面、机器人模型、坐标系、轨迹和点等元素的显示/隐藏。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (RenderInterface)`

12.229 render_interface.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_CAPS_INTERFACE_RENDER_INTERFACE_H
00002 #define AUBO_CAPS_INTERFACE_RENDER_INTERFACE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/class_forward.h>
00006
00007 class QGridLayout;
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011
00012 ARCS_CLASS_FORWARD(RenderInterface);
00013
00014 /**
00015  * @ingroup UserInterfaceApi
00016  * \chinese
00017  * 渲染接口
00018  * 提供 3D 渲染相关的显示选项控制，包括地面、机器人模型、坐标系、轨迹和点等元素的
00019  * 显示/隐藏。
00020  * \endchinese
00021  * \english
00022  * RenderInterface
00023  * Provides display option controls related to 3D rendering, including show/hide
00024  * of ground, robot models, coordinate systems, trajectories and points.
00025  * \endenglish
00026  */
00027
00028 class ARCS_ABI_EXPORT RenderInterface
00029 {
00030 public:
00031     enum ShowOptions : uint16_t
00032     {
00033         None = 0,
00034         Ground = 0x01,
00035         TargetRobot = 0x02,
00036         RealRobot = 0x04,
00037         Coordinate = 0x08,
00038         Track = 0x10,
00039         Point = 0x20,
00040         View = 0x40,
00041         Obstacle = 0x80,
00042         SafetyPlane = 0x0100,
00043         ToolPosition = 0x0200,
00044         PointCloud = 0x0400,
00045         All = 0xFFFF,
00046     };
00047
00048 public:
00049     RenderInterface(RenderInterface &f);
00050     RenderInterface(RenderInterface &&f);
00051     virtual ~RenderInterface();
00052
00053     /**
00054      * @ingroup RenderInterface
00055      * \chinese
00056      * 设置 Render 窗口嵌入/移除 (isEmbedded) 布局 gly 中
00057      *
00058      * @param is_embedded 是否嵌入
00059      * @param gly 布局
00060      * \endchinese
00061      * \english
00062      * Set the render window to be embedded/removed in the layout gly.
00063      *
00064      * @param is_embedded whether to embed
00065      * @param gly the layout
00066      * \endenglish
00067      */
00068     void setRenderEmbedded(bool is_embedded, QGridLayout *gly);
00069
00070     /**
00071      * @ingroup RenderInterface
00072      * \chinese
00073      * 设置显示模式，控制各种元素的显隐
00074      *
00075      * @param options 选项值，详情查看 ShowOptions 枚举
00076      * \endchinese
00077      * \english
00078      * Set the display mode to control the visibility of various elements.
00079      *
00080      * @param options option value, see ShowOptions enum for details
00081      * \endenglish
00082      */
00083     void setShowOptions(uint16_t options);

```

```

00084
00085 /**
00086  * @ingroup RenderInterface
00087  * \chinese
00088  * 添加/修改点云
00089  *
00090  * @param name 点云名称
00091  * @param points 点集合
00092  * @param color 点云颜色 rgba
00093  * \endchinese
00094  * \english
00095  * Add/modify point cloud.
00096  *
00097  * @param name point cloud name
00098  * @param points point collection
00099  * @param color point cloud color rgba
00100  * \endenglish
00101  */
00102 void addPointCloud(const std::string &name,
00103                  const std::vector<std::vector<double>> &points,
00104                  const std::vector<double> &color);
00105
00106 /**
00107  * @ingroup RenderInterface
00108  * \chinese
00109  * 移除指定名称的点云
00110  *
00111  * @param name 点云名称
00112  * \endchinese
00113  * \english
00114  * Remove the point cloud with the specified name.
00115  *
00116  * @param name point cloud name
00117  * \endenglish
00118  */
00119 void removePointCloud(const std::string &name);
00120
00121 /**
00122  * @ingroup RenderInterface
00123  * \chinese
00124  * 查找指定名称的点云是否存在
00125  *
00126  * @param name 点云名称
00127  * @retval true 存在
00128  * @retval false 不存在
00129  * \endchinese
00130  * \english
00131  * Check whether the point cloud with the specified name exists.
00132  *
00133  * @param name point cloud name
00134  * @retval true exists
00135  * @retval false does not exist
00136  * \endenglish
00137  */
00138 bool hasPointCloud(const std::string &name);
00139
00140 /**
00141  * @ingroup RenderInterface
00142  * \chinese
00143  * 移除所有的点云
00144  * \endchinese
00145  * \english
00146  * Remove all point clouds.
00147  * \endenglish
00148  */
00149 void clearPointCloud();
00150
00151 /**
00152  * @ingroup RenderInterface
00153  * \chinese
00154  * 添加/修改路点
00155  *
00156  * @param wp 代表路点的唯一对象
00157  * @param pose 路点位姿 x,y,z,rx,ry,rz
00158  * @param color 路点颜色 {r, g, b, a}, 范围是 0.0-1.0
00159  * @param axis_visible 路点上是否显示位姿方向箭头
00160  * @retval true 成功
00161  * @retval false 失败
00162  * \endchinese
00163  * \english
00164  * Add/modify waypoint.
00165  *
00166  * @param wp the unique object representing the waypoint
00167  * @param pose waypoint pose x,y,z,rx,ry,rz
00168  * @param color waypoint color {r, g, b, a}, range 0.0-1.0
00169  * @param axis_visible whether to show pose direction arrows on the waypoint
00170  * @retval true success

```

```

00171     * @retval false failure
00172     * \endenglish
00173     */
00174     bool addWaypoint(void *wp, const std::vector<double> &pose,
00175                     const std::vector<double> &color = { 0.0, 1.0, 0.0, 1.0 },
00176                     bool axis_visible = false);
00177
00178     /**
00179     * @ingroup RenderInterface
00180     * \chinese
00181     * 移除指定路点
00182     *
00183     * @param wp 代表路点的唯一对象
00184     * \endchinese
00185     * \english
00186     * Remove the specified waypoint.
00187     *
00188     * @param wp the unique object representing the waypoint
00189     * \endenglish
00190     */
00191     void removeWaypoint(void *wp);
00192
00193     /**
00194     * @ingroup RenderInterface
00195     * \chinese
00196     * 查找指定路点是否存在
00197     *
00198     * @param wp 代表路点的唯一对象
00199     * @retval true 存在
00200     * @retval false 不存在
00201     * \endchinese
00202     * \english
00203     * Check whether the specified waypoint exists.
00204     *
00205     * @param wp the unique object representing the waypoint
00206     * @retval true exists
00207     * @retval false does not exist
00208     * \endenglish
00209     */
00210     bool hasWaypoint(void *wp);
00211
00212     /**
00213     * @ingroup RenderInterface
00214     * \chinese
00215     * 移除所有的路点
00216     * \endchinese
00217     * \english
00218     * Remove all waypoints.
00219     * \endenglish
00220     */
00221     void clearWaypoints();
00222
00223     /**
00224     * @ingroup RenderInterface
00225     * \chinese
00226     * 添加/修改指定路点的交融轨迹
00227     *
00228     * @param wp 代表路点的唯一对象
00229     * @param track 路点姿态 x,y,z,rx,ry,rz 集合
00230     * \endchinese
00231     * \english
00232     * Add/modify the move track for the specified waypoint.
00233     *
00234     * @param wp the unique object representing the waypoint
00235     * @param track waypoint pose collection x,y,z,rx,ry,rz
00236     * \endenglish
00237     */
00238     void setMoveTrack(void *wp, const std::vector<std::vector<double>> &track);
00239
00240     /**
00241     * @ingroup RenderInterface
00242     * \chinese
00243     * @brief setBlendTrack 设置混合轨迹
00244     * @param wp 路点对象
00245     * @param track 轨迹集合
00246     * \endchinese
00247     * \english
00248     * @brief setBlendTrack
00249     * @param wp
00250     * @param track
00251     * \endenglish
00252     */
00253     void setBlendTrack(void *wp, const std::vector<std::vector<double>> &track);
00254
00255     /**
00256     * @ingroup RenderInterface
00257     * \chinese

```

```

00258     * @brief setMoveTrack 设置移动轨迹
00259     * @param wp 路点对象
00260     * @param track 轨迹集合
00261     * @param color 颜色
00262     * \endchinese
00263     * \english
00264     * @brief setMoveTrack
00265     * @param wp
00266     * @param track
00267     * @param color
00268     * \endenglish
00269     */
00270 void setMoveTrack(void *wp, const std::vector<std::vector<double>> &track,
00271                  const std::vector<double> &color);
00272
00273 /**
00274  * @ingroup RenderInterface
00275  * \chinese
00276  * @brief setBlendTrack 设置混合轨迹
00277  * @param wp 路点对象
00278  * @param track 轨迹集合
00279  * @param color 颜色
00280  * \endchinese
00281  * \english
00282  * @brief setBlendTrack
00283  * @param wp
00284  * @param track
00285  * @param color
00286  * \endenglish
00287  */
00288 void setBlendTrack(void *wp, const std::vector<std::vector<double>> &track,
00289                   const std::vector<double> &color);
00290
00291 /**
00292  * @ingroup RenderInterface
00293  * \chinese
00294  * 添加/修改坐标轴
00295  *
00296  * @param name 坐标轴名称
00297  * @param pose 位姿 x,y,z,rx,ry,rz
00298  * \endchinese
00299  * \english
00300  * Add/modify coordinate axis.
00301  *
00302  * @param name coordinate name
00303  * @param pose pose x,y,z,rx,ry,rz
00304  * \endenglish
00305  */
00306 void addCoordinate(const std::string &name,
00307                  const std::vector<double> &pose);
00308
00309 /**
00310  * @ingroup RenderInterface
00311  * \chinese
00312  * 移除指定坐标轴
00313  *
00314  * @param name 坐标轴名称
00315  * \endchinese
00316  * \english
00317  * Remove the specified coordinate axis.
00318  *
00319  * @param name coordinate name
00320  * \endenglish
00321  */
00322 void removeCoordinate(const std::string &name);
00323
00324 /**
00325  * @ingroup RenderInterface
00326  * \chinese
00327  * 查找指定坐标轴是否存在
00328  *
00329  * @param name 坐标轴名称
00330  * @retval true 存在
00331  * @retval false 不存在
00332  * \endchinese
00333  * \english
00334  * Check whether the specified coordinate axis exists.
00335  *
00336  * @param name coordinate name
00337  * @retval true exists
00338  * @retval false does not exist
00339  * \endenglish
00340  */
00341 bool hasCoordinate(const std::string &name);
00342
00343 /**
00344  * @ingroup RenderInterface

```

```

00345     * \chinese
00346     * 添加/修改线
00347     *
00348     * @param name 线名称
00349     * @param link_num: -1: TCP; 0: 底座; 1-6: 关节;
00350     * @param pose: 点的参考坐标系, 默认 6 个 0
00351     * @param points 点集合
00352     * @param color 线颜色 {r, g, b, a}, 范围是 0.0~1.0
00353     * \endchinese
00354     * \english
00355     * Add/modify line.
00356     *
00357     * @param name line name
00358     * @param link_num: -1: TCP; 0: base; 1-6: joints;
00359     * @param pose: reference coordinate system for points, default 6 zeros
00360     * @param points point collection
00361     * @param color line color {r, g, b, a}, range 0.0~1.0
00362     * \endenglish
00363     */
00364 void addLine(const std::string &name,
00365             const std::vector<std::vector<double>> &points,
00366             const std::vector<double> &color);
00367
00368 /**
00369     * @ingroup RenderInterface
00370     * \chinese
00371     * @brief addLine 添加线
00372     * @param name 线名称
00373     * @param link_num 关节编号
00374     * @param pose 位姿
00375     * @param points 点集合
00376     * @param color 颜色
00377     * \endchinese
00378     * \english
00379     * @brief addLine
00380     * @param name
00381     * @param link_num
00382     * @param pose
00383     * @param points
00384     * @param color
00385     * \endenglish
00386     */
00387 void addLine(const std::string &name, int link_num,
00388             const std::vector<double> &pose,
00389             const std::vector<std::vector<double>> &points,
00390             const std::vector<double> &color);
00391
00392 /**
00393     * @ingroup RenderInterface
00394     * \chinese
00395     * 移除指定线
00396     *
00397     * @param name 线名称
00398     * \endchinese
00399     * \english
00400     * Remove the specified line.
00401     *
00402     * @param name line name
00403     * \endenglish
00404     */
00405 void removeLine(const std::string &name);
00406
00407 /**
00408     * @ingroup RenderInterface
00409     * \chinese
00410     * 移除所有的线
00411     * \endchinese
00412     * \english
00413     * Remove all lines.
00414     * \endenglish
00415     */
00416 void clearLines();
00417
00418 /**
00419     * @ingroup RenderInterface
00420     * \chinese
00421     * 添加/修改平面
00422     *
00423     * @param name 平面名称
00424     * @param link_num: -1: TCP; 0: 底座; 1-6: 关节;
00425     * @param pose 位置
00426     * @param size 平面大小 (长, 宽)
00427     * @param color 线颜色 {r, g, b, a}, 范围是 0.0~1.0
00428     * \endchinese
00429     * \english
00430     * Add/modify plane.
00431     *

```

```

00432     * @param name plane name
00433     * @param link_num: -1: TCP; 0: base; 1-6: joints;
00434     * @param pose position
00435     * @param size plane size (length, width)
00436     * @param color color {r, g, b, a}, range 0.0-1.0
00437     * \endenglish
00438     */
00439 void addPlane(const std::string &name, const std::vector<double> &pose,
00440             const std::vector<double> &size,
00441             const std::vector<double> &color);
00442
00443 /**
00444     * @ingroup RenderInterface
00445     * \chinese
00446     * @brief addPlane 添加平面
00447     * @param name 平面名称
00448     * @param link_num 关节编号
00449     * @param pose 位姿
00450     * @param size 大小
00451     * @param color 颜色
00452     * \endchinese
00453     * \english
00454     * @brief addPlane
00455     * @param name
00456     * @param link_num
00457     * @param pose
00458     * @param size
00459     * @param color
00460     * \endenglish
00461     */
00462 void addPlane(const std::string &name, int link_num,
00463             const std::vector<double> &pose,
00464             const std::vector<double> &size,
00465             const std::vector<double> &color);
00466
00467 /**
00468     * @ingroup RenderInterface
00469     * \chinese
00470     * 移除指定平面
00471     *
00472     * @param name 名称
00473     * \endchinese
00474     * \english
00475     * Remove the specified plane.
00476     *
00477     * @param name plane name
00478     * \endenglish
00479     */
00480 void removePlane(const std::string &name);
00481
00482 /**
00483     * @ingroup RenderInterface
00484     * \chinese
00485     * 移除所有的平面
00486     * \endchinese
00487     * \english
00488     * Remove all planes.
00489     * \endenglish
00490     */
00491 void clearPlanes();
00492
00493 /**
00494     * @ingroup RenderInterface
00495     * \chinese
00496     * 添加/修改立方体
00497     *
00498     * @param name 立方体名称
00499     * @param link_num: -1: TCP; 0: 底座; 1-6: 关节;
00500     * @param pose 位置
00501     * @param size 立方体大小 (长, 宽, 高)
00502     * @param color 立方体颜色 {r, g, b, a}, 范围是 0.0-1.0
00503     * @param type 立方体绘制类型 0 代表画 6 个平面, 1 代表只画边框线
00504     * \endchinese
00505     * \english
00506     * Add/modify cube.
00507     *
00508     * @param name cube name
00509     * @param link_num: -1: TCP; 0: base; 1-6: joints;
00510     * @param pose position
00511     * @param size cube size (length, width, height)
00512     * @param color cube color {r, g, b, a}, range 0.0-1.0
00513     * @param type cube rendering type: 0 for 6 planes, 1 for wireframe
00514     * \endenglish
00515     */
00516 void addCube(const std::string &name, const std::vector<double> &pose,
00517             const std::vector<double> &size,
00518             const std::vector<double> &color, int type = 0);

```



```

00519  /**
00520  * @ingroup RenderInterface
00521  * \chinese
00522  * @brief addCube 添加立方体
00523  * @param name 立方体名称
00524  * @param link_num 关节编号
00525  * @param pose 位姿
00526  * @param size 大小
00527  * @param color 颜色
00528  * @param type 类型
00529  * \endchinese
00530  * \english
00531  * @brief addCube
00532  * @param name
00533  * @param link_num
00534  * @param pose
00535  * @param size
00536  * @param color
00537  * @param type
00538  * \endenglish
00539  */
00540  void addCube(const std::string &name, int link_num,
00541              const std::vector<double> &pose,
00542              const std::vector<double> &size,
00543              const std::vector<double> &color, int type = 0);
00544
00545  /**
00546  * @ingroup RenderInterface
00547  * \chinese
00548  * 添加/修改立方体（带尺寸范围）
00549  *
00550  * @param name 立方体名称
00551  * @param pose 位置
00552  * @param length_min 最小长度
00553  * @param length_max 最大长度
00554  * @param width_min 最小宽度
00555  * @param width_max 最大宽度
00556  * @param height_min 最小高度
00557  * @param height_max 最大高度
00558  * @param color 立方体颜色 {r, g, b, a}, 范围是 0.0~1.0
00559  * @param type 立方体绘制类型 0 代表画 6 个平面, 1 代表只画边框线
00560  * \endchinese
00561  * \english
00562  * Add/modify cube with size ranges.
00563  *
00564  * @param name cube name
00565  * @param pose position
00566  * @param length_min
00567  * @param length_max
00568  * @param width_min
00569  * @param width_max
00570  * @param height_min
00571  * @param height_max
00572  * @param color cube color {r, g, b, a}, range 0.0~1.0
00573  * @param type cube rendering type: 0 for 6 planes, 1 for wireframe
00574  * \endenglish
00575  */
00576  void addCube(const std::string &name, const std::vector<double> &pose,
00577              float length_min, float length_max, float width_min,
00578              float width_max, float height_min, float height_max,
00579              const std::vector<double> &color, int type = 0);
00580
00581  /**
00582  * @ingroup RenderInterface
00583  * \chinese
00584  * @brief addCube 添加立方体
00585  * @param name 立方体名称
00586  * @param link_num 关节编号
00587  * @param pose 位姿
00588  * @param length_min 最小长度
00589  * @param length_max 最大长度
00590  * @param width_min 最小宽度
00591  * @param width_max 最大宽度
00592  * @param height_min 最小高度
00593  * @param height_max 最大高度
00594  * @param color 颜色
00595  * @param type 类型
00596  * \endchinese
00597  * \english
00598  * @brief addCube
00599  * @param name
00600  * @param link_num
00601  * @param pose
00602  * @param length_min
00603  * @param length_max
00604  * @param width_min
00605  * @param width_max

```

```

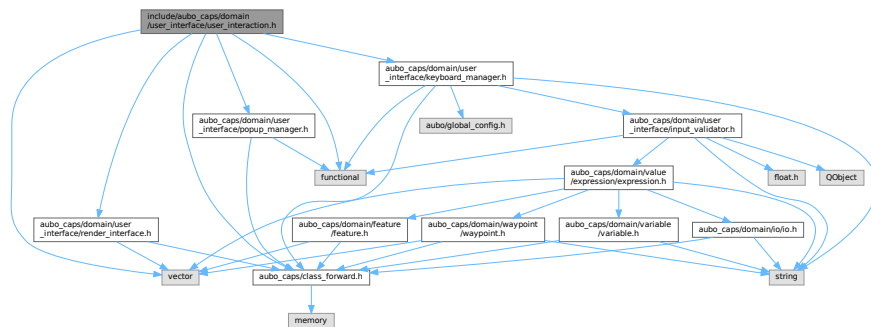
00606     * @param height_min
00607     * @param height_max
00608     * @param color
00609     * @param type
00610     * \endenglish
00611     */
00612 void addCube(const std::string &name, int link_num,
00613             const std::vector<double> &pose, float length_min,
00614             float length_max, float width_min, float width_max,
00615             float height_min, float height_max,
00616             const std::vector<double> &color, int type = 0);
00617
00618 /**
00619  * @ingroup RenderInterface
00620  * \chinese
00621  * 移除指定立方体
00622  *
00623  * @param name 立方体名称
00624  * \endchinese
00625  * \english
00626  * Remove the specified cube.
00627  *
00628  * @param name cube name
00629  * \endenglish
00630  */
00631 void removeCube(const std::string &name);
00632
00633 /**
00634  * @ingroup RenderInterface
00635  * \chinese
00636  * 移除所有的立方体
00637  * \endchinese
00638  * \english
00639  * Remove all cubes.
00640  * \endenglish
00641  */
00642 void clearCubes();
00643
00644 // 模型文件
00645 /**
00646  * @ingroup RenderInterface
00647  * @brief addMesh 添加任意模型文件
00648  * @param name: 唯一标识
00649  * @param link_num: -1: TCP; 0: 底座; 1-6: 关节;
00650  * @param model_path: 3ds 文件绝对路径
00651  * @param pose: 模型文件安装位置 {x, y, z, rx, ry, rz}
00652  * @return 0: 成功; 其他失败
00653  */
00654 int addMesh(const std::string &name, int link_num,
00655            const std::string &model_path, const std::vector<double> &pose);
00656
00657 /**
00658  * @ingroup RenderInterface
00659  * @brief getMeshes 获取关节 link_num 的外设模型名字
00660  * @param link_num: -1: TCP; 0: 底座; 1-6: 关节;
00661  * @return 名字列表
00662  */
00663 std::vector<std::string> getMeshes(int link_num);
00664
00665 /**
00666  * @ingroup RenderInterface
00667  * @brief removeMesh
00668  * @param name
00669  * @return
00670  */
00671 int removeMesh(const std::string &name);
00672
00673 /**
00674  * @ingroup RenderInterface
00675  * @brief clearMeshes
00676  * @param link_num
00677  * @return
00678  */
00679 int clearMeshes(int link_num);
00680
00681 private:
00682     friend class DataSwitch;
00683     RenderInterface();
00684     void *d_{ nullptr };
00685 };
00686
00687 } // namespace aubo_scope
00688 } // namespace arcs
00689
00690 #endif // AUBO_CAPS_INTERFACE_RENDER_INTERFACE_H

```

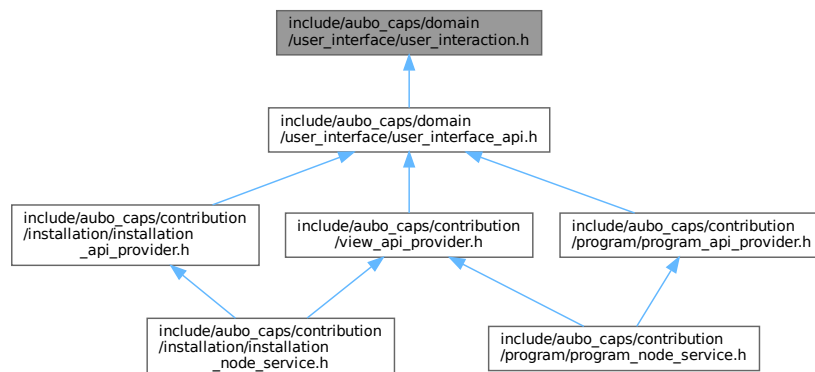
12.230 include/aubo_caps/domain/user_interface/user_ interaction.h 文件参考

```
#include <vector>
#include <functional>
#include <aubo_caps/class_forward.h>
#include <aubo_caps/domain/user_interface/keyboard_manager.h>
#include <aubo_caps/domain/user_interface/render_interface.h>
#include <aubo_caps/domain/user_interface/popup_manager.h>
```

user_interaction.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::UserInteraction](#)
用户交互接口此接口提供向终端用户请求输入或操作的功能。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

类型定义

- using [arcs::aubo_scope::RobotPositionCallback](#)
- using [arcs::aubo_scope::RobotMovementCallback](#) = std::function<void(bool accepted, int error)>

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (UserInteraction)`
- `arcs::aubo_scope::ARCS_CLASS_FORWARD (TCP)`

12.231 user_interaction.h

浏览该文件的文档.

```

00001 #ifndef AUBO_SCOPE_USER_INTERACTION_H
00002 #define AUBO_SCOPE_USER_INTERACTION_H
00003
00004 #include <vector>
00005 #include <functional>
00006
00007 #include <aubo_caps/class_forward.h>
00008 #include <aubo_caps/domain/user_interface/keyboard_manager.h>
00009 #include <aubo_caps/domain/user_interface/render_interface.h>
00010 #include <aubo_caps/domain/user_interface/popup_manager.h>
00011
00012 namespace arcs {
00013 namespace aubo_scope {
00014
00015 ARCS_CLASS_FORWARD(UserInteraction);
00016 ARCS_CLASS_FORWARD(TCP);
00017
00018 using RobotPositionCallback = std::function<void(
00019     bool accepted, const std::vector<double> &pose,
00020     const std::vector<double> &q, const std::vector<double> &offset)>;
00021 using RobotMovementCallback = std::function<void(bool accepted, int error)>;
00022
00023 /**
00024  * @ingroup UserInterfaceApi
00025  * \chinese
00026  * 用户交互接口
00027  * 此接口提供向终端用户请求输入或操作的功能。
00028  * \endchinese
00029  * \english
00030  * UserInteraction
00031  * This interface provides functionality for requesting input or actions from
00032  * end users.
00033  * \endenglish
00034  */
00035 class ARCS_ABI_EXPORT UserInteraction
00036 {
00037 public:
00038     UserInteraction(UserInteraction &f);
00039     UserInteraction(UserInteraction &&f);
00040     virtual ~UserInteraction();
00041
00042     /**
00043      * @ingroup UserInteraction
00044      * <p>
00045      * Request the end user to use the robot to define a robot position.
00046      * Override the
00047      * {@link RobotPositionCallback2#onOk(PositionParameters)} method to execute
00048      * code once the end user is done. Optionally override the {@link
00049      * RobotPositionCallback2#onCancel()} method to execute code if the end user
00050      * cancels the operation.
00051      * </p>
00052      *
00053      * <p>
00054      * This method is asynchronous, i.e. the method will return immediately.
00055      * Only when the end user is done, either
00056      * {@link RobotPositionCallback2#onOk(PositionParameters)} or {@link
00057      * RobotPositionCallback2#onCancel()} will be called.
00058      * </p>
00059      *
00060      * NOTE: This functionality should not be used in a toolbar contribution
00061      * (see {@link ToolbarService}).
00062      *
00063      * @param callback the instance callbacks will be made to.
00064      */
00065     void getUserDefinedRobotPosition(RobotPositionCallback callback);
00066
00067     /**
00068      * @ingroup UserInteraction
00069      * \chinese
00070      * 指定 tcpOffset 获取用户定义的机器人位置
00071      *
00072      * @param tcpOffset TCP 偏移量
00073      * @param callback 回调函数
00074      * \endchinese
00075      * \english

```

```

00076     * Request the end user to define a robot position with specified tcpOffset.
00077     *
00078     * @param tcpOffset
00079     * @param callback
00080     * \endenglish
00081     */
00082 void getUserDefinedRobotPosition(const std::vector<double> &tcpOffset,
00083                                 RobotPositionCallback callback);
00084
00085 /**
00086  * @ingroup UserInteraction
00087  * \chinese
00088  * 指定 tcpOffset 字符串获取用户定义的机器人位置
00089  *
00090  * @param tcpOffset TCP 偏移量名称
00091  * @param callback 回调函数
00092  * \endchinese
00093  * \english
00094  * Request the end user to define a robot position with specified tcpOffset string.
00095  *
00096  * @param tcpOffset
00097  * @param callback
00098  * \endenglish
00099  */
00100 void getUserDefinedRobotPosition(const std::string &tcpOffset,
00101                                 RobotPositionCallback callback);
00102
00103 /**
00104  * @ingroup UserInteraction
00105  * \chinese
00106  * 指定 TCP 对象获取用户定义的机器人位置
00107  *
00108  * @param tcpOffset TCP 对象
00109  * @param callback 回调函数
00110  * \endchinese
00111  * \english
00112  * Request the end user to define a robot position with specified TCP object.
00113  *
00114  * @param tcpOffset
00115  * @param callback
00116  * \endenglish
00117  */
00118 void getUserDefinedRobotPosition(const TCPPtr tcpOffset,
00119                                 RobotPositionCallback callback);
00120
00121 /**
00122  * @ingroup UserInteraction
00123  * <p>
00124  * This method provides a factory for creating keyboard inputs which are
00125  * used to configure a virtual keyboard/keypad and to request it to be
00126  * displayed for a Qt GUI component.
00127  * </p>
00128  *
00129  * NOTE: This functionality is only relevant for AuboCap nodes with a
00130  * Qt-based user interface (see {@link ProgramNodeService},
00131  * {@link InstallationNodeService} and {@link ToolbarService}).
00132  *
00133  * @return factory providing access to keyboard inputs.
00134  */
00135 KeyboardManagerPtr getKeyboardManager();
00136
00137 /**
00138  * @ingroup UserInteraction
00139  * <p>
00140  * Transition to the move tab widget to allow/request the end user to move
00141  * the robot (either automatically or manually) to a desired target position
00142  * specified by joint positions.
00143  * </p>
00144  *
00145  * <p>
00146  * The Automove function will move the robot to the target joint positions
00147  * linearly in joint space.
00148  * </p>
00149  *
00150  * <b>NOTE:</b> This method does not take the currently active TCP offset
00151  * into account, i.e. the Automove function will have the specified joint
00152  * positions as target.
00153  *
00154  * @param q The target joint positions to move to.
00155  * @param callback The instance which callbacks will be made to.
00156  * @throws IllegalStateException If the plugin screen is not active
00157  * (visible).
00158  */
00159 int requestUserToMoveJoint(const std::vector<double> &q,
00160                           RobotMovementCallback callback);
00161
00162 /**
00163  * @ingroup UserInteraction

```

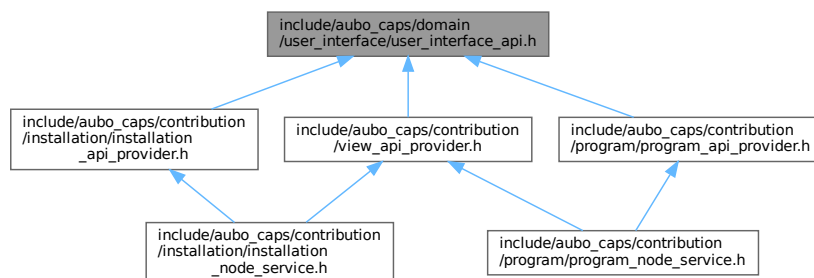
```

00163     * @brief requestUserToMoveJoint
00164     * @param q
00165     * @param callback
00166     * @return
00167     */
00168     int requestUserToMoveJoint(const std::string &q,
00169                               RobotMovementCallback callback);
00170 /**
00171     * @ingroup UserInteraction
00172     * @param pose The target pose the robot TCP will move to.
00173     * @param q The reference joint Angle.
00174     * @param callback The instance which callbacks will be made to.
00175     * @throws IllegalStateException If the plugin screen is not active
00176     * (visible).
00177     */
00178     int requestUserToMoveJoint(const std::vector<double> &pose,
00179                               const std::vector<double> &q,
00180                               RobotMovementCallback callback);
00181
00182 /**
00183     * @ingroup UserInteraction
00184     * <p>
00185     * Transition to the move tab widget to allow/request the end user to move
00186     * the robot (either automatically or manually) to a desired target position
00187     * specified by a pose.
00188     * </p>
00189     *
00190     * <p>
00191     * The Automove function will move the robot TCP to the target position
00192     * linearly in Cartesian space. If this it not possible, the robot will move
00193     * to the target linearly in joint space.
00194     * </p>
00195     *
00196     * The current joint positions of the robot will be used as starting point
00197     * for inverse kinematics to calculate a target joint vector corresponding
00198     * to desired pose, taking the currently active TCP offset into account.
00199     *
00200     * @param pose The target pose the robot TCP will move to.
00201     * @param callback The instance which callbacks will be made to.
00202     * @throws IllegalStateException If the plugin screen is not active
00203     * (visible).
00204     */
00205     QT_DEPRECATED int requestUserToMoveLine(const std::vector<double> &pose,
00206                                             RobotMovementCallback callback);
00207
00208 /**
00209     * @ingroup UserInteraction
00210     * <p>
00211     * Transition to the move tab widget to allow/request the end user to move
00212     * the robot (either automatically or manually) to a desired target position
00213     * specified by a pose.
00214     * </p>
00215     *
00216     * <p>
00217     * The Automove function will move the robot TCP to the target position
00218     * linearly in Cartesian space. If this it not possible, the robot will move
00219     * to the target linearly in joint space.
00220     * </p>
00221     *
00222     * The current joint positions of the robot will be used as starting point
00223     * for inverse kinematics to calculate a target joint vector corresponding
00224     * to desired pose, taking the currently active TCP offset into account.
00225     *
00226     * @param pose The target pose the robot TCP will move to.
00227     * @param tcpOffset
00228     * @param callback The instance which callbacks will be made to.
00229     * @throws IllegalStateException If the plugin screen is not active
00230     * (visible).
00231     */
00232     int requestUserToMoveLine(const std::vector<double> &pose,
00233                               const std::vector<double> &tcpOffset,
00234                               RobotMovementCallback callback);
00235
00236 /**
00237     * @ingroup UserInteraction
00238     * \chinese
00239     * 获取机器人缩略图
00240     *
00241     * @param q 关节角度
00242     * @param pose 位姿
00243     * @return 机器人缩略图
00244     * \endchinese
00245     * \english
00246     * Get robot thumbnail.
00247     *
00248     * @param q joint angles
00249     * @param pose pose

```

12.232 `include/aubo_caps/domain/user_interface/user_↵
interface_api.h` 文件参考

此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::UserInterfaceApi`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (UserInterfaceApi)`

12.233 user__interface__api.h

浏览该文件的文档.

```

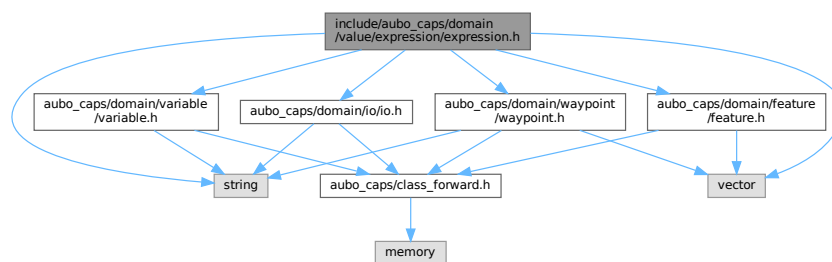
00001 #ifndef AUBO_SCOPE_USER_INTERFACE_API_H
00002 #define AUBO_SCOPE_USER_INTERFACE_API_H
00003
00004 #include <aubo_caps/domain/user_interface/user_interaction.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(UserInterfaceApi);
00009
00010 /**
00011  * \chinese
00012  * @defgroup UserInterfaceApi UserInterfaceApi(用户界面)
00013  * @ingroup UserInterfaceApi
00014  * 用户界面接口
00015  * 提供对 AuboScope 中与用户界面和终端用户交互相关的功能和服务访问。
00016  * \endchinese
00017  * \english
00018  * @defgroup UserInterfaceApi UserInterfaceApi
00019  * @ingroup UserInterfaceApi
00020  * UserInterfaceApi
00021  * Provides access to functionality and services available from within AuboScope
00022  * related to user interface and end user interaction.
00023  * \endenglish
00024  */
00025 class ARCS_ABI_EXPORT UserInterfaceApi
00026 {
00027 public:
00028     UserInterfaceApi(UserInterfaceApi &f);
00029     UserInterfaceApi(UserInterfaceApi &&f);
00030     virtual ~UserInterfaceApi();
00031
00032     /**
00033      * @ingroup UserInterfaceApi
00034      * @ref UserInteraction
00035      * \chinese
00036      * @return 提供向终端用户请求输入和操作功能的接口
00037      * \endchinese
00038      * \english
00039      * @return An interface which provides functionality for requesting input
  
```



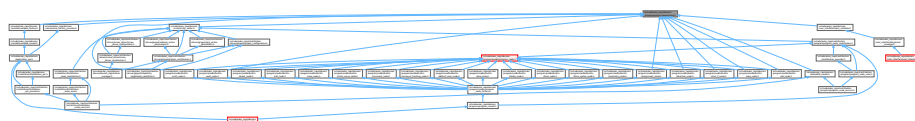
```
00040      * and actions from end users.
00041      * \endenglish
00042      */
00043      UserInteractionPtr getUserInteraction();
00044
00045 private:
00046     friend class DataSwitch;
00047     UserInterfaceApi();
00048     void *d_{ nullptr };
00049 };
00050
00051 } // namespace aubo_scope
00052 } // namespace arcs
00053
00054 #endif // AUBO_SCOPE_USER_INTERFACE_API_H
```

12.234 include/auco_caps/domain/value/expression/↵ expression.h 文件参考

```
#include <string>
#include <vector>
#include <auco_caps/domain/variable/variable.h>
#include <auco_caps/domain/feature/feature.h>
#include <auco_caps/domain/waypoint/waypoint.h>
#include <auco_caps/domain/io/io.h>
expression.h 的引用 (Include) 关系图:
```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::auco_scope::Expression`

表达式表达式可用于配置各种内置程序节点，例如 *If* 和 *Wait* 节点。表达式通常是条件表达式（计算结果为 *true* 或 *false*）。

命名空间

- namespace `arcs`
- namespace `arcs::auco_scope`

函数

- `arcs::auco_scope::ARCS_CLASS_FORWARD (Expression)`

12.235 expression.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_EXPRESSION_H
00002 #define AUBO_SCOPE_EXPRESSION_H
00003
00004 #include <string>
00005 #include <vector>
00006
00007 #include <aubo_caps/domain/variable/variable.h>
00008 #include <aubo_caps/domain/feature/feature.h>
00009 #include <aubo_caps/domain/waypoint/waypoint.h>
00010 #include <aubo_caps/domain/io/io.h>
00011
00012 namespace arcs {
00013 namespace aubo_scope {
00014 ARCS_CLASS_FORWARD(Expression);
00015
00016 /**
00017  * \chinese
00018  * 表达式
00019  * 表达式可用于配置各种内置程序节点，例如 If 和 Wait 节点。表达式通常是条件表达式
00020  * (计算结果为 true 或 false)。
00021  *
00022  * 表达式通过 {@link ExpressionBuilder} 接口构建。
00023  * \endchinese
00024  * \english
00025  * Expression
00026  * An expression can be used for, e.g. configuration of various built-in program
00027  * nodes, such as If and Wait nodes. Expressions are typically conditional
00028  * expressions (which evaluates to true or false).
00029  *
00030  * An Expression is built using the {@link ExpressionBuilder} interface.
00031  * \endenglish
00032  */
00033 class ARCS_ABI_EXPORT Expression
00034 {
00035 public:
00036     Expression(Expression &f);
00037     Expression(Expression &&f);
00038     virtual ~Expression();
00039
00040     std::string toString();
00041
00042     /**
00043      * \chinese
00044      * 向表达式追加字符串部分。
00045      * @param expressionPart 要追加的表达式部分。
00046      * \endchinese
00047      * \english
00048      * Append a string part of the expression.
00049      * @param expressionPart the expression part to be appended.
00050      * \endenglish
00051      */
00052     void append(const std::string &expressionPart);
00053
00054     /**
00055      * \chinese 向表达式追加变量对象。 \endchinese
00056      * \english Append a variable object to the expression. @param variable the variable. \endenglish
00057      */
00058     void appendVariable(VariablePtr variable);
00059
00060     /**
00061      * \chinese 向表达式追加特征对象。 \endchinese
00062      * \english Append a Feature object to the expression. @param feature the Feature. \endenglish
00063      */
00064     void appendFeature(FeaturePtr feature);
00065
00066     /**
00067      * \chinese 向表达式追加路点对象。 \endchinese
00068      * \english Append a waypoint to the expression. @param waypoint the waypoint. \endenglish
00069      */
00070     void appendWaypoint(WaypointPtr waypoint);
00071
00072     /**
00073      * \chinese 向表达式追加 I/O 对象。 \endchinese
00074      * \english Append an I/O object to the expression. @param io the I/O. \endenglish
00075      */
00076     void appendIo(IoPtr io);
00077
00078 private:
00079     friend class DataSwitch;
00080     Expression();
00081     void *d_{ nullptr };
00082 };
00083

```

```

00084 } // namespace aubo_scope
00085 } // namespace arcs
00086 #endif // AUBO_SCOPE_EXPRESSION_H

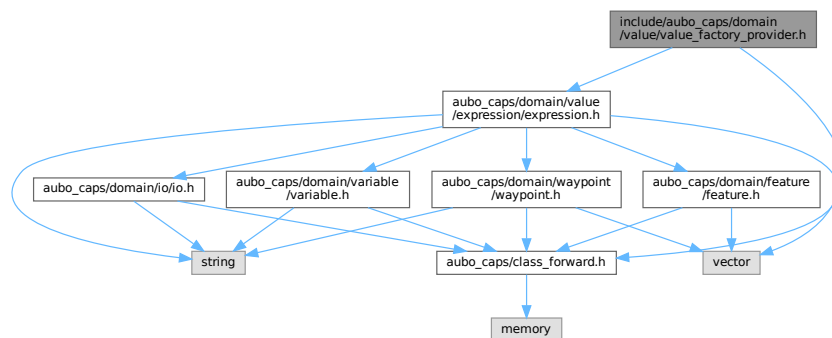
```

12.236 include/aubo_caps/domain/value/value_factory_provider.h 文件参考

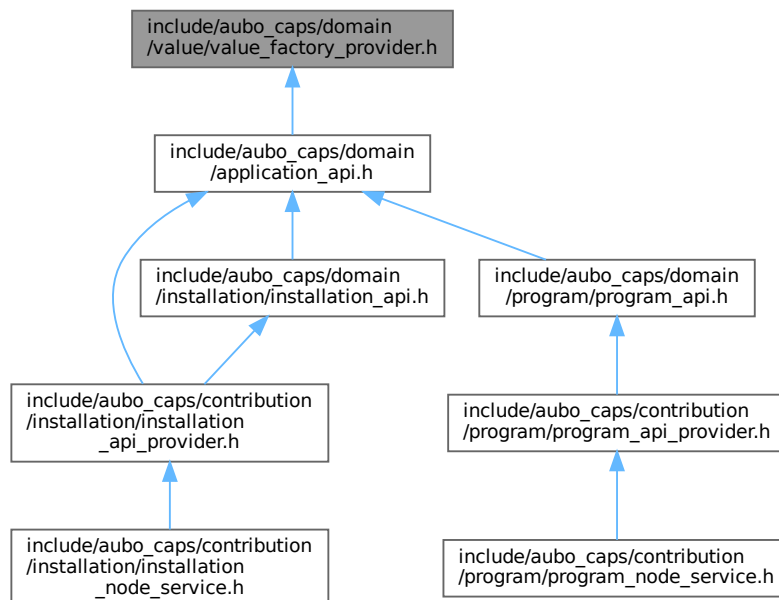
```

#include <aubo_caps/class_forward.h>
#include <aubo_caps/domain/value/expression/expression.h>
value_factory_provider.h 的引用 (Include) 关系图:

```



此图展示该文件被哪些文件直接或间接地引用了:



类

- class arcs::aubo_scope::ValueFactory
值工厂接口

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`ValueFactory`)

12.237 value_factory_provider.h

浏览该文件的文档.

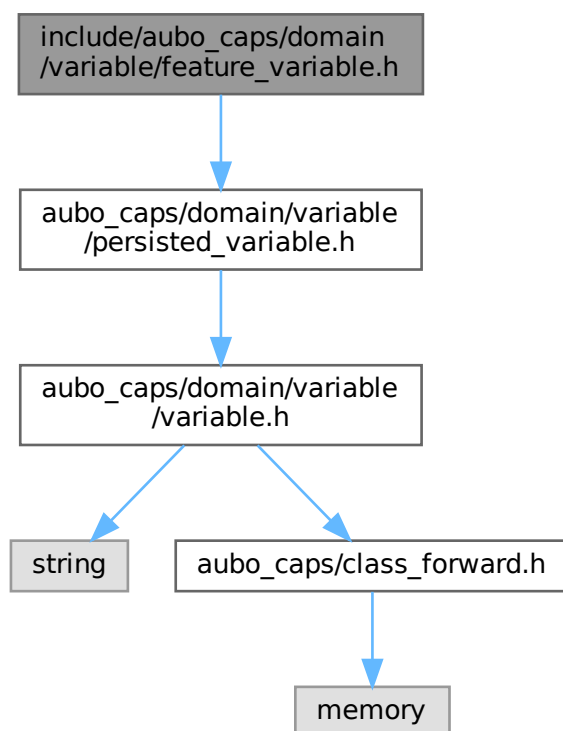
```

00001 #ifndef AUBO_SCOPE_VALUE_FACTORY_H
00002 #define AUBO_SCOPE_VALUE_FACTORY_H
00003
00004 #include <aubo_caps/class_forward.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(ValueFactory);
00010
00011 /**
00012  * @ingroup ApplicationApi
00013  * \chinese
00014  * 值工厂接口
00015  * <h1> 概述 </h1>
00016  * 此接口提供使用工厂创建不可变值对象的方法。{@link SimpleValueFactory} 帮助创建
00017  * 简单值对象。{@link ExpressionBuilder} 帮助构建有效表达式。{@link BlendFactory}
00018  * 帮助创建过渡对象。{@link JointPositionFactory} 帮助创建关节位置对象。
00019  * {@link PoseFactory} 帮助创建姿态对象。
00020  * \endchinese
00021  * \english
00022  * ValueFactory
00023  * <h1>Summary</h1>
00024  * This interface provides a mean to create immutable value objects using
00025  * factories. The {@link SimpleValueFactory} helps creating simple value
00026  * objects. The {@link ExpressionBuilder} helps building valid expressions. The
00027  * {@link BlendFactory} helps creating blend objects. The {@link
00028  * JointPositionFactory} helps creating joint positions objects. The {@link
00029  * PoseFactory} helps creating Pose objects.
00030  * \endenglish
00031  */
00032 class ARCS_ABI_EXPORT ValueFactory
00033 {
00034 public:
00035     ValueFactory(ValueFactory &f);
00036     ValueFactory(ValueFactory &&f);
00037     virtual ~ValueFactory();
00038
00039     /**
00040      * \chinese
00041      * 实例化一个新的 {@link ExpressionBuilder} 来创建单个有效表达式。
00042      * @return 表达式构建器。
00043      * \endchinese
00044      * \english
00045      * This method instantiates a new {@link ExpressionBuilder} to create a
00046      * single valid expression.
00047      *
00048      * @return the builder to build a single expression.
00049      * \endenglish
00050      */
00051     ExpressionPtr createExpression();
00052     ExpressionPtr createExpression(const std::string &str);
00053
00054 private:
00055     friend class DataSwitch;
00056     ValueFactory();
00057     void *d_{ nullptr };
00058 };
00059 } // namespace aubo_scope
00060 } // namespace arcs
00061
00062 #endif

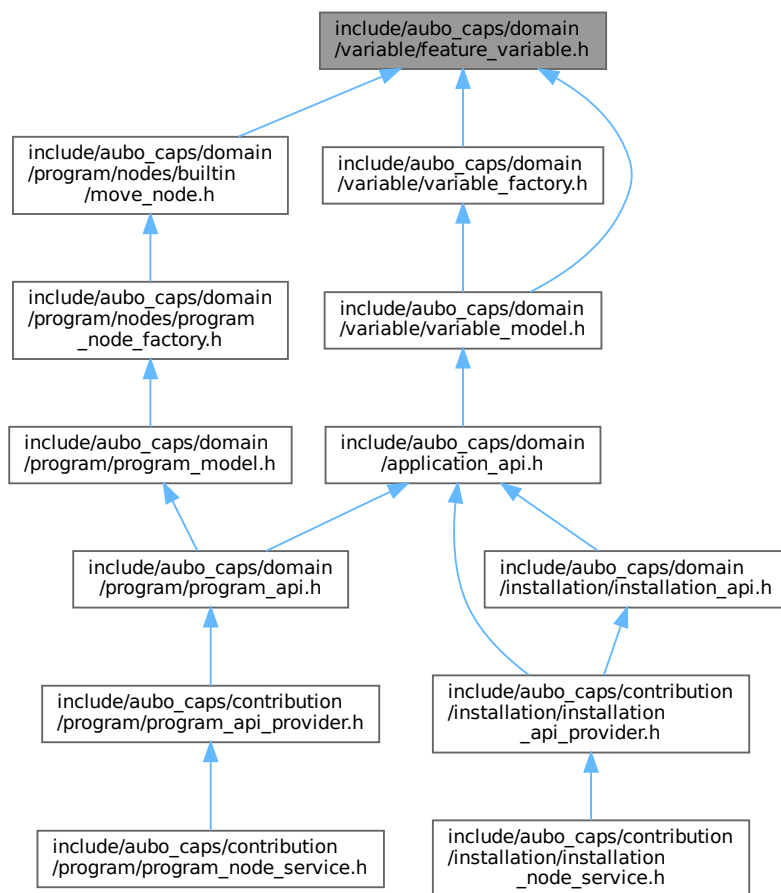
```

12.238 include/aubo_caps/domain/variable/feature__variable.h 文件参考

#include <aubo_caps/domain/variable/persisted_variable.h>
feature_variable.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::FeatureVariable`
特征变量在安装中被标记为变量的特征，即使在机器人关闭和重新启动后也会存在。
- struct `arcs::aubo_scope::FeatureVariableFilter`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (FeatureVariable)`

12.239 feature_variable.h

浏览该文件的文档.

```

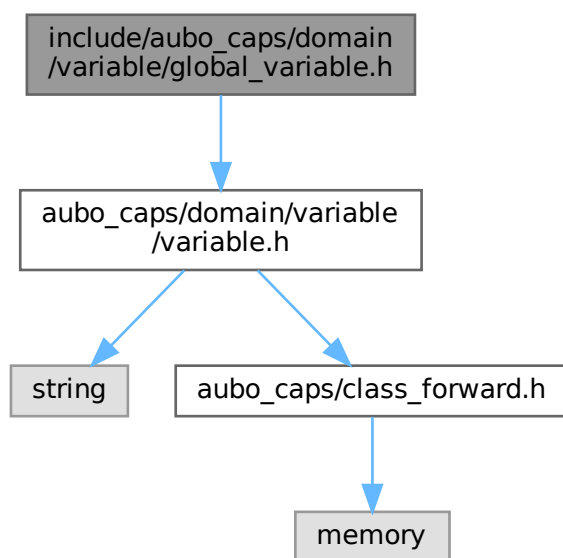
00001 #ifndef AUBO_SCOPE_FEATURE_VARIABLE_H
00002 #define AUBO_SCOPE_FEATURE_VARIABLE_H
00003
00004 #include <aubo_caps/domain/variable/persisted_variable.h>
00005

```

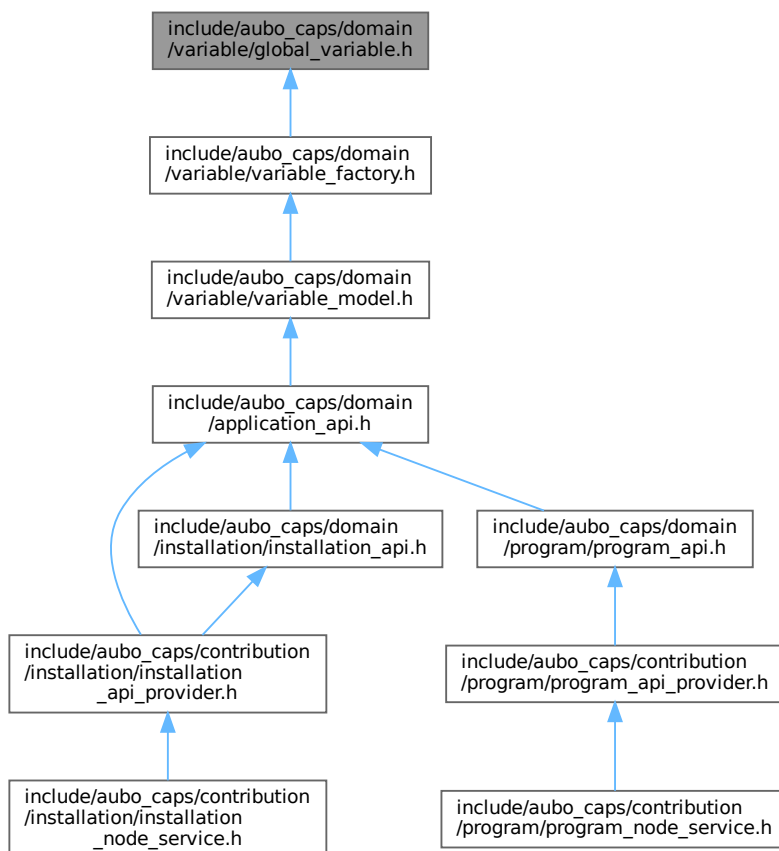
```
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(FeatureVariable);
00009
00010 /**
00011  * \chinese
00012  * 特征变量
00013  * 在安装中被标记为变量的特征，即使在机器人关闭和重新启动后也会存在。
00014  * \endchinese
00015  * \english
00016  * FeatureVariable
00017  * A Feature marked as a Variable in the installation will exist even when a
00018  * robot is turned off and on.
00019  * \endenglish
00020 */
00021 class ARCS_ABI_EXPORT FeatureVariable : public PersistedVariable
00022 {
00023 public:
00024     FeatureVariable(FeatureVariable &f);
00025     FeatureVariable(FeatureVariable &&f);
00026     virtual ~FeatureVariable();
00027
00028 private:
00029     friend class DataSwitch;
00030     FeatureVariable();
00031     void *d_{ nullptr };
00032 };
00033
00034 struct FeatureVariableFilter
00035 {
00036     bool operator()(VariablePtr var)
00037     {
00038         return std::dynamic_pointer_cast<FeatureVariable>(var) != nullptr &&
00039             var->getType() == Variable::FEATURE;
00040     }
00041 };
00042
00043 } // namespace aubo_scope
00044 } // namespace arcs
00045 #endif // AUBO_SCOPE_FEATURE_VARIABLE_H
```

12.240 include/aubo_caps/domain/variable/global_variable.h 文件参考

`#include <aubo_caps/domain/variable/variable.h>`
global_variable.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了：



类

- class [arcs::aubo_scope::GlobalVariable](#)
Global variables are available everywhere in a program.
- struct [arcs::aubo_scope::GlobalVariableFilter](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD\(GlobalVariable\)](#)

12.241 global_variable.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_GLOBAL_VARIABLE_H
00002 #define AUBO_SCOPE_GLOBAL_VARIABLE_H
00003
00004 #include <aubo_caps/domain/variable/variable.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {

```

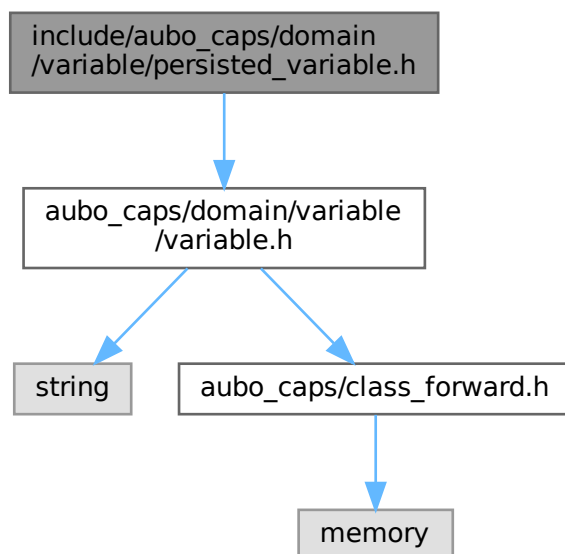
```

00008 ARCS_CLASS_FORWARD(GlobalVariable);
00009
00010 /**
00011  * Global variables are available everywhere in a program.
00012  */
00013 class ARCS_ABI_EXPORT GlobalVariable : public Variable
00014 {
00015 public:
00016     GlobalVariable(GlobalVariable &f);
00017     GlobalVariable(GlobalVariable &&f);
00018     virtual ~GlobalVariable();
00019
00020 protected:
00021     GlobalVariable();
00022
00023 private:
00024     friend class DataSwitch;
00025     void *d_{ nullptr };
00026 };
00027
00028 struct GlobalVariableFilter
00029 {
00030     bool operator()(VariablePtr var)
00031     {
00032         return std::dynamic_pointer_cast<GlobalVariable>(var) != nullptr &&
00033             var->getType() == Variable::GLOBAL;
00034     }
00035 };
00036
00037 } // namespace aubo_scope
00038 } // namespace arcs
00039
00040 #endif // AUBO_SCOPE_GLOBAL_VARIABLE_H

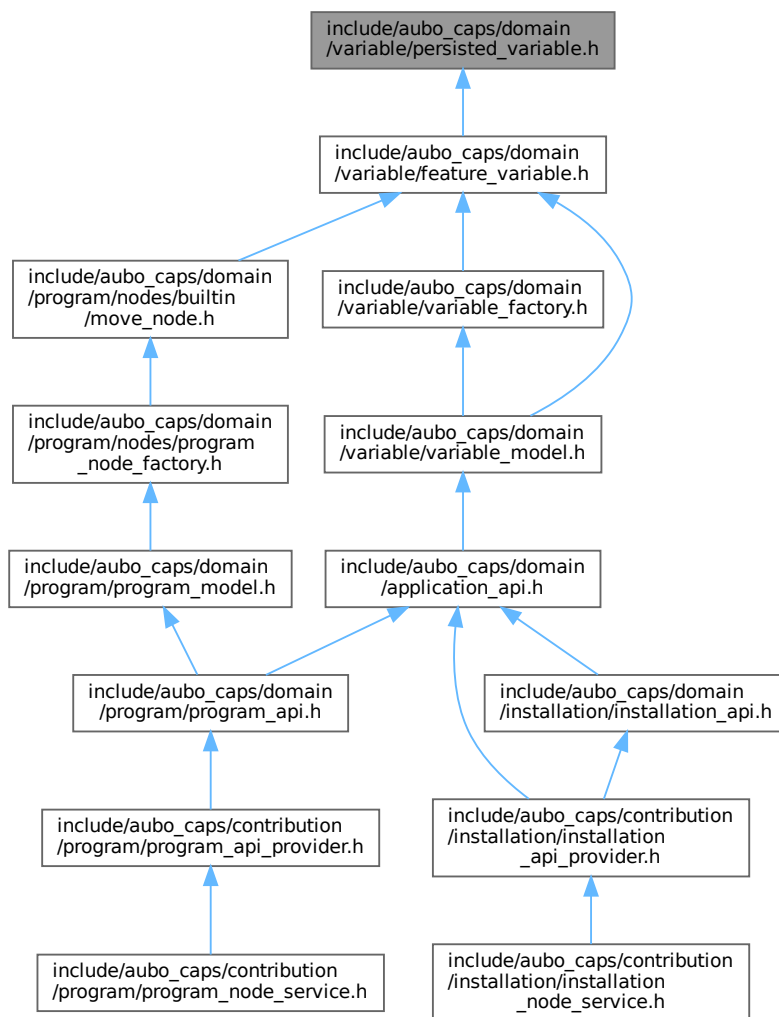
```

12.242 include/auco_caps/domain/variable/persisted_↵ variable.h 文件参考

#include <auco_caps/domain/variable/variable.h>
persisted_variable.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::PersistedVariable](#)
ingroup ApplicationApi This interface represents variables stored with the installation.
- struct [arcs::aubo_scope::PersistedVariableFilter](#)

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(PersistedVariable\)](#)

12.243 persisted_variable.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_PERSISTED_VARIABLE_H
00002 #define AUBO_SCOPE_PERSISTED_VARIABLE_H
00003
00004 #include <aubo_caps/domain/variable/variable.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(PersistedVariable);
00009
00010 /**
00011 * ingroup ApplicationApi
00012 * This interface represents variables stored with the installation. These
00013 * variables are defined in the installation and will exist even when a robot is
00014 * turned off and on.
00015 */
00016 class ARCS_ABI_EXPORT PersistedVariable : public Variable
00017 {
00018 public:
00019     PersistedVariable(PersistedVariable &f);
00020     PersistedVariable(PersistedVariable &&f);
00021     virtual ~PersistedVariable();
00022
00023     /**
00024      * @ingroup PersistedVariable
00025      * \chinese
00026      * 持久化变量无法保证始终存在。当终端用户加载不包含该变量的不同安装或删除该变量时，
00027      * 持久化变量将不可用。如果变量无法解析，存储了 PersistedVariable 的程序节点将变
00028      * 为未定义。
00029      * @return 如果持久化变量可解析（即存在于当前安装中）返回 <code>true</code>。
00030      * \endchinese
00031      * \english
00032      * A persisted variable can not be guaranteed to be present. This method can
00033      * be used to determine, if the variable is present.
00034      *
00035      * The persisted variable will not be present, if the end user loads a
00036      * different installation which does not contain the variable, or if the
00037      * variable is deleted by the end user.
00038      *
00039      * A program node storing a <code>PersistedVariable</code> in its Data
00040      * Model will be undefined if the variable cannot be resolved (i.e. the
00041      * variable is not present).
00042      *
00043      * @return <code>true</code>, if this persisted variable is resolvable (i.e.
00044      * present in the current installation).
00045      * \endenglish
00046      */
00047     bool isResolvable();
00048
00049 protected:
00050     PersistedVariable();
00051
00052 private:
00053     friend class DataSwitch;
00054     void *d_{ nullptr };
00055 };
00056
00057 struct PersistedVariableFilter
00058 {
00059     bool operator()(VariablePtr var)
00060     {
00061         return std::dynamic_pointer_cast<PersistedVariable>(var) != nullptr &&
00062             var->getType() == Variable::VALUE_PERSISTED;
00063     }
00064 };
00065
00066 } // namespace aubo_scope
00067 } // namespace arcs
00068 #endif // AUBO_SCOPE_PERSISTED_VARIABLE_H

```

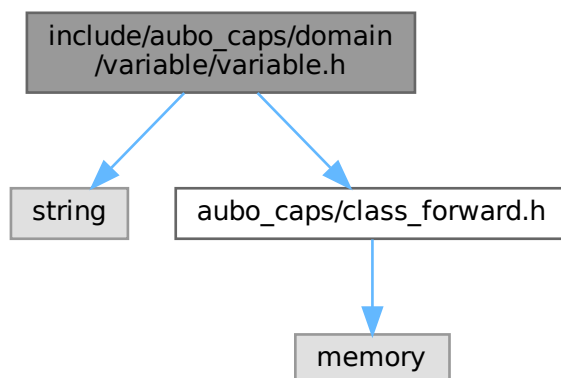
12.244 include/aubo_caps/domain/variable/variable.h 文件参考

```

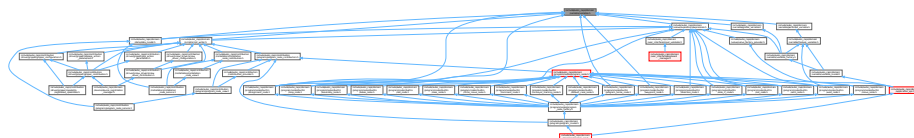
#include <string>
#include <aubo_caps/class_forward.h>

```

variable.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::Variable`
变量接口表示 *AuboScope* 中的变量。变量可用于在程序中存储和传递数据。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (Variable)`

12.245 variable.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_VARIABLE_H
00002 #define AUBO_SCOPE_VARIABLE_H
00003
00004 #include <string>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(Variable);
00010
00011 /**
00012  * @ingroup ApplicationApi
00013  * \chinese
00014  * 变量接口
00015  * 表示 AuboScope 中的变量。变量可用于在程序中存储和传递数据。
  
```

```

00016 * \endchinese
00017 * \english
00018 * Variable
00019 * Represents a variable in AuboScope. Variables can be used to store and pass
00020 * data in programs.
00021 * \endenglish
00022 */
00023
00024 class ARCS_ABI_EXPORT Variable
00025 {
00026 public:
00027     Variable(Variable &f);
00028     Variable(Variable &&f);
00029     virtual ~Variable();
00030
00031     /**
00032     * The variable type used to determine which type of variable this instance
00033     * is.
00034     */
00035     enum Type : int
00036     {
00037
00038         /**
00039         * <p>
00040         * GLOBAL means that the variable is available everywhere in a program.
00041         * </p>
00042         *
00043         * This variable instance can be cast to {@link GlobalVariable}.
00044         */
00045         GLOBAL,
00046
00047         /**
00048         * <p>
00049         * VALUE_PERSISTED means that the variable is stored with the
00050         * installation. These variables are defined in the installation and
00051         * will retain their values even when a robot is turned off and on.
00052         * </p>
00053         *
00054         * This variable instance can be cast to {@link ValuePersistedVariable}.
00055         */
00056         VALUE_PERSISTED,
00057
00058         /**
00059         * <p>
00060         * FEATURE means that a Feature is marked as a variable. The variable is
00061         * defined in the installation and will exist even when a robot is
00062         * turned off and on.
00063         * </p>
00064         *
00065         * This variable instance can be cast to {@link FeatureVariable}.
00066         */
00067         FEATURE
00068     };
00069
00070     /**
00071     * @ingroup Variable
00072     * \chinese
00073     * 获取变量类型
00074     * @return 变量类型枚举。
00075     * \endchinese
00076     * \english
00077     * Returns the type of variable.
00078     * @return the type of variable.
00079     * \endenglish
00080     */
00081     Type getType();
00082
00083     /**
00084     * @ingroup Variable
00085     * \chinese
00086     * 获取变量显示名称。注意：变量可随时被重命名。
00087     * @return 变量显示名称。
00088     * \endchinese
00089     * \english
00090     * Note: The variable can be renamed at any time.
00091     * @return the display name of the variable.
00092     * \endenglish
00093     */
00094     std::string getDisplayName();
00095
00096     /**
00097     * @ingroup Variable
00098     * \chinese
00099     * 判断是否引用相同的内部变量
00100     * @return 如果对象引用相同的内部变量返回 true。
00101     * \endchinese
00102     * \english

```

```

00103     * Checks if the object references the same internal Variable.
00104     * @return true if object references the same internal Variable.
00105     * \endenglish
00106     */
00107     bool equals(VariablePtr object);
00108
00109     /**
00110     * @ingroup Variable
00111     * \chinese
00112     * 获取哈希码
00113     * @return 此对象的哈希码。
00114     * \endchinese
00115     * \english
00116     * Returns the hash code for this object.
00117     * @return hashCode for this object.
00118     * \endenglish
00119     */
00120     size_t hashCode();
00121
00122     /**
00123     * @ingroup Variable
00124     * \chinese
00125     * 获取变量的字符串表示
00126     * @return 变量的字符串表示。
00127     * \endchinese
00128     * \english
00129     * Returns a string representation of the variable.
00130     * @return a string representation of the variable.
00131     * \endenglish
00132     */
00133     std::string toString();
00134
00135 protected:
00136     Variable();
00137
00138 private:
00139     friend class DataSwitch;
00140     void *d_{ nullptr };
00141 };
00142
00143 } // namespace aubo_scope
00144 } // namespace arcs
00145 #endif // AUBO_SCOPE_VARIABLE_H

```

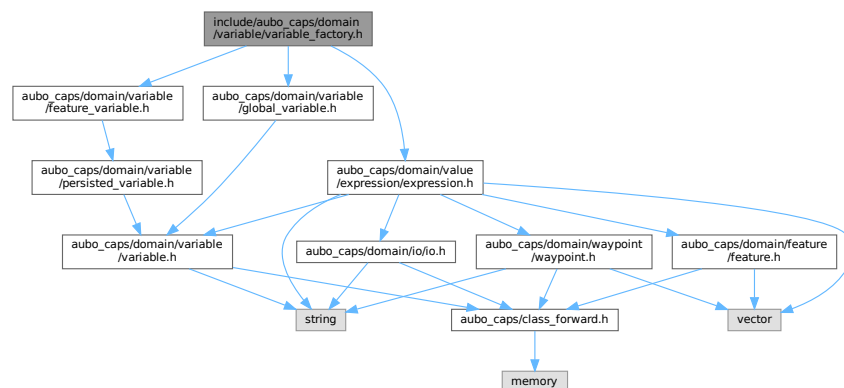
12.246 include/auco_caps/domain/variable/variable__factory.h 文件参考

```

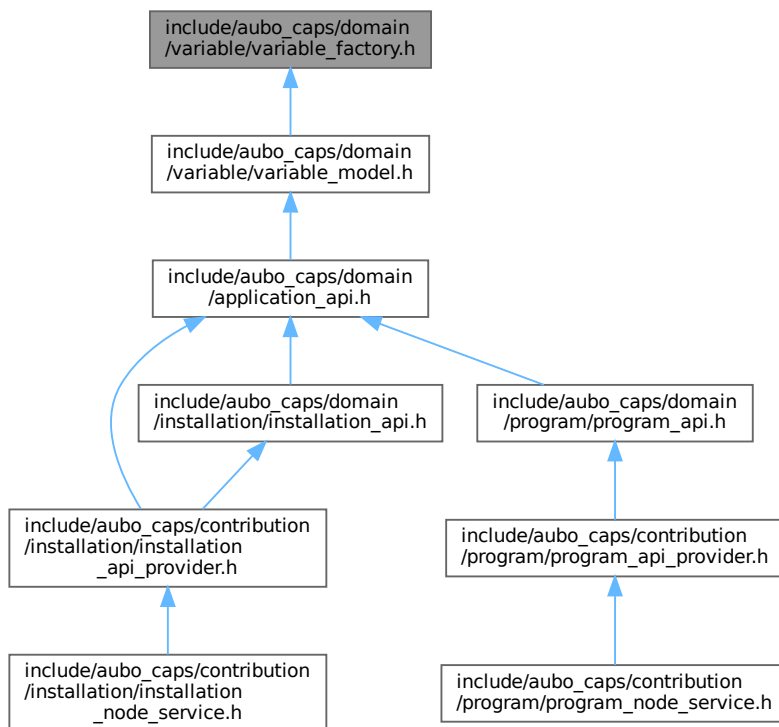
#include <auco_caps/domain/value/expression/expression.h>
#include <auco_caps/domain/variable/global_variable.h>
#include <auco_caps/domain/variable/feature_variable.h>

```

variable_factory.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::VariableFactory`

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (VariableFactory)`

12.247 variable_factory.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_VARIABLE_FACTORY_H
00002 #define AUBO_SCOPE_VARIABLE_FACTORY_H
00003
00004 #include <aubo_caps/domain/value/expression/expression.h>
00005 #include <aubo_caps/domain/variable/global_variable.h>
00006 #include <aubo_caps/domain/variable/feature_variable.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(VariableFactory);
00011
00012 /**
00013  * @ingroup ApplicationApi
00014  * <p>
00015  * This interface supplies methods to create variable objects.
00016  * </p>
00017  *

```



```

00018 * <b>NOTE:</b> Only use this functionality in a AuboCap program node (not
00019 * relevant for an installation node).
00020 */
00021 class ARCS_ABI_EXPORT VariableFactory
00022 {
00023 public:
00024     VariableFactory(VariableFactory &f);
00025     VariableFactory(VariableFactory &&f);
00026     ~VariableFactory();
00027
00028     /**
00029      * @ingroup VariableFactory
00030      * \chinese
00031      * <p>
00032      * 创建全局变量。在 AuboScope 中注册后将对所有程序节点可见。
00033      * </p>
00034      * @param suggested_name 建议的变量名称。
00035      * @return GlobalVariable。
00036      * @throws VariableException 名称非法时抛出。
00037      * \endchinese
00038      * \english
00039      * <p>
00040      * Creates a global variable. Registering the variable in AuboScope (by
00041      * storing it in a {@link DataModel} instance or using it for the
00042      * configuration of a built-in AuboScope program node) will make it
00043      * accessible to all program nodes.
00044      * </p>
00045      *
00046      * @param suggested_name Suggested name of the variable.
00047      * @return GlobalVariable.
00048      * @throws VariableException In case of an illegal name.
00049      * \endenglish
00050      */
00051     GlobalVariablePtr createGlobalVariable(const std::string &suggested_name);
00052
00053     /**
00054      * @ingroup VariableFactory
00055      * \chinese
00056      * <p> 创建带初始值的全局变量。</p>
00057      * @param suggested_name 建议的变量名称。
00058      * @param initialValue 初始化表达式。
00059      * @return GlobalVariable。
00060      * \endchinese
00061      * \english
00062      * <p>
00063      * Creates a global variable with an initial value.
00064      * </p>
00065      * @param suggested_name Suggested name of the variable.
00066      * @param initialValue The expression that will be evaluated in order to
00067      * initialize the value of the variable.
00068      * @return GlobalVariable.
00069      * @throws VariableException In case of an illegal name.
00070      * \endenglish
00071      */
00072     GlobalVariablePtr createGlobalVariable(const std::string &suggested_name,
00073                                           ExpressionPtr initialValue);
00074
00075     /**
00076      * @ingroup VariableFactory
00077      * \chinese
00078      * <p> 创建特征变量。</p>
00079      * @param suggested_name 建议的变量名称。
00080      * @param pose 特征的位姿。
00081      * @return FeatureVariable。
00082      * @throws VariableException 名称非法时抛出。
00083      * \endchinese
00084      * \english
00085      * <p>
00086      * Creates a feature variable.
00087      * </p>
00088      * @param suggested_name Suggested name of the variable.
00089      * @param pose the pose of the feature.
00090      * @return FeatureVariable.
00091      * @throws VariableException In case of an illegal name.
00092      * \endenglish
00093      */
00094     FeatureVariablePtr createFeatureVariable(const std::string &suggested_name,
00095                                             const std::vector<double> &pose);
00096
00097 private:
00098     friend class DataSwitch;
00099     VariableFactory();
00100     void *d_{ nullptr };
00101 };
00102
00103 } // namespace aubo_scope
00104 } // namespace arcs

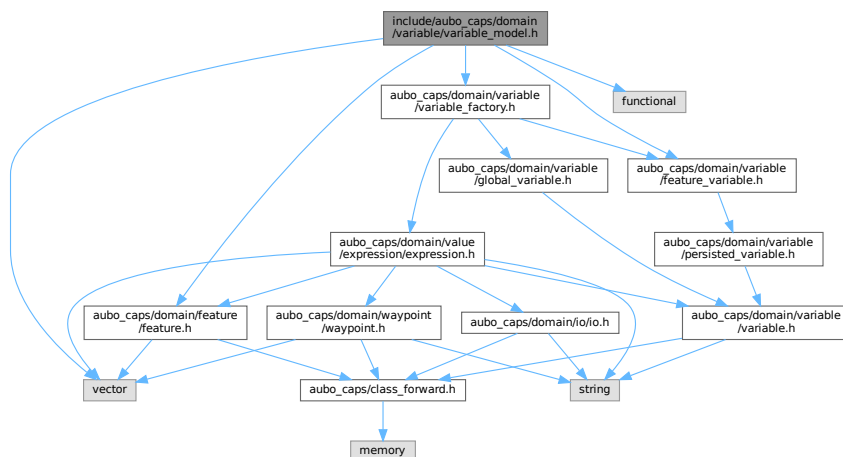
```

```
00105 #endif // AUBO_SCOPE_VARIABLE_FACTORY_H
```

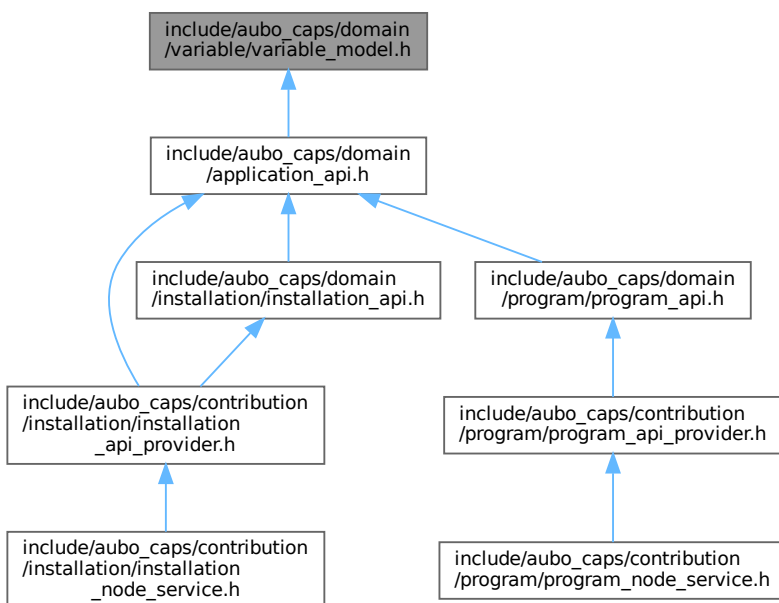
12.248 include/aubo_caps/domain/variable/variable_model.h 文件参考

```
#include <vector>
#include <functional>
#include <aubo_caps/domain/feature/feature.h>
#include <aubo_caps/domain/variable/variable_factory.h>
#include <aubo_caps/domain/variable/feature_variable.h>
```

variable_model.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::VariableModel`

变量模型提供返回系统中当前已注册变量的方法。变量在存储到 *DataModel* 实例中或用于配置内置 *AuboScope* 程序节点时即被注册。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (VariableModel)`

12.249 variable__model.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_VARIABLE_MODEL_H
00002 #define AUBO_SCOPE_VARIABLE_MODEL_H
00003
00004 #include <vector>
00005 #include <functional>
00006 #include <aubo_caps/domain/feature/feature.h>
00007 #include <aubo_caps/domain/variable/variable_factory.h>
00008 #include <aubo_caps/domain/variable/feature_variable.h>
00009
00010 namespace arcs {
00011 namespace aubo_scope {
00012 ARCS_CLASS_FORWARD(VariableModel);
00013
00014 /**
00015  * @ingroup ApplicationApi
00016  * \chinese
00017  * 变量模型
00018  * 提供返回系统中当前已注册变量的方法。变量在存储到 {@link DataModel} 实例中或
00019  * 用于配置内置 AuboScope 程序节点时即被注册。
00020  * \endchinese
00021  * \english
00022  * VariableModel
00023  * Provides methods that returns the variables that currently are registered in
00024  * the system. A variable is registered if it has been stored in a {@link
00025  * DataModel} instance or used for the configuration of a built-in AuboScope
00026  * program node.
00027  * \endenglish
00028  */
00029 class ARCS_ABI_EXPORT VariableModel
00030 {
00031 public:
00032     VariableModel(VariableModel &f);
00033     VariableModel(VariableModel &&f);
00034     ~VariableModel();
00035
00036     /**
00037      * @ingroup VariableModel
00038      * \chinese
00039      * 获取系统中当前已注册的所有变量
00040      * @return 已注册变量的集合。
00041      * \endchinese
00042      * \english
00043      * Returns the collection of all variables that currently are registered in
00044      * the system.
00045      * @return the collection of all registered variables.
00046      * \endenglish
00047      */
00048     std::vector<VariablePtr> getAll();
00049
00050     VariablePtr get(const std::string &name);
00051
00052     /**
00053      * @ingroup VariableModel
00054      * \chinese
00055      * 使用过滤器获取已注册变量的子集
00056      * @param filter 过滤器。
00057      * @return 被过滤器接受的变量集合。
00058      * \endchinese
00059      * \english
00060      * Get a subset of all the variables registered in the system using a filter.
00061      *

```

```

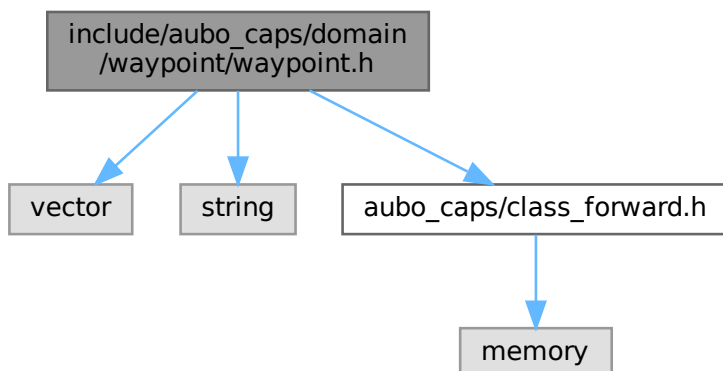
00062     * @param filter see {@link VariableFilterFactory} for examples.
00063     * @return the collection of Variables that are accepted by the filter.
00064     * \endenglish
00065     */
00066     std::vector<VariablePtr> get(std::function<bool(VariablePtr)> filter =
00067         [](VariablePtr) { return true; });
00068
00069     /**
00070     * @ingroup VariableModel
00071     * \chinese
00072     * 获取变量工厂，用于创建变量对象
00073     * @return 变量工厂。
00074     * \endchinese
00075     * \english
00076     * Returns a {@link VariableFactory} to create variable objects.
00077     * @return a {@link VariableFactory} to create variable objects.
00078     * \endenglish
00079     */
00080     VariableFactoryPtr getVariableFactory();
00081
00082     /**
00083     * @ingroup VariableModel
00084     * \chinese
00085     * 获取特征对应的特征变量
00086     * @param feature 带有变量的特征。
00087     * @return 特征的变量。
00088     * @throws IllegalArgumentException 如果特征不在安装中或特征没有变量。
00089     * \endchinese
00090     * \english
00091     * Get the feature variable for a {@link Feature}.
00092     *
00093     * @param feature the feature with a variable.
00094     * @return the variable for the feature.
00095     * @throws IllegalArgumentException If feature not is present in the
00096     * installation or feature has no variable, see {@link Feature#isVariable()}.
00097     * \endenglish
00098     */
00099     FeatureVariablePtr getFeatureVariable(FeaturePtr feature);
00100
00101     /**
00102     * @ingroup VariableModel
00103     * \chinese
00104     * 注册变更回调
00105     * @param receiver 接收者标识。
00106     * @param slot 回调函数。
00107     * @return 连接标识符。
00108     * \endchinese
00109     * \english
00110     * Register callback for changed.
00111     *
00112     * @param receiver the receiver identifier.
00113     * @param slot the callback function.
00114     * @return the connection identifier.
00115     * \endenglish
00116     */
00117     std::string connectedTo(const std::string &receiver,
00118         const std::function<void(VariableModel *)> &slot);
00119
00120     /**
00121     * @ingroup VariableModel
00122     * \chinese
00123     * 断开回调连接
00124     * @param receiver 接收者标识。
00125     * \endchinese
00126     * \english
00127     * Disconnect a callback.
00128     * @param receiver the receiver identifier.
00129     * \endenglish
00130     */
00130     void disconnectFrom(const std::string &receiver);
00131
00132 private:
00133     friend class DataSwitch;
00134     VariableModel();
00135     void *d_{ nullptr };
00136 };
00137
00138 } // namespace aubo_scope
00139 } // namespace arcs
00140 #endif // AUBO_SCOPE_VARIABLE_MODEL_H

```

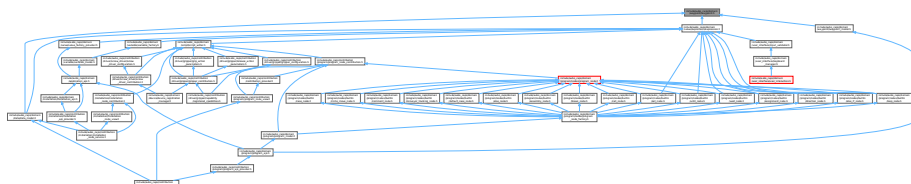
12.250 include/aubo_caps/domain/waypoint/waypoint.h 文件参考

```
#include <vector>
#include <string>
#include <aubo_caps/class_forward.h>
```

waypoint.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class [arcs::aubo_scope::Waypoint](#)
路点接口此接口组合了定义机器人位置的参数。

命名空间

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

函数

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(Waypoint\)](#)

12.251 waypoint.h

[浏览该文件的文档.](#)

```
00001 #ifndef AUBO_SCOPE_WAYPOINT_H
00002 #define AUBO_SCOPE_WAYPOINT_H
00003
00004 #include <vector>
```

```

00005 #include <string>
00006 #include <aubo_caps/class_forward.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(Waypoint);
00011
00012 /**
00013  * @ingroup ProgramApi
00014  * \chinese
00015  * 路点接口
00016  * 此接口组合了定义机器人位置的参数。
00017  * \endchinese
00018  * \english
00019  * Waypoint
00020  * This interface composes parameters that define a position of the robot.
00021  * \endenglish
00022  */
00023 class ARCS_ABI_EXPORT Waypoint
00024 {
00025 public:
00026     Waypoint(Waypoint &f);
00027     Waypoint(Waypoint &&f);
00028     ~Waypoint();
00029
00030     /**
00031      * @ingroup Waypoint
00032      * \chinese
00033      * 获取路点显示名称
00034      * @return 显示名称。
00035      * \endchinese
00036      * \english
00037      * Returns the display name of the Waypoint.
00038      * @return Display name.
00039      * \endenglish
00040      */
00041     std::string getDisplayName();
00042
00043     /**
00044      * @ingroup Waypoint
00045      * \chinese
00046      * 获取机器人位置的位姿（包含 TCP 偏移）
00047      * @return 位姿。
00048      * \endchinese
00049      * \english
00050      * Returns the pose for the position of the robot (including the TCP offset).
00051      * @return the pose for the position of the robot.
00052      * \endenglish
00053      */
00054     std::vector<double> getPose();
00055
00056     /**
00057      * @ingroup Waypoint
00058      * \chinese
00059      * 获取与机器人位姿对应的关节位置（考虑 TCP 偏移）
00060      * @return 关节位置。
00061      * \endchinese
00062      * \english
00063      * Returns the joint positions corresponding to the pose of the robot (when
00064      * taking the TCP offset into account).
00065      * @return the joint positions.
00066      * \endenglish
00067      */
00068     std::vector<double> getJointPositions();
00069
00070     /**
00071      * @ingroup Waypoint
00072      * \chinese
00073      * 获取 TCP 偏移位姿（用于定义机器人位置）
00074      * @return TCP 偏移。
00075      * \endchinese
00076      * \english
00077      * Returns the pose for the TCP offset (used when defining the robot
00078      * position).
00079      * @return the TCP offset.
00080      * \endenglish
00081      */
00082     std::vector<double> getTcpOffset();
00083
00084     /**
00085      * @ingroup Waypoint
00086      * \chinese
00087      * 检查路点是否已定义
00088      * @return 是否已定义。
00089      * \endchinese
00090      * \english
00091      * Returns whether the Waypoint is defined.

```

```

00092     * @return <code>true</code> if defined.
00093     * \endenglish
00094     */
00095     bool isDefined();
00096
00097     /**
00098     * @ingroup Waypoint
00099     * \chinese
00100     * 获取哈希码
00101     * @return 哈希码。
00102     * \endchinese
00103     * \english
00104     * Returns the hash code of the Waypoint.
00105     * @return the hash code.
00106     * \endenglish
00107     */
00108     size_t hashCode();
00109
00110 private:
00111     friend class DataSwitch;
00112     Waypoint();
00113     void *d_{ nullptr };
00114 };
00115
00116 } // namespace aubo_scope
00117 } // namespace arcs
00118
00119 #endif // AUBO_SCOPE_WAYPOINT_H

```

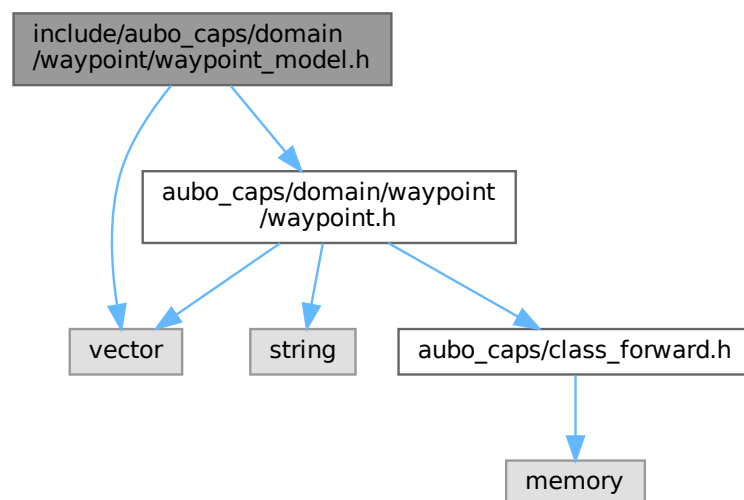
12.252 include/auco_caps/domain/waypoint/waypoint__ model.h 文件参考

```

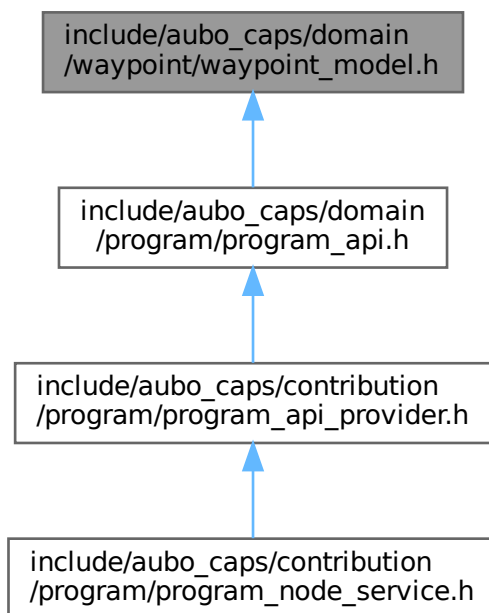
#include <vector>
#include <auco_caps/domain/waypoint/waypoint.h>

```

waypoint_model.h 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- class `arcs::aubo_scope::WaypointModel`
路点模型提供从当前机器人获取路点的方法。

命名空间

- namespace `arcs`
- namespace `arcs::aubo_scope`

函数

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (WaypointModel)`

12.253 waypoint_model.h

[浏览该文件的文档.](#)

```

00001 #ifndef AUBO_SCOPE_WAYPOINT_MODEL_H
00002 #define AUBO_SCOPE_WAYPOINT_MODEL_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/waypoint/waypoint.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(WaypointModel);
00010
00011 /**
00012 * @ingroup ProgramApi
00013 * \chinese
00014 * 路点模型
00015 * 提供从当前机器人获取路点的方法。
00016 * \endchinese
00017 * \english

```



```

00018 * WaypointModel
00019 * Provides methods that returns Waypoints from the current robot installation.
00020 * \endenglish
00021 */
00022 class ARCS_ABI_EXPORT WaypointModel
00023 {
00024 public:
00025     WaypointModel(WaypointModel &f);
00026     WaypointModel(WaypointModel &&f);
00027     ~WaypointModel();
00028
00029     /**
00030      * @ingroup WaypointModel
00031      * \chinese
00032      * 获取当前安装中的几何路点
00033      * @return 当前安装中的几何路点集合。
00034      * \endchinese
00035      * \english
00036      * Returns the geometric Waypoints of the current installation.
00037      * @return the geometric Waypoints of the current installation.
00038      * \endenglish
00039      */
00040     std::vector<WaypointPtr> getWaypoints();
00041
00042     /**
00043      * @ingroup WaypointModel
00044      * \chinese
00045      * 根据名称获取路点
00046      * @param name 路点名称。
00047      * @return 对应名称的路点。
00048      * \endchinese
00049      * \english
00050      * Returns the Waypoint by name.
00051      * @param name the name of the Waypoint.
00052      * @return the Waypoint with the given name.
00053      * \endenglish
00054      */
00055     WaypointPtr getByName(const std::string &name);
00056
00057     /**
00058      * @ingroup WaypointModel
00059      * \chinese
00060      * 向当前 AuboScope 安装中添加一个路点。添加后终端用户可选择该路点，但不可修改。
00061      *
00062      * @param suggestedName 建议的路点名称。有效名称必须匹配正则表达式
00063      * [a-zA-Z][a-zA-Z0-9_]{0,14}，总共 15 个字符。最终名称可从返回的 Waypoint 实例获取。
00064      * @param pose 路点相对于机器人基座的位姿。
00065      * @return 在 AuboScope 中创建并注册的路点。
00066      * @throws WaypointAlreadyAddedException 如果之前已用相同的 <code>idKey</code> 标
00067      * 识符添加过路点。使用 {@link #getWaypoint(String)} 检查路点是否已添加到当前安装中。
00068      * 使用 {@link #updateWaypoint(String, Pose)} 来更新路点。
00069      * @throws IllegalWaypointNameException 如果建议的名称不匹配所需的正则表达式。
00070      * \endchinese
00071      * \english
00072      * Add a Waypoint to the current AuboScope installation. This makes it
00073      * selectable by the end user. The Waypoint is not modifiable by the end
00074      * user.
00075      *
00076      * @param suggestedName Suggested name for the Waypoint. Valid names must
00077      * match regex [a-zA-Z][a-zA-Z0-9_]{0,14} for a total of 15 characters. The
00078      * final name can be retrieved from the returned Waypoint instance.
00079      * @param pose The pose of the Waypoint with respect to the robot base.
00080      * @return The Waypoint created and registered in AuboScope.
00081      * @throws WaypointAlreadyAddedException If a Waypoint has previously been
00082      * added the same <code>idKey</code> identifier. Use {@link
00083      * #getWaypoint(String)} to check if the Waypoint has already been added to
00084      * the current installation. Use {@link #updateWaypoint(String, Pose)} to
00085      * update the Waypoint.
00086      * @throws IllegalWaypointNameException If the suggested name does not match
00087      * required regex.
00088      * \endenglish
00089      */
00090     WaypointPtr addWaypoint(const std::string &suggestedName,
00091                             const std::vector<double> &pose);
00092
00093     /**
00094      * @ingroup WaypointModel
00095      * \chinese
00096      * 向当前 AuboScope 安装中添加一个包含关节角度和偏移的路点。
00097      *
00098      * @param suggestedName 建议的路点名称。有效名称必须匹配正则表达式
00099      * [a-zA-Z][a-zA-Z0-9_]{0,14}，总共 15 个字符。最终名称可从返回的 Waypoint 实例获取。
00100      * @param pose 路点相对于机器人基座的位姿。
00101      * @param joint 路点相对于机器人基座的关节角度。
00102      * @param offset 路点相对于机器人基座的 TCP 偏移。
00103      * @return 在 AuboScope 中创建并注册的路点。
00104      * @throws WaypointAlreadyAddedException 如果之前已用相同的 <code>idKey</code> 标
00105      * 识符添加过路点。

```

```

00105     * @throws IllegalWaypointNameException 如果建议的名称不匹配所需的正则表达式。
00106     * \endchinese
00107     * \english
00108     * Add a Waypoint to the current AuboScope installation with joint angles
00109     * and offset.
00110     *
00111     * @param suggestedName Suggested name for the Waypoint. Valid names must
00112     * match regex [a-zA-Z][a-zA-Z0-9_]{0,14} for a total of 15 characters. The
00113     * final name can be retrieved from the returned Waypoint instance.
00114     * @param pose The pose of the Waypoint with respect to the robot base.
00115     * @param joint The joint of the Waypoint with respect to the robot base.
00116     * @param offset The tcp of the Waypoint with respect to the robot base.
00117     * @return The Waypoint created and registered in AuboScope.
00118     * @throws WaypointAlreadyAddedException If a Waypoint has previously been
00119     * added the same <code>idKey</code> identifier. Use {@link
00120     * #getWaypoint(String)} to check if the Waypoint has already been added to
00121     * the current installation. Use {@link #updateWaypoint(String, Pose)} to
00122     * update the Waypoint.
00123     * @throws IllegalWaypointNameException If the suggested name does not match
00124     * required regex.
00125     * \endenglish
00126     */
00127     WaypointPtr addWaypoint(const std::string &suggestedName,
00128                             const std::vector<double> &pose,
00129                             const std::vector<double> &joint,
00130                             const std::vector<double> &offset);
00131
00132     /**
00133     * @ingroup WaypointModel
00134     * \chinese
00135     * 从此 AuboCap 中移除已添加的路点。使用该路点的程序节点将变为未定义状态，因为路点
00136     * 不再可解析。
00137     *
00138     * @param name 添加路点时使用的名称。
00139     * @throws WaypointNotFoundException 如果没有与提供的 <code>idKey</code> 对应
00140     * 的路点存在。
00141     * \endchinese
00142     * \english
00143     * Remove a Waypoint added by this AuboCap from AuboScope. Program nodes
00144     * using the Waypoint will be become undefined because the Waypoint is no
00145     * longer resolvable.
00146     *
00147     * @param name The key used to add the Waypoint with.
00148     * @throws WaypointNotFoundException If no Waypoint exists with the provided
00149     * <code>idKey</code>.
00150     * \endenglish
00151     */
00152     void removeWaypoint(const std::string &name);
00153
00154     /**
00155     * @ingroup WaypointModel
00156     * \chinese
00157     * 根据名称重命名模型中的路点
00158     * @param name 原始名称。
00159     * @param newName 新名称。
00160     * \endchinese
00161     * \english
00162     * Rename the waypoint in the model by the name.
00163     * @param name the original name.
00164     * @param newName the new name.
00165     * \endenglish
00166     */
00167     void renameWaypoint(const std::string &name, const std::string &newName);
00168 private:
00169     friend class DataSwitch;
00170     WaypointModel();
00171     void *d_{ nullptr };
00172 };
00173
00174 } // namespace aubo_scope
00175 } // namespace arcs
00176
00177 #endif // AUBO_SCOPE_WAYPOINT_MODEL_H

```

Index

- ~AnalogIo
 - arcs::aubo_scope::AnalogIo, [119](#)
- ~ApplicationApi
 - arcs::aubo_scope::ApplicationApi, [123](#)
- ~AssemblyNode
 - arcs::aubo_scope::AssemblyNode, [128](#)
- ~AssignmentNode
 - arcs::aubo_scope::AssignmentNode, [132](#)
- ~AuboCapInfo
 - arcs::aubo_scope::AuboCapInfo, [133](#)
- ~BaseFeature
 - arcs::aubo_scope::BaseFeature, [136](#)
- ~BoolRegister
 - arcs::aubo_scope::BoolRegister, [138](#)
- ~BreakNode
 - arcs::aubo_scope::BreakNode, [141](#)
- ~CallNode
 - arcs::aubo_scope::CallNode, [143](#)
- ~CaseNode
 - arcs::aubo_scope::CaseNode, [146](#)
- ~CircleMoveNode
 - arcs::aubo_scope::CircleMoveNode, [149](#)
- ~CommentNode
 - arcs::aubo_scope::CommentNode, [151](#)
- ~ContributionProvider
 - arcs::aubo_scope::ContributionProvider< T >, [153](#)
- ~ConveyorTrackingNode
 - arcs::aubo_scope::ConveyorTrackingNode, [156](#)
- ~DataModel
 - arcs::aubo_scope::DataModel, [160](#)
- ~DefaultCaseNode
 - arcs::aubo_scope::DefaultCaseNode, [169](#)
- ~Device
 - arcs::aubo_scope::Device, [171](#)
- ~DeviceManager
 - arcs::aubo_scope::DeviceManager, [172](#)
- ~DeviceRegistrationManager
 - arcs::aubo_scope::DeviceRegistrationManager, [173](#)
- ~DigitalIo
 - arcs::aubo_scope::DigitalIo, [176](#)
- ~DirectionNode
 - arcs::aubo_scope::DirectionNode, [182](#)
- ~DoubleRegister
 - arcs::aubo_scope::DoubleRegister, [189](#)
- ~DriveScrewNotOKParameters
 - arcs::aubo_scope::DriveScrewNotOKParameters, [189](#)
- ~DriveScrewOKParameters
 - arcs::aubo_scope::DriveScrewOKParameters, [190](#)
- ~ElseIfNode
 - arcs::aubo_scope::ElseIfNode, [192](#)
- ~ElseNode
 - arcs::aubo_scope::ElseNode, [195](#)
- ~EtherNetIP
 - arcs::aubo_scope::EtherNetIP, [199](#)
- ~Expression
 - arcs::aubo_scope::Expression, [203](#)
- ~Feature
 - arcs::aubo_scope::Feature, [206](#)
- ~FeatureContributionModel
 - arcs::aubo_scope::FeatureContributionModel, [209](#)
- ~FeatureLine
 - arcs::aubo_scope::FeatureLine, [214](#)
- ~FeatureModel
 - arcs::aubo_scope::FeatureModel, [217](#)
- ~FeaturePlane
 - arcs::aubo_scope::FeaturePlane, [221](#)
- ~FeaturePoint
 - arcs::aubo_scope::FeaturePoint, [224](#)
- ~FeatureTool
 - arcs::aubo_scope::FeatureTool, [227](#)
- ~FeatureVariable
 - arcs::aubo_scope::FeatureVariable, [230](#)
- ~FeedScrewParameters
 - arcs::aubo_scope::FeedScrewParameters, [232](#)
- ~FolderNode
 - arcs::aubo_scope::FolderNode, [234](#)
- ~ForceNode
 - arcs::aubo_scope::ForceNode, [237](#)
- ~ForceOptionNode
 - arcs::aubo_scope::ForceOptionNode, [240](#)
- ~Function
 - arcs::aubo_scope::Function, [242](#)
- ~FunctionModel
 - arcs::aubo_scope::FunctionModel, [245](#)
- ~GlobalVariable
 - arcs::aubo_scope::GlobalVariable, [249](#)
- ~GripActionParameters
 - arcs::aubo_scope::GripActionParameters, [250](#)
- ~GripForceCapability
 - arcs::aubo_scope::GripForceCapability, [252](#)
- ~GripVacuumCapability
 - arcs::aubo_scope::GripVacuumCapability, [268](#)
- ~GripperApiProvider
 - arcs::aubo_scope::GripperApiProvider, [253](#)
- ~GripperConfiguration

- arcs::aubo_scope::GripperConfiguration, 254
- ~GripperContribution
 - arcs::aubo_scope::GripperContribution, 260
- ~GripperDevice
 - arcs::aubo_scope::GripperDevice, 264
- ~GripperNode
 - arcs::aubo_scope::GripperNode, 267
- ~HaltNode
 - arcs::aubo_scope::HaltNode, 271
- ~IfNode
 - arcs::aubo_scope::IfNode, 274
- ~InstallationApi
 - arcs::aubo_scope::InstallationApi, 279
- ~InstallationApiProvider
 - arcs::aubo_scope::InstallationApiProvider, 281
- ~InstallationContributionConfiguration
 - arcs::aubo_scope::InstallationContributionConfiguration, 282
- ~InstallationCreationContext
 - arcs::aubo_scope::InstallationCreationContext, 284
- ~InstallationNodeContribution
 - arcs::aubo_scope::InstallationNodeContribution, 285
- ~InstallationNodeService
 - arcs::aubo_scope::InstallationNodeService, 286
- ~InstallationNodeView
 - arcs::aubo_scope::InstallationNodeView, 288
- ~IntegerRegister
 - arcs::aubo_scope::IntegerRegister, 293
- ~Io
 - arcs::aubo_scope::Io, 296
- ~IoModel
 - arcs::aubo_scope::IoModel, 301
- ~KeyboardManager
 - arcs::aubo_scope::KeyboardManager, 304
- ~LoopNode
 - arcs::aubo_scope::LoopNode, 310
- ~ModbusIo
 - arcs::aubo_scope::ModbusIo, 315
- ~MoveNode
 - arcs::aubo_scope::MoveNode, 321
- ~OfflineTrackNode
 - arcs::aubo_scope::OfflineTrackNode, 327
- ~PalletNode
 - arcs::aubo_scope::PalletNode, 330
- ~Payload
 - arcs::aubo_scope::Payload, 332
- ~PayloadContributionModel
 - arcs::aubo_scope::PayloadContributionModel, 336
- ~PayloadModel
 - arcs::aubo_scope::PayloadModel, 342
- ~PersistedVariable
 - arcs::aubo_scope::PersistedVariable, 346
- ~PopupManager
 - arcs::aubo_scope::PopupManager, 348
- ~PopupNode
 - arcs::aubo_scope::PopupNode, 352
- ~PrepareToStartScrewdriverParameters
 - arcs::aubo_scope::PrepareToStartScrewdriverParameters, 354
- ~Profinet
 - arcs::aubo_scope::Profinet, 357
- ~ProgramApi
 - arcs::aubo_scope::ProgramApi, 362
- ~ProgramApiProvider
 - arcs::aubo_scope::ProgramApiProvider, 364
- ~ProgramContributionConfiguration
 - arcs::aubo_scope::ProgramContributionConfiguration, 367
- ~ProgramCreationContext
 - arcs::aubo_scope::ProgramCreationContext, 372
- ~ProgramHomeNode
 - arcs::aubo_scope::ProgramHomeNode, 375
- ~ProgramModel
 - arcs::aubo_scope::ProgramModel, 377
- ~ProgramNode
 - arcs::aubo_scope::ProgramNode, 381
- ~ProgramNodeContribution
 - arcs::aubo_scope::ProgramNodeContribution, 382
- ~ProgramNodeFactory
 - arcs::aubo_scope::ProgramNodeFactory, 386
- ~ProgramNodeService
 - arcs::aubo_scope::ProgramNodeService, 391
- ~ProgramNodeView
 - arcs::aubo_scope::ProgramNodeView, 393
- ~ProgramSelectionParameters
 - arcs::aubo_scope::ProgramSelectionParameters, 394
- ~RecordTrackNode
 - arcs::aubo_scope::RecordTrackNode, 397
- ~Register
 - arcs::aubo_scope::Register, 399
- ~RegisteredCapabilities
 - arcs::aubo_scope::RegisteredCapabilities, 399
- ~ReleaseActionParameters
 - arcs::aubo_scope::ReleaseActionParameters, 401
- ~RenderInterface
 - arcs::aubo_scope::RenderInterface, 406
- ~ScrewdriverApiProvider
 - arcs::aubo_scope::ScrewdriverApiProvider, 417
- ~ScrewdriverConfiguration
 - arcs::aubo_scope::ScrewdriverConfiguration, 418
- ~ScrewdriverContribution
 - arcs::aubo_scope::ScrewdriverContribution, 424
- ~ScrewdriverParameters
 - arcs::aubo_scope::ScrewdriverParameters, 426
- ~ScrewdriverProgram
 - arcs::aubo_scope::ScrewdriverProgram, 427
- ~ScrewdriverReadyParameters
 - arcs::aubo_scope::ScrewdriverReadyParameters, 428
- ~ScrewdrivingNode
 - arcs::aubo_scope::ScrewdrivingNode, 430
- ~ScriptNode
 - arcs::aubo_scope::ScriptNode, 433

- ~ScriptWriter
 - arcs::aubo_scope::ScriptWriter, [436](#)
- ~SeekNode
 - arcs::aubo_scope::SeekNode, [444](#)
- ~SetNode
 - arcs::aubo_scope::SetNode, [448](#)
- ~SetPayloadNode
 - arcs::aubo_scope::SetPayloadNode, [454](#)
- ~SpeedCapability
 - arcs::aubo_scope::SpeedCapability, [455](#)
- ~SubProgNode
 - arcs::aubo_scope::SubProgNode, [460](#)
- ~SwitchNode
 - arcs::aubo_scope::SwitchNode, [463](#)
- ~SystemApi
 - arcs::aubo_scope::SystemApi, [466](#)
- ~TCP
 - arcs::aubo_scope::TCP, [468](#)
- ~TcpContributionModel
 - arcs::aubo_scope::TcpContributionModel, [471](#)
- ~TcpModel
 - arcs::aubo_scope::TcpModel, [475](#)
- ~ThreadNode
 - arcs::aubo_scope::ThreadNode, [479](#)
- ~TimerNode
 - arcs::aubo_scope::TimerNode, [482](#)
- ~TreeNode
 - arcs::aubo_scope::TreeNode, [484](#)
- ~UndoRedoManager
 - arcs::aubo_scope::UndoRedoManager, [492](#)
- ~UntilNode
 - arcs::aubo_scope::UntilNode, [498](#)
- ~UserInteraction
 - arcs::aubo_scope::UserInteraction, [505](#)
- ~UserInterfaceApi
 - arcs::aubo_scope::UserInterfaceApi, [511](#)
- ~ValueFactory
 - arcs::aubo_scope::ValueFactory, [513](#)
- ~Variable
 - arcs::aubo_scope::Variable, [516](#)
- ~VariableFactory
 - arcs::aubo_scope::VariableFactory, [519](#)
- ~VariableModel
 - arcs::aubo_scope::VariableModel, [523](#)
- ~ViewApiProvider
 - arcs::aubo_scope::ViewApiProvider, [527](#)
- ~VisualLocalizationProgramNode
 - arcs::aubo_scope::VisualLocalizationProgramNode, [530](#)
- ~WaitNode
 - arcs::aubo_scope::WaitNode, [534](#)
- ~Waypoint
 - arcs::aubo_scope::Waypoint, [540](#)
- ~WaypointModel
 - arcs::aubo_scope::WaypointModel, [543](#)
- ~WaypointNode
 - arcs::aubo_scope::WaypointNode, [551](#)
- ~WidthCapability
 - arcs::aubo_scope::WidthCapability, [557](#)
- ABANDON_TRACKING
 - arcs::aubo_scope::ConveyorTrackingNode, [155](#)
- addAssemblyUntilNode
 - arcs::aubo_scope::AssemblyNode, [128](#)
- addChild
 - arcs::aubo_scope::TreeNode, [484](#)
- addCoordinate
 - arcs::aubo_scope::RenderInterface, [406](#)
- addCube
 - arcs::aubo_scope::RenderInterface, [406–408](#)
- addFeature
 - arcs::aubo_scope::FeatureContributionModel, [209](#)
- addFunction
 - arcs::aubo_scope::FunctionModel, [245](#)
- addLine
 - arcs::aubo_scope::RenderInterface, [408, 409](#)
- addMesh
 - arcs::aubo_scope::RenderInterface, [409](#)
- addPayload
 - arcs::aubo_scope::PayloadContributionModel, [337](#)
- addPlane
 - arcs::aubo_scope::RenderInterface, [409, 410](#)
- addPointCloud
 - arcs::aubo_scope::RenderInterface, [410](#)
- addScrewdriverProgram
 - arcs::aubo_scope::ScrewdriverConfiguration, [418](#)
- addScrewdriverPrograms
 - arcs::aubo_scope::ScrewdriverConfiguration, [418](#)
- addTcp
 - arcs::aubo_scope::TcpContributionModel, [471](#)
- addWaypoint
 - arcs::aubo_scope::RenderInterface, [410](#)
 - arcs::aubo_scope::WaypointModel, [544](#)
- All
 - arcs::aubo_scope::RenderInterface, [405](#)
- ALWAYS
 - arcs::aubo_scope::LoopNode, [309](#)
- ANALOG
 - arcs::aubo_scope::Io, [295](#)
- ANALOG_INPUT
 - arcs::aubo_scope::UntilNode, [497](#)
- ANALOG_INPUT_CURRENT
 - arcs::aubo_scope::WaitNode, [533](#)
- ANALOG_INPUT_VOLTAGE
 - arcs::aubo_scope::WaitNode, [533](#)
- ANALOG_OUTPUT_CURRENT
 - arcs::aubo_scope::SetNode, [447](#)
- ANALOG_OUTPUT_VOLTAGE
 - arcs::aubo_scope::SetNode, [447](#)
- AnalogIo
 - arcs::aubo_scope::AnalogIo, [119, 120](#)
- AnalogIoFilter
 - arcs::aubo_scope::AnalogIoFilter, [121](#)
- Angular
 - arcs::aubo_scope::UntilNode, [497](#)
- anonyFunction
 - arcs::aubo_scope::ScriptWriter, [436](#)

- append
 - arcs::aubo_scope::Expression, 203
- appendFeature
 - arcs::aubo_scope::Expression, 203
- appendIo
 - arcs::aubo_scope::Expression, 203
- appendLine
 - arcs::aubo_scope::ScriptWriter, 436
- appendRaw
 - arcs::aubo_scope::ScriptWriter, 436
- appendVariable
 - arcs::aubo_scope::Expression, 203
- appendVectorDouble
 - arcs::aubo_scope::ScriptWriter, 436
- appendWaypoint
 - arcs::aubo_scope::Expression, 203
- ApplicationApi
 - arcs::aubo_scope::ApplicationApi, 123, 124
- ApplicationApi(应用程序), 65
 - getDeviceManager, 66
 - getFeatureModel, 66
 - getIoModel, 66
 - getPayloadModel, 66
 - getTcpModel, 66
 - getValueFactory, 67
 - getVariableModel, 67
- arcs, 97
- arcs::aubo_scope, 97
 - ARCS_CLASS_FORWARD, 105–116
 - ContributionProviderPtr, 104
 - IMPERIAL, 105
 - LOOSEN, 105
 - METRIC, 105
 - RobotMovementCallback, 104
 - RobotPositionCallback, 104
 - ScrewdriverOperationType, 105
 - TIGHTEN, 105
 - UnitType, 105
 - WidgetHandle, 104
- arcs::aubo_scope::AnalogIo, 117
 - ~AnalogIo, 119
 - AnalogIo, 119, 120
 - d_, 121
 - DataSwitch, 120
 - getMaxRangeValue, 120
 - getMinRangeValue, 120
 - getValue, 120
 - isCurrent, 120
 - isVoltage, 120
 - setValue, 120
- arcs::aubo_scope::AnalogIoFilter, 121
 - AnalogIoFilter, 121
 - interface_, 121
 - is_input_, 121
 - operator(), 121
- arcs::aubo_scope::ApplicationApi, 122
 - ~ApplicationApi, 123
 - ApplicationApi, 123, 124
 - d_, 125
 - DataSwitch, 125
 - getInstallationNode, 124, 125
 - getRobotProxy, 125
- arcs::aubo_scope::AssemblyNode, 125
 - ~AssemblyNode, 128
 - addAssemblyUntilNode, 128
 - AssemblyNode, 127, 128
 - COMPLIANT_INSERTION, 127
 - ConfigType, 127
 - d_, 129
 - DataSwitch, 128
 - HOLE_SEARCH, 127
 - NONE, 127
- arcs::aubo_scope::AssignmentNode, 129
 - ~AssignmentNode, 132
 - AssignmentNode, 131, 132
 - ConfigType, 130
 - d_, 133
 - DataSwitch, 133
 - EXPRESSION, 130
 - getConfigType, 132
 - getExpression, 132
 - getVariable, 132
 - INPUT_BOOL, 131
 - INPUT_FLOAT, 131
 - INPUT_INTEGER, 131
 - INPUT_STRING, 131
 - InputMode, 130
 - OPERATOR, 130
 - setConfigType, 132
 - setExpression, 132
 - setVariable, 132
- arcs::aubo_scope::AuboCapInfo, 133
 - ~AuboCapInfo, 133
 - getName, 133
 - getSymbolicName, 133
 - getVendor, 134
- arcs::aubo_scope::BaseFeature, 134
 - ~BaseFeature, 136
 - BaseFeature, 135, 136
 - d_, 137
 - DataSwitch, 136
- arcs::aubo_scope::BoolRegister, 137
 - ~BoolRegister, 138
- arcs::aubo_scope::BreakNode, 139
 - ~BreakNode, 141
 - BreakNode, 140, 141
 - d_, 141
 - DataSwitch, 141
- arcs::aubo_scope::CallNode, 141
 - ~CallNode, 143
 - CallNode, 143
 - d_, 144
 - DataSwitch, 144
- arcs::aubo_scope::CaseNode, 144
 - ~CaseNode, 146
 - CaseNode, 145, 146

- d_, 146
- DataSwitch, 146
- arcs::aubo_scope::CircleMoveNode, 147
 - ~CircleMoveNode, 149
 - CircleMoveNode, 148, 149
 - d_, 149
 - DataSwitch, 149
- arcs::aubo_scope::CommentNode, 149
 - ~CommentNode, 151
 - CommentNode, 151
 - d_, 152
 - DataSwitch, 152
 - getComment, 152
 - setComment, 152
- arcs::aubo_scope::ContributionProvider< T >, 152
 - ~ContributionProvider, 153
 - get, 153
- arcs::aubo_scope::ConveyorTrackingNode, 153
 - ~ConveyorTrackingNode, 156
 - ABANDON_TRACKING, 155
 - ASK, 155
 - CLEAR_QUEUE, 155
 - CONTINUE_TRACKING, 155
 - ConveyorStartActionQueueType, 155
 - ConveyorStartActionType, 155
 - ConveyorTrackingNode, 155, 156
 - ConveyorTrackingTargetType, 155
 - d_, 157
 - DataSwitch, 156
 - NEW_WORKPIECE, 155
 - PREVIOUS_WORKPIECE, 155
 - RESERVE_QUEUE, 155
- arcs::aubo_scope::DataModel, 157
 - ~DataModel, 160
 - d_, 167
 - DataModel, 159, 160
 - DataSwitch, 167
 - get, 160–163
 - getKeys, 163
 - isSet, 163
 - remove, 163
 - set, 164–167
- arcs::aubo_scope::DefaultCaseNode, 167
 - ~DefaultCaseNode, 169
 - d_, 170
 - DataSwitch, 170
 - DefaultCaseNode, 169
- arcs::aubo_scope::DefaultSingletonTraits< Type >, 170
- arcs::aubo_scope::Device, 170
 - ~Device, 171
 - getContributorInfo, 171
 - getDisplayName, 171
 - getId, 171
 - isResolvable, 171
- arcs::aubo_scope::DeviceManager, 172
 - ~DeviceManager, 172
 - getGrippers, 172
 - getScrewDrivers, 172
- arcs::aubo_scope::DeviceRegistrationManager, 173
 - ~DeviceRegistrationManager, 173
 - registerGripperContribution, 173
 - registerScrewDriverContribution, 173
- arcs::aubo_scope::DigitalIo, 174
 - ~DigitalIo, 176
 - d_, 177
 - DataSwitch, 177
 - DigitalIo, 175, 176
 - getValue, 177
 - setValue, 177
- arcs::aubo_scope::DigitalIoFilter, 177
 - DigitalIoFilter, 178
 - interface_, 178
 - is_input_, 178
 - operator(), 178
- arcs::aubo_scope::DirectionNode, 178
 - ~DirectionNode, 182
 - AXIS, 180
 - Axis, 180
 - d_, 184
 - DataSwitch, 184
 - DirectionNode, 181, 182
 - DirectionSelectionType, 180
 - EXPRESSION, 180
 - FEATURE, 181
 - FeatureSelectionType, 180
 - getAxis, 182
 - getDirectionSelectionType, 182
 - getExpression, 182
 - getFeature, 182
 - getFeatureSelectionType, 183
 - getToolAcceleration, 183
 - getToolSpeed, 183
 - getType, 183
 - ParameterType, 181
 - setAxis, 183
 - setDirectionSelectionType, 183
 - setExpression, 183
 - setFeature, 183
 - setFeatureSelectionType, 183
 - setToolAcceleration, 183
 - setToolSpeed, 183
 - setType, 184
 - SHARED, 181
 - SHARED_MOTION, 181
 - TOOL_MOTION, 181
 - UNRESOLVED_FEATURE, 181
 - X_MINUS, 180
 - X_PLUS, 180
 - Y_MINUS, 180
 - Y_PLUS, 180
 - Z_MINUS, 180
 - Z_PLUS, 180
- arcs::aubo_scope::DoubleRangeValidator, 184
 - DoubleRangeValidator, 185
 - getMessage, 186

- is_max_inclusive_, 186
- is_min_inclusive_, 186
- isValid, 186
- max_value_, 186
- min_value_, 186
- range_func_, 187
- range_func_with_flags_, 187
- updateValue, 186
- arcs::aubo_scope::DoubleRegister, 187
 - ~DoubleRegister, 189
- arcs::aubo_scope::DriveScrewNotOKParameters, 189
 - ~DriveScrewNotOKParameters, 189
- arcs::aubo_scope::DriveScrewOKParameters, 189
 - ~DriveScrewOKParameters, 190
- arcs::aubo_scope::ElseIfNode, 190
 - ~ElseIfNode, 192
 - d_, 193
 - DataSwitch, 193
 - ElseIfNode, 191, 192
 - getExpression, 192
 - setExpression, 192
- arcs::aubo_scope::ElseNode, 193
 - ~ElseNode, 195
 - d_, 196
 - DataSwitch, 196
 - ElseNode, 195
- arcs::aubo_scope::EtherNetIP, 196
 - ~EtherNetIP, 199
 - d_, 200
 - DataSwitch, 200
 - EtherNetIP, 198, 199
 - getBitOffset, 199
 - getByteOffset, 199
 - getValue, 199
 - isSlave, 199
 - setValue, 200
- arcs::aubo_scope::EtherNetIPFilter, 200
 - EtherNetIPFilter, 201
 - is_input_, 201
 - operator(), 201
 - type_, 201
- arcs::aubo_scope::Expression, 201
 - ~Expression, 203
 - append, 203
 - appendFeature, 203
 - appendIo, 203
 - appendVariable, 203
 - appendWaypoint, 203
 - d_, 204
 - DataSwitch, 204
 - Expression, 202, 203
 - toString, 204
- arcs::aubo_scope::Feature, 204
 - ~Feature, 206
 - d_, 207
 - DataSwitch, 207
 - Feature, 205, 206
 - getDisplayName, 206
 - getPose, 206
 - hashCode, 206
 - isDefined, 207
 - isJoggable, 207
 - isResolvable, 207
 - isVariable, 207
- arcs::aubo_scope::FeatureContributionModel, 207
 - ~FeatureContributionModel, 209
 - addFeature, 209
 - d_, 211
 - DataSwitch, 211
 - FeatureContributionModel, 208, 209
 - getFeature, 210
 - removeFeature, 210
 - renameFeature, 210
 - updateFeature, 210
- arcs::aubo_scope::FeatureDefinedFilter, 211
 - operator(), 212
- arcs::aubo_scope::FeatureJoggableFilter, 212
 - operator(), 212
- arcs::aubo_scope::FeatureLine, 212
 - ~FeatureLine, 214
 - d_, 215
 - DataSwitch, 215
 - FeatureLine, 214, 215
- arcs::aubo_scope::FeatureModel, 215
 - ~FeatureModel, 217
 - d_, 218
 - DataSwitch, 218
 - FeatureModel, 216, 217
 - getAll, 217
 - getBaseFeature, 217
 - getByName, 217
 - getFeatures, 217
 - getGeomFeture, 218
 - getToolFeature, 218
- arcs::aubo_scope::FeaturePlane, 218
 - ~FeaturePlane, 221
 - d_, 221
 - DataSwitch, 221
 - FeaturePlane, 220, 221
- arcs::aubo_scope::FeaturePoint, 221
 - ~FeaturePoint, 224
 - d_, 224
 - DataSwitch, 224
 - FeaturePoint, 223, 224
- arcs::aubo_scope::FeatureTool, 224
 - ~FeatureTool, 227
 - d_, 227
 - DataSwitch, 227
 - FeatureTool, 226, 227
- arcs::aubo_scope::FeatureVariable, 227
 - ~FeatureVariable, 230
 - d_, 231
 - DataSwitch, 231
 - FeatureVariable, 230
- arcs::aubo_scope::FeatureVariableFilter, 231
 - operator(), 231

- arcs::aubo_scope::FeedScrewParameters, 231
 - ~FeedScrewParameters, 232
- arcs::aubo_scope::FolderNode, 232
 - ~FolderNode, 234
 - d_, 235
 - DataSwitch, 235
 - FolderNode, 233, 234
 - setFolderName, 234
- arcs::aubo_scope::ForceNode, 235
 - ~ForceNode, 237
 - ConfigType, 236
 - d_, 238
 - DataSwitch, 238
 - ForceNode, 237
 - FRAME, 236
 - MOTION, 236
 - POINT, 236
 - SIMPLE, 236
- arcs::aubo_scope::ForceOptionNode, 238
 - ~ForceOptionNode, 240
 - d_, 240
 - DataSwitch, 240
 - ForceOptionNode, 239, 240
- arcs::aubo_scope::Function, 241
 - ~Function, 242
 - d_, 243
 - DataSwitch, 243
 - Function, 241, 242
 - getArguments, 242
 - getContributorInfo, 242
 - getName, 242
- arcs::aubo_scope::FunctionModel, 243
 - ~FunctionModel, 245
 - addFunction, 245
 - d_, 246
 - DataSwitch, 246
 - FunctionModel, 244, 245
 - getFunction, 245
 - getFunctions, 246
 - removeFunction, 246
- arcs::aubo_scope::GlobalVariable, 247
 - ~GlobalVariable, 249
 - d_, 249
 - DataSwitch, 249
 - GlobalVariable, 248, 249
- arcs::aubo_scope::GlobalVariableFilter, 250
 - operator(), 250
- arcs::aubo_scope::GripActionParameters, 250
 - ~GripActionParameters, 250
 - getForce, 250
 - getSpeed, 251
 - getVacuum, 251
 - getWidth, 251
 - isGripDetectionEnabled, 251
- arcs::aubo_scope::GripForceCapability, 252
 - ~GripForceCapability, 252
 - updateCapability, 253
- arcs::aubo_scope::GripperApiProvider, 253
 - ~GripperApiProvider, 253
 - getIoModel, 254
 - getSystemApi, 254
- arcs::aubo_scope::GripperConfiguration, 254
 - ~GripperConfiguration, 254
 - registerGripDetectedCapability, 255
 - registerGrippingForceCapability, 255
 - registerGrippingVacuumCapability, 256
 - registerReleaseDetectedCapability, 257
 - registerSpeedCapability, 258
 - registerWidthCapability, 259
- arcs::aubo_scope::GripperContribution, 260
 - ~GripperContribution, 260
 - configureGripper, 261
 - generateGripActionScript, 261
 - generateReleaseActionScript, 261
 - getTitle, 262
- arcs::aubo_scope::GripperDevice, 262
 - ~GripperDevice, 264
- arcs::aubo_scope::GripperNode, 264
 - ~GripperNode, 267
 - ConfigType, 265
 - CUSTOM_PARAMETERS, 266
 - d_, 268
 - DataSwitch, 268
 - getConfigType, 267
 - getGripperDevice, 267
 - getSettingType, 267
 - GRIP_ACTION, 265
 - GripperNode, 266, 267
 - PAYLOAD_SELECTION, 266
 - RELEASE_ACTION, 266
 - setConfigType, 267
 - SettingType, 266
 - UNDEFINED, 265
 - UNDEFINED0, 266
- arcs::aubo_scope::GripVacuumCapability, 268
 - ~GripVacuumCapability, 268
 - updateCapability, 268
- arcs::aubo_scope::HaltNode, 269
 - ~HaltNode, 271
 - d_, 271
 - DataSwitch, 271
 - HaltNode, 270, 271
- arcs::aubo_scope::IfNode, 272
 - ~IfNode, 274
 - d_, 275
 - DataSwitch, 274
 - getExpression, 274
 - IfNode, 273, 274
 - setExpression, 274
- arcs::aubo_scope::InputIoFilter, 275
 - operator(), 275
- arcs::aubo_scope::InputValidator, 275
 - getMessage, 276
 - isValid, 276
- arcs::aubo_scope::InstallationApi, 277
 - ~InstallationApi, 279

- d_, 280
- DataSwitch, 279
- InstallationApi, 278, 279
- arcs::aubo_scope::InstallationApiProvider, 280
 - ~InstallationApiProvider, 281
 - d_, 282
 - DataSwitch, 282
 - getInstallationApi, 281
 - getSystemApi, 282
 - getUserInterfaceApi, 282
 - InstallationApiProvider, 280, 281
- arcs::aubo_scope::InstallationContributionConfiguration, 282
 - ~InstallationContributionConfiguration, 282
- arcs::aubo_scope::InstallationCreationContext, 283
 - ~InstallationCreationContext, 284
 - d_, 285
 - DataSwitch, 285
 - getCreatedVersion, 284
 - getLastSavedVersion, 284
 - getNodeCreationType, 285
 - InstallationCreationContext, 283, 284
 - LOAD, 283
 - NEW, 283
 - NodeCreationType, 283
- arcs::aubo_scope::InstallationNodeContribution, 285
 - ~InstallationNodeContribution, 285
 - closeView, 285
 - generateScript, 285
 - openView, 286
- arcs::aubo_scope::InstallationNodeService, 286
 - ~InstallationNodeService, 286
 - configureContribution, 286
 - createInstallationNode, 287
 - createView, 287
 - getIcon, 287
 - getTitle, 288
- arcs::aubo_scope::InstallationNodeView, 288
 - ~InstallationNodeView, 288
 - buildUi, 288
- arcs::aubo_scope::IntegerRangeValidator, 289
 - getMessage, 290
 - IntegerRangeValidator, 290
 - isValid, 290
 - max_value_, 291
 - min_value_, 291
 - range_func_, 291
 - updateValue, 290
- arcs::aubo_scope::IntegerRegister, 291
 - ~IntegerRegister, 293
- arcs::aubo_scope::Io, 293
 - ~Io, 296
 - ANALOG, 295
 - BOOLEAN, 295
 - CONFIGURABLE, 294
 - d_, 298
 - DataSwitch, 298
 - DIGITAL, 295
 - DOUBLE, 295
 - ETHERNET_IP, 295
 - GENERAL_PURPOSE, 295
 - getAction, 296
 - getDefaultName, 296
 - getDisplayName, 296
 - getInterfaceType, 297
 - getType, 297
 - getValueStr, 297
 - hashCode, 297
 - INTEGER, 295
 - InterfaceType, 294
 - Io, 295, 296
 - IoType, 295
 - isInput, 297
 - isResolvable, 297
 - LINKIO, 295
 - MODBUS, 295
 - PROFINET, 295
 - STANDARD, 294
 - TOOL, 294
- arcs::aubo_scope::IoConfigure, 298
 - function_lock, 299
 - functions, 299
 - label, 299
 - maxs, 299
 - mins, 299
 - names, 299
 - num, 299
 - safety_functions, 299
 - safety_functions_fixed, 299
 - sequences, 299
 - types, 299
- arcs::aubo_scope::IoModel, 300
 - ~IoModel, 301
 - d_, 302
 - DataSwitch, 302
 - getAll, 301
 - getIo, 301
 - getIos, 302
 - IoModel, 300, 301
- arcs::aubo_scope::KeyboardManager, 302
 - ~KeyboardManager, 304
 - d_, 307
 - DataSwitch, 307
 - KeyboardManager, 304, 305
 - registerCodeEditorKeyboard, 305
 - registerDoubleKeyboard, 305
 - registerExpressionKeyboard, 305
 - registerIntegerKeyboard, 305
 - registerIPAddressKeyboard, 305
 - registerNumberKeyboard, 305
 - registerPasswordKeyboard, 306
 - registerPositiveDoubleKeyboard, 306
 - registerPositiveIntegerKeyboard, 306
 - registerStandardKeyboard, 306
 - registerStringKeyboard, 307
- arcs::aubo_scope::LoopNode, 307

- ~LoopNode, 310
- ALWAYS, 309
- ConfigType, 309
- COUNTER, 309
- d_, 312
- DataSwitch, 312
- EXPRESSION, 309
- getConfigType, 310
- getExpression, 311
- getLoopCount, 311
- getLoopCountVariable, 311
- isEvaluateContinously, 311
- LoopNode, 309, 310
- setConfigType, 311
- setEvaluateContinously, 311
- setExpression, 311
- setLoopCount, 311
- setLoopCountVariable, 312
- arcs::aubo_scope::ModbusIo, 312
 - ~ModbusIo, 315
 - d_, 317
 - DataSwitch, 316
 - getIpAddress, 315
 - getSignalAddress, 316
 - getValue, 316
 - isSlave, 316
 - ModbusIo, 314, 315
 - setValue, 316
- arcs::aubo_scope::ModbusIoFilter, 317
 - is_input_, 317
 - ModbusIoFilter, 317
 - operator(), 317
 - type_, 317
- arcs::aubo_scope::MoveNode, 318
 - ~MoveNode, 321
 - ConfigType, 319
 - d_, 324
 - DataSwitch, 324
 - FEATURE, 320
 - FeatureSelectionType, 320
 - getBlend, 321
 - getConfigType, 321
 - getFeature, 322
 - getJointAcceleration, 322
 - getJointSpeed, 322
 - getTCPSelection, 322
 - getTcpSelectionType, 322
 - getToolAcceleration, 322
 - getToolSpeed, 322
 - getUseJoints, 322
 - IGNORE_ACTIVE_TCP, 320
 - MOVE_J, 319
 - MOVE_L, 319
 - MOVE_S, 320
 - MOVE_TCP, 320
 - MoveNode, 320, 321
 - setBlend, 322
 - setConfigType, 322
 - setFeature, 322, 323
 - setJointAcceleration, 323
 - setJointSpeed, 323
 - setTCPSelection, 323
 - setTcpSelectionType, 323
 - setToolAcceleration, 324
 - setToolSpeed, 324
 - setUseJoints, 324
 - TcpSelectionType, 320
 - UNRESOLVED_FEATURE, 320
 - USE_ACTIVE_TCP, 320
- arcs::aubo_scope::OfflineTrackNode, 325
 - ~OfflineTrackNode, 327
 - d_, 327
 - DataSwitch, 327
 - OfflineTrackNode, 326, 327
- arcs::aubo_scope::OutputIoFilter, 327
 - operator(), 327
- arcs::aubo_scope::PalletNode, 328
 - ~PalletNode, 330
 - d_, 330
 - DataSwitch, 330
 - PalletNode, 329, 330
- arcs::aubo_scope::Payload, 330
 - ~Payload, 332
 - d_, 335
 - DataSwitch, 334
 - getCenterOfGravity, 333
 - getDisplayName, 333
 - getInertiaMatrix, 333
 - getMass, 333
 - getParametersWarning, 333
 - hashCode, 334
 - isResolvable, 334
 - isUsable, 334
 - MASS_OKG, 331
 - MASS_AND_CENTER_OF_GRAVITY_EXCEEDED, 331
 - NO_WARNING, 331
 - ParametersWarning, 331
 - Payload, 332
 - RATED_MASS_EXCEEDED, 331
- arcs::aubo_scope::PayloadContributionModel, 335
 - ~PayloadContributionModel, 336
 - addPayload, 337
 - d_, 340
 - DataSwitch, 340
 - getPayload, 338
 - PayloadContributionModel, 336
 - removePayload, 339
 - updatePayload, 339, 340
- arcs::aubo_scope::PayloadModel, 341
 - ~PayloadModel, 342
 - d_, 343
 - DataSwitch, 343
 - getCurrentPayload, 342
 - getDefaultPayload, 342
 - getPayload, 343

- getPayloads, 343
- PayloadModel, 341, 342
- arcs::aubo_scope::PersistedVariable, 343
 - ~PersistedVariable, 346
 - d_, 347
 - DataSwitch, 346
 - isResolvable, 346
 - PersistedVariable, 345, 346
- arcs::aubo_scope::PersistedVariableFilter, 347
 - operator(), 347
- arcs::aubo_scope::PopupManager, 347
 - ~PopupManager, 348
 - d_, 349
 - DefaultSingletonTraits< PopupManager >, 349
 - isPopupCountZero, 349
 - notifyPopupHidden, 349
 - notifyPopupShown, 349
 - popupCount, 349
 - PopupManager, 348, 349
 - setPopupCountChangedCallback, 349
- arcs::aubo_scope::PopupNode, 350
 - ~PopupNode, 352
 - d_, 353
 - DataSwitch, 353
 - ERROR, 351
 - getMessage, 352
 - getType, 352
 - MESSAGE, 351
 - PopupNode, 351, 352
 - setMessage, 352
 - setType, 353
 - Type, 351
 - WARNING, 351
- arcs::aubo_scope::PrepareToStartScrewdriverParameters, 353
 - ~PrepareToStartScrewdriverParameters, 354
- arcs::aubo_scope::Profinet, 354
 - ~Profinet, 357
 - d_, 358
 - DataSwitch, 358
 - getBitOffset, 357
 - getByteOffset, 357
 - getValue, 357
 - isSlave, 358
 - Profinet, 356, 357
 - setValue, 358
- arcs::aubo_scope::ProfinetFilter, 358
 - is_input_, 359
 - operator(), 359
 - ProfinetFilter, 359
 - type_, 359
- arcs::aubo_scope::ProgramApi, 359
 - ~ProgramApi, 362
 - d_, 362
 - DataSwitch, 362
 - ProgramApi, 361, 362
- arcs::aubo_scope::ProgramApiProvider, 363
 - ~ProgramApiProvider, 364
- d_, 365
- DataSwitch, 365
- getProgramApi, 364
- getSystemApi, 364
- getUiInterfaceApi, 364
- ProgramApiProvider, 363, 364
- arcs::aubo_scope::ProgramContributionConfiguration, 365
 - ~ProgramContributionConfiguration, 367
 - d_, 370
 - DataSwitch, 370
 - ProgramContributionConfiguration, 366, 367
 - setAllowBreakpointOnChildNodesInSubtree, 367
 - setAllowBreakpointOnNode, 367
 - setAllowInsertOnlyIntoNodes, 367
 - setAllowStartFromChildNodesInSubtree, 368
 - setAllowStartFromNode, 368
 - setChildrenAllowed, 368
 - setChildrenAllowedOnly, 369
 - setDeprecated, 369
 - setDisplayOrderId, 369
 - setUserInsertable, 369
- arcs::aubo_scope::ProgramCreationContext, 370
 - ~ProgramCreationContext, 372
 - d_, 372
 - DataSwitch, 372
 - getCreatedVersion, 372
 - getLastSavedVersion, 372
 - getNodeCreationType, 372
 - LOAD, 371
 - NEW, 371
 - NodeCreationType, 370
 - ProgramCreationContext, 371, 372
- arcs::aubo_scope::ProgramHomeNode, 372
 - ~ProgramHomeNode, 375
 - d_, 375
 - DataSwitch, 375
 - ProgramHomeNode, 374, 375
- arcs::aubo_scope::ProgramModel, 375
 - ~ProgramModel, 377
 - d_, 378
 - DataSwitch, 378
 - getProgramNodeFactory, 377
 - getTreeNode, 377
 - ProgramModel, 376, 377
- arcs::aubo_scope::ProgramNode, 378
 - ~ProgramNode, 381
 - d_, 381
 - DataSwitch, 381
 - getProgramNodeContribution, 381
 - ProgramNode, 380, 381
- arcs::aubo_scope::ProgramNodeContribution, 381
 - ~ProgramNodeContribution, 382
 - closeView, 382
 - generateScript, 382
 - getAction, 382
 - getTitle, 383
 - isDefined, 383

- openView, 383
- postAction, 383
- arcs::aubo_scope::ProgramNodeFactory, 384
 - ~ProgramNodeFactory, 386
 - createAssemblyNode, 386
 - createAssignmentNode, 386
 - createBreakNode, 386
 - createCallNode, 386
 - createCaseNode, 386
 - createCircleMoveNode, 386
 - createCommentNode, 387
 - createConveyorTrackingNode, 387
 - createDefaultCaseNode, 387
 - createDirectionNode, 387
 - createElseIfNode, 387
 - createElseNode, 387
 - createFolderNode, 387
 - createForceNode, 387
 - createForceOptionNode, 387
 - createGripperNode, 387
 - createHaltNode, 388
 - createIfNode, 388
 - createLoopNode, 388
 - createMoveNode, 388
 - createMoveNodeNoTemplate, 388
 - createOfflineTrackNode, 388
 - createPalletNode, 388
 - createPopupNode, 388
 - createProgramHomeNode, 388
 - createRecordTrackNode, 388
 - createScrewdrivingNode, 388
 - createScriptNode, 388
 - createSeekNode, 389
 - createSetNode, 389
 - createSetPayloadNode, 389
 - createSubProgNode, 389
 - createSwitchNode, 389
 - createThreadNode, 389
 - createTimerNode, 389
 - createUntilNode, 389
 - createUserProgramNode, 389
 - createVisualLocalizationProgramNode, 390
 - createWaitNode, 390
 - createWaypointNode, 390
 - d_, 391
 - DataSwitch, 390
 - ProgramNodeFactory, 385, 386
- arcs::aubo_scope::ProgramNodeService, 391
 - ~ProgramNodeService, 391
 - configureContribution, 391
 - createNode, 391
 - createView, 392
 - getIcon, 392
 - getId, 392
 - getTitle, 392
- arcs::aubo_scope::ProgramNodeView, 393
 - ~ProgramNodeView, 393
 - buildUi, 393
- arcs::aubo_scope::ProgramSelectionParameters, 393
 - ~ProgramSelectionParameters, 394
 - getScrewdriverOperationType, 394
 - getScrewdriverProgram, 394
- arcs::aubo_scope::RecordTrackNode, 395
 - ~RecordTrackNode, 397
 - d_, 397
 - DataSwitch, 397
 - RecordTrackNode, 396, 397
- arcs::aubo_scope::Register, 397
 - ~Register, 399
- arcs::aubo_scope::RegisteredCapabilities, 399
 - ~RegisteredCapabilities, 399
 - getGripForceCapability, 399
 - getGripVacuumCapability, 400
 - getSpeedCapability, 400
 - getWidthCapability, 400
- arcs::aubo_scope::ReleaseActionParameters, 401
 - ~ReleaseActionParameters, 401
 - getSpeed, 401
 - getWidth, 401
 - isReleaseDetectionEnabled, 402
- arcs::aubo_scope::RenderInterface, 402
 - ~RenderInterface, 406
 - addCoordinate, 406
 - addCube, 406–408
 - addLine, 408, 409
 - addMesh, 409
 - addPlane, 409, 410
 - addPointCloud, 410
 - addWaypoint, 410
 - All, 405
 - clearCubes, 411
 - clearLines, 411
 - clearMeshes, 411
 - clearPlanes, 411
 - clearPointCloud, 411
 - clearWaypoints, 411
 - Coordinate, 405
 - d_, 416
 - DataSwitch, 416
 - getMeshes, 412
 - Ground, 405
 - hasCoordinate, 412
 - hasPointCloud, 412
 - hasWaypoint, 413
 - None, 405
 - Obstacle, 405
 - Point, 405
 - PointCloud, 405
 - RealRobot, 405
 - removeCoordinate, 413
 - removeCube, 413
 - removeLine, 413
 - removeMesh, 414
 - removePlane, 414
 - removePointCloud, 414
 - removeWaypoint, 414

- RenderInterface, 405, 406
- SafetyPlane, 405
- setBlendTrack, 414, 415
- setMoveTrack, 415
- setRenderEmbedded, 416
- setShowOptions, 416
- ShowOptions, 404
- TargetRobot, 405
- ToolPosition, 405
- Track, 405
- View, 405
- arcs::aubo_scope::ScrewdriverApiProvider, 416
 - ~ScrewdriverApiProvider, 417
 - getIOModel, 417
 - getSystemAPI, 417
- arcs::aubo_scope::ScrewdriverConfiguration, 417
 - ~ScrewdriverConfiguration, 418
 - addScrewdriverProgram, 418
 - addScrewdriverPrograms, 418
 - registerDriveScrewNotOKCapability, 419
 - registerDriveScrewOKCapability, 420
 - registerFeedScrewCapability, 420
 - registerOperationTypeCapability, 421
 - registerPrepareToStartScrewdriverCapability, 421
 - registerProgramSelectionCapability, 422
 - registerScrewdriverReadyCapability, 423
- arcs::aubo_scope::ScrewdriverContribution, 423
 - ~ScrewdriverContribution, 424
 - configureScrewdriver, 424
 - generateStartScrewdriverScript, 424
 - generateStopScrewdriverScript, 425
 - getTitle, 425
- arcs::aubo_scope::ScrewdriverParameters, 425
 - ~ScrewdriverParameters, 426
 - getScrewdriverOperationType, 426
 - getScrewdriverProgram, 426
- arcs::aubo_scope::ScrewdriverProgram, 427
 - ~ScrewdriverProgram, 427
 - getDisplayName, 427
 - getId, 427
- arcs::aubo_scope::ScrewdriverReadyParameters, 427
 - ~ScrewdriverReadyParameters, 428
- arcs::aubo_scope::ScrewdrivingNode, 428
 - ~ScrewdrivingNode, 430
 - d_, 430
 - DataSwitch, 430
 - ScrewdrivingNode, 429, 430
- arcs::aubo_scope::ScriptNode, 431
 - ~ScriptNode, 433
 - d_, 433
 - DataSwitch, 433
 - ScriptNode, 432, 433
- arcs::aubo_scope::ScriptWriter, 433
 - ~ScriptWriter, 436
 - anonyFunction, 436
 - appendLine, 436
 - appendRaw, 436
 - appendVectorDouble, 436
 - assign, 436–438
 - d_, 442
 - DataSwitch, 441
 - decreaseIndent, 438
 - defineFunction, 438
 - defineThread, 438
 - elseCondition, 438
 - elseifCondition, 438
 - end, 438
 - forCondition, 439
 - generateScript, 439
 - getResolvedVariable, 439
 - ifCondition, 439
 - ifNotCondition, 439
 - increaseIndent, 439
 - incrementVariable, 439
 - killThread, 440
 - lineFeed, 440
 - note, 440
 - returnMethod, 440
 - runThread, 440
 - ScriptWriter, 435, 436
 - setLabel, 440
 - setPayload, 440
 - setRobotIndex, 440
 - setTcp, 440
 - sleep, 440
 - sync, 441
 - whileCondition, 441
 - whileNot, 441
 - whileTrue, 441
 - writeChildren, 441
- arcs::aubo_scope::SeekNode, 442
 - ~SeekNode, 444
 - d_, 444
 - DataSwitch, 444
 - SeekNode, 443, 444
- arcs::aubo_scope::SetNode, 444
 - ~SetNode, 448
 - ANALOG_OUTPUT_CURRENT, 447
 - ANALOG_OUTPUT_VOLTAGE, 447
 - ConfigType, 446
 - d_, 451
 - DataSwitch, 451
 - DIGITAL_OUTPUT, 447
 - EXPRESSION_OUTPUT, 447
 - FLOAT_REGISTER_OUTPUT, 447
 - getAnalogOutput, 448
 - getBoolValueToSet, 449
 - getConfigType, 449
 - getCurrent, 449
 - getExpression, 449
 - getFloatValueToSet, 449
 - getOutput, 449
 - getOutputSelectionType, 449
 - getPulseTime, 449
 - getVoltage, 450
 - INCREMENT_VARIABLE, 447

- NO_ACTION, 447
- NO_OUTPUT, 447
- NO_TYPE, 446
- OUTPUT, 447
- OutputSelectionType, 447
- setAnalogOutput, 450
- setBoolValueToSet, 450
- setConfigType, 450
- setCurrent, 450
- setExpression, 450
- setFloatValueToSet, 450
- SetNode, 447, 448
- setOutput, 450
- setPulseTime, 450
- setVoltage, 450
- SINGLE_DIGITAL_PULSE, 447
- UNRESOLVED_OUTPUT, 447
- arcs::aubo_scope::SetPayloadNode, 451
 - ~SetPayloadNode, 454
 - ConfigType, 452
 - CUSTOM_PARAMETERS, 453
 - d_, 455
 - DataSwitch, 454
 - getCenterOfGravity, 454
 - getConfigType, 454
 - getMass, 454
 - PAYLOAD_SELECTION, 452
 - setCenterOfGravity, 454
 - setConfigType, 454
 - setMass, 454
 - SetPayloadNode, 453, 454
- arcs::aubo_scope::SpeedCapability, 455
 - ~SpeedCapability, 455
 - updateCapability, 455
- arcs::aubo_scope::StringLengthValidator, 456
 - getMessage, 457
 - isValid, 457
 - max_length_, 458
 - min_length_, 458
 - range_func_, 458
 - StringLengthValidator, 457
 - updateValue, 457
- arcs::aubo_scope::SubProgNode, 458
 - ~SubProgNode, 460
 - d_, 460
 - DataSwitch, 460
 - SubProgNode, 459, 460
- arcs::aubo_scope::SwitchNode, 461
 - ~SwitchNode, 463
 - d_, 463
 - DataSwitch, 463
 - SwitchNode, 462, 463
- arcs::aubo_scope::SystemApi, 463
 - ~SystemApi, 466
 - d_, 466
 - DataSwitch, 466
 - SystemApi, 465, 466
- arcs::aubo_scope::TCP, 467
 - ~TCP, 468
 - d_, 469
 - DataSwitch, 469
 - getDisplayName, 468
 - getOffset, 468
 - hashCode, 468
 - isResolvable, 469
 - TCP, 467, 468
- arcs::aubo_scope::TcpContributionModel, 469
 - ~TcpContributionModel, 471
 - addTcp, 471
 - d_, 473
 - DataSwitch, 473
 - getTcp, 471
 - removeTcp, 472
 - TcpContributionModel, 470, 471
 - updateTcp, 472
- arcs::aubo_scope::TcpModel, 473
 - ~TcpModel, 475
 - connectedTo, 475
 - d_, 477
 - DataSwitch, 476
 - disconnectFrom, 475
 - getAll, 476
 - getDefaultTcp, 476
 - getTcp, 476
 - TcpModel, 474, 475
- arcs::aubo_scope::ThreadNode, 477
 - ~ThreadNode, 479
 - d_, 479
 - DataSwitch, 479
 - ThreadNode, 478, 479
- arcs::aubo_scope::TimerNode, 479
 - ~TimerNode, 482
 - d_, 482
 - DataSwitch, 482
 - TimerNode, 481, 482
- arcs::aubo_scope::TreeNode, 482
 - ~TreeNode, 484
 - addChild, 484
 - cutChildNode, 485
 - d_, 488
 - DataSwitch, 488
 - getChildren, 485
 - getChildrenNotSuppressed, 485
 - getParent, 485
 - getProgramNode, 485
 - insertChildAfter, 485
 - insertChildBefore, 486
 - isInProgramTree, 486
 - isSuppressed, 487
 - locateDescendantTreeNode, 487
 - removeChild, 487
 - setChildSequenceLocked, 487
 - traverse, 488
 - TreeNode, 483, 484
- arcs::aubo_scope::UIntegerRangeValidator, 488
 - getMessage, 490

- isValid, 490
- max_value_, 490
- min_value_, 490
- range_func_, 491
- UIntegerRangeValidator, 490
- updateValue, 490
- arcs::aubo_scope::UndoRedoManager, 491
 - ~UndoRedoManager, 492
 - d_, 493
 - DataSwitch, 493
 - recordChanges, 492
 - UndoRedoManager, 491, 492
- arcs::aubo_scope::UntilNode, 493
 - ~UntilNode, 498
 - ANALOG_INPUT, 497
 - Angular, 497
 - BlendParameterType, 496
 - Cartesian, 497
 - CompareOperator, 496
 - ConfigType, 496
 - CUSTOM_BLEND, 496
 - CUSTOM_DECELERATION, 497
 - CustomDecelerationParameterType, 497
 - d_, 503
 - DataSwitch, 503
 - DecelerationParameterType, 497
 - DIGITAL_INPUT, 497
 - DISTANCE, 496
 - EXPRESSION, 496
 - FLOAT_REGISTER_INPUT, 497
 - getAnalogInputValueToProceed, 499
 - getAngularDeceleration, 499
 - getBlendParameterType, 499
 - getCartesianDeceleration, 499
 - getCompareOperator, 499
 - getConfigType, 499
 - getCustomDecelerationParameterType, 499
 - getDecelerationParameters, 499
 - getDecelerationParameterType, 499
 - getDigitalInputValueToProceed, 499
 - getDistance, 500
 - getExpressionConfig, 500
 - getIOInputSelectionType, 500
 - getWaypointBlendParameter, 500
 - GREATER_THAN, 496
 - IO_INPUT, 496
 - IOInputSelectionType, 497
 - LESS_THAN, 496
 - NO_BLEND, 496
 - NONE_CONFIG_TYPE, 496
 - NONE_INPUT_SELECT, 497
 - REACHED_WAYPOINT, 496
 - setAnalogInputValueToProceed, 500
 - setAngularDeceleration, 500
 - setBlendParameterType, 501
 - setCartesianDeceleration, 501
 - setCompareOperator, 501
 - setConfigType, 501
 - setCustomDecelerationParameterType, 501
 - setDecelerationParameters, 501
 - setDecelerationParameterType, 501
 - setDigitalInputValueToProceed, 501
 - setDistance, 502
 - setExpressionConfig, 502
 - setIOInputSelectionType, 502
 - setWaypointBlendParameter, 502
 - SHARED_DECELERATION, 497
 - TOOL_CONTACT, 496
 - Unknown, 497
 - UntilNode, 498
- arcs::aubo_scope::UserInteraction, 503
 - ~UserInteraction, 505
 - d_, 510
 - DataSwitch, 509
 - getKeyboardManager, 505
 - getPopupManager, 505
 - getRenderInterface, 505
 - getRobotThumb, 506
 - getUserDefinedRobotPosition, 506, 507
 - requestUserToMoveJoint, 507, 508
 - requestUserToMoveLine, 508, 509
 - setOperationLogDisplayName, 509
 - UserInteraction, 504, 505
- arcs::aubo_scope::UserInterfaceApi, 510
 - ~UserInterfaceApi, 511
 - d_, 512
 - DataSwitch, 511
 - UserInterfaceApi, 510, 511
- arcs::aubo_scope::ValueFactory, 512
 - ~ValueFactory, 513
 - createExpression, 513, 514
 - d_, 514
 - DataSwitch, 514
 - ValueFactory, 512, 513
- arcs::aubo_scope::Variable, 514
 - ~Variable, 516
 - d_, 518
 - DataSwitch, 517
 - equals, 517
 - FEATURE, 515
 - getDisplayName, 517
 - getType, 517
 - GLOBAL, 515
 - hashCode, 517
 - toString, 517
 - Type, 515
 - VALUE_PERSISTED, 515
 - Variable, 516
- arcs::aubo_scope::VariableFactory, 518
 - ~VariableFactory, 519
 - createFeatureVariable, 520
 - createGlobalVariable, 520
 - d_, 521
 - DataSwitch, 521
 - VariableFactory, 518, 519
- arcs::aubo_scope::VariableModel, 521

- ~VariableModel, 523
- connectedTo, 523
- d_, 525
- DataSwitch, 525
- disconnectFrom, 524
- get, 524
- getAll, 524
- getFeatureVariable, 525
- getVariableFactory, 525
- VariableModel, 522, 523
- arcs::aubo_scope::ViewApiProvider, 525
 - ~ViewApiProvider, 527
 - d_, 528
 - DataSwitch, 527
 - getSystemApi, 527
 - getUserInterfaceApi, 527
 - ViewApiProvider, 526, 527
- arcs::aubo_scope::VisualLocalizationProgramNode, 528
 - ~VisualLocalizationProgramNode, 530
 - d_, 530
 - DataSwitch, 530
 - VisualLocalizationProgramNode, 529, 530
- arcs::aubo_scope::WaitNode, 531
 - ~WaitNode, 534
 - ANALOG_INPUT_CURRENT, 533
 - ANALOG_INPUT_VOLTAGE, 533
 - CompareOperator, 533
 - ConfigType, 533
 - d_, 539
 - DataSwitch, 538
 - DIGITAL_INPUT, 533
 - EXPRESSION_INPUT, 533
 - FLOAT_REGISTER_INPUT, 533
 - getAnalogInput, 534
 - getBoolRegister, 534
 - getBoolValueToWaitFor, 535
 - getCompareOperator, 535
 - getConfigType, 535
 - getDigitalInput, 535
 - getDoubleRegisterInput, 535
 - getExpressionInputConfig, 535
 - getFloatValueToWaitFor, 535
 - getModbusInput, 535
 - getWaitTime, 535
 - GREATER_THAN, 533
 - LESS_THAN, 533
 - NO_TYPE, 533
 - NO_WAIT, 533
 - setAnalogInput, 535
 - setBoolRegister, 536
 - setBoolValueToWaitFor, 536
 - setCompareOperator, 536
 - setConfigType, 536
 - setDigitalInput, 537
 - setDoubleRegisterInput, 537
 - setExpressionInputConfig, 537
 - setFloatValueToWaitFor, 538
 - setModbusInput, 538
 - setWaitTime, 538
 - TIME, 533
 - WaitNode, 533, 534
- arcs::aubo_scope::Waypoint, 539
 - ~Waypoint, 540
 - d_, 542
 - DataSwitch, 542
 - getDisplayName, 541
 - getJointPositions, 541
 - getPose, 541
 - getTcpOffset, 541
 - hashCode, 541
 - isDefined, 541
 - Waypoint, 540
- arcs::aubo_scope::WaypointModel, 542
 - ~WaypointModel, 543
 - addWaypoint, 544
 - d_, 546
 - DataSwitch, 546
 - getByName, 545
 - getWaypoints, 545
 - removeWaypoint, 545
 - renameWaypoint, 545
 - WaypointModel, 543
- arcs::aubo_scope::WaypointNode, 546
 - ~WaypointNode, 551
 - BlendType, 548
 - ConfigType, 548
 - CUSTOM, 550
 - d_, 556
 - DataSwitch, 556
 - DEFINED, 549, 550
 - FIXED_POSITION, 549
 - getBlend, 551
 - getBlendType, 552
 - getConfigType, 552
 - getJointAcceleration, 552
 - getJointPositions, 552
 - getJointSpeed, 552
 - getMotionType, 552
 - getPose, 552
 - getPositionDefinition, 552
 - getTcpOffset, 553
 - getTime, 553
 - getToolAcceleration, 553
 - getToolSpeed, 553
 - getVariable, 553
 - getVariableDefinition, 553
 - INPUT, 550
 - JOINT_MOTION, 549
 - MotionType, 549
 - NO_BLEND, 548
 - PositionDefinition, 549
 - RELATIVE_POSITION, 549
 - RelativeOffsetType, 549
 - RelativePositionType, 550
 - setFixedPositionConfig, 553, 554

- setJointMotionParameters, 554
- setSharedMotionParameters, 554
- setTimeMotionParameters, 555
- setToolMotionParameters, 555
- setVariablePositionConfig, 555, 556
- SHARED_BLEND, 548
- SHARED_MOTION, 549
- TEACHING, 550
- TIME_MOTION, 549
- TOOL_MOTION, 549
- UNDEFINED, 549, 550
- VARIABLE, 550
- VARIABLE_JOINT, 550
- VARIABLE_POSE, 550
- VARIABLE_POSITION, 549
- VariableDefinition, 550
- VariableType, 550
- WAYPOINT_BLEND, 548
- WaypointNode, 550, 551
- arcs::aubo_scope::WidthCapability, 556
 - ~WidthCapability, 557
 - updateCapability, 557
- arcs::aubo_sdk, 116
 - ARCS_CLASS_FORWARD, 116
- ARCS_ABI_EXPORT
 - class_forward.h, 560
- ARCS_ABI_IMPORT
 - class_forward.h, 560
- ARCS_ABI_LOCAL
 - class_forward.h, 560
- ARCS_CLASS_FORWARD
 - arcs::aubo_scope, 105–116
 - arcs::aubo_sdk, 116
 - class_forward.h, 560
- ARCS_DECLARE_PTR
 - class_forward.h, 560
- ARCS_DECLARE_PTR_MEMBER
 - class_forward.h, 560
- ARCS_STRUCT_FORWARD
 - class_forward.h, 560
- ASK
 - arcs::aubo_scope::ConveyorTrackingNode, 155
- AssemblyNode
 - arcs::aubo_scope::AssemblyNode, 127, 128
- assign
 - arcs::aubo_scope::ScriptWriter, 436–438
- AssignmentNode
 - arcs::aubo_scope::AssignmentNode, 131, 132
- AuboStudio SDK, 23
- AXIS
 - arcs::aubo_scope::DirectionNode, 180
- Axis
 - arcs::aubo_scope::DirectionNode, 180
- BaseFeature
 - arcs::aubo_scope::BaseFeature, 135, 136
- BlendParameterType
 - arcs::aubo_scope::UntilNode, 496
- BlendType
 - arcs::aubo_scope::WaypointNode, 548
- BOOLEAN
 - arcs::aubo_scope::Io, 295
- BreakNode
 - arcs::aubo_scope::BreakNode, 140, 141
- buildUi
 - arcs::aubo_scope::InstallationNodeView, 288
 - arcs::aubo_scope::ProgramNodeView, 393
- CallNode
 - arcs::aubo_scope::CallNode, 143
- Cartesian
 - arcs::aubo_scope::UntilNode, 497
- CaseNode
 - arcs::aubo_scope::CaseNode, 145, 146
- CircleMoveNode
 - arcs::aubo_scope::CircleMoveNode, 148, 149
- class_forward.h
 - ARCS_ABI_EXPORT, 560
 - ARCS_ABI_IMPORT, 560
 - ARCS_ABI_LOCAL, 560
 - ARCS_CLASS_FORWARD, 560
 - ARCS_DECLARE_PTR, 560
 - ARCS_DECLARE_PTR_MEMBER, 560
 - ARCS_STRUCT_FORWARD, 560
- CLEAR_QUEUE
 - arcs::aubo_scope::ConveyorTrackingNode, 155
- clearCubes
 - arcs::aubo_scope::RenderInterface, 411
- clearLines
 - arcs::aubo_scope::RenderInterface, 411
- clearMeshes
 - arcs::aubo_scope::RenderInterface, 411
- clearPlanes
 - arcs::aubo_scope::RenderInterface, 411
- clearPointCloud
 - arcs::aubo_scope::RenderInterface, 411
- clearWaypoints
 - arcs::aubo_scope::RenderInterface, 411
- closeView
 - arcs::aubo_scope::InstallationNodeContribution, 285
 - arcs::aubo_scope::ProgramNodeContribution, 382
- CommentNode
 - arcs::aubo_scope::CommentNode, 151
- CompareOperator
 - arcs::aubo_scope::UntilNode, 496
 - arcs::aubo_scope::WaitNode, 533
- COMPLIANT_INSERTION
 - arcs::aubo_scope::AssemblyNode, 127
- ConfigType
 - arcs::aubo_scope::AssemblyNode, 127
 - arcs::aubo_scope::AssignmentNode, 130
 - arcs::aubo_scope::ForceNode, 236
 - arcs::aubo_scope::GripperNode, 265
 - arcs::aubo_scope::LoopNode, 309
 - arcs::aubo_scope::MoveNode, 319
 - arcs::aubo_scope::SetNode, 446
 - arcs::aubo_scope::SetPayloadNode, 452

- arcs::aubo_scope::UntilNode, [496](#)
- arcs::aubo_scope::WaitNode, [533](#)
- arcs::aubo_scope::WaypointNode, [548](#)
- CONFIGURABLE
 - arcs::aubo_scope::Io, [294](#)
- configureContribution
 - arcs::aubo_scope::InstallationNodeService, [286](#)
 - arcs::aubo_scope::ProgramNodeService, [391](#)
- configureGripper
 - arcs::aubo_scope::GripperContribution, [261](#)
- configureScrewdriver
 - arcs::aubo_scope::ScrewdriverContribution, [424](#)
- connectedTo
 - arcs::aubo_scope::TcpModel, [475](#)
 - arcs::aubo_scope::VariableModel, [523](#)
- CONTINUE_TRACKING
 - arcs::aubo_scope::ConveyorTrackingNode, [155](#)
- ContributionProviderPtr
 - arcs::aubo_scope, [104](#)
- ConveyorStartActionQueueType
 - arcs::aubo_scope::ConveyorTrackingNode, [155](#)
- ConveyorStartActionType
 - arcs::aubo_scope::ConveyorTrackingNode, [155](#)
- ConveyorTrackingNode
 - arcs::aubo_scope::ConveyorTrackingNode, [155](#), [156](#)
- ConveyorTrackingTargetType
 - arcs::aubo_scope::ConveyorTrackingNode, [155](#)
- Coordinate
 - arcs::aubo_scope::RenderInterface, [405](#)
- COUNTER
 - arcs::aubo_scope::LoopNode, [309](#)
- createAssemblyNode
 - arcs::aubo_scope::ProgramNodeFactory, [386](#)
- createAssignmentNode
 - arcs::aubo_scope::ProgramNodeFactory, [386](#)
- createBreakNode
 - arcs::aubo_scope::ProgramNodeFactory, [386](#)
- createCallNode
 - arcs::aubo_scope::ProgramNodeFactory, [386](#)
- createCaseNode
 - arcs::aubo_scope::ProgramNodeFactory, [386](#)
- createCircleMoveNode
 - arcs::aubo_scope::ProgramNodeFactory, [386](#)
- createCommentNode
 - arcs::aubo_scope::ProgramNodeFactory, [387](#)
- createConveyorTrackingNode
 - arcs::aubo_scope::ProgramNodeFactory, [387](#)
- createDefaultCaseNode
 - arcs::aubo_scope::ProgramNodeFactory, [387](#)
- createDirectionNode
 - arcs::aubo_scope::ProgramNodeFactory, [387](#)
- createElseIfNode
 - arcs::aubo_scope::ProgramNodeFactory, [387](#)
- createElseNode
 - arcs::aubo_scope::ProgramNodeFactory, [387](#)
- createExpression
 - arcs::aubo_scope::ValueFactory, [513](#), [514](#)
- createFeatureVariable
 - arcs::aubo_scope::VariableFactory, [520](#)
- createFolderNode
 - arcs::aubo_scope::ProgramNodeFactory, [387](#)
- createForceNode
 - arcs::aubo_scope::ProgramNodeFactory, [387](#)
- createForceOptionNode
 - arcs::aubo_scope::ProgramNodeFactory, [387](#)
- createGlobalVariable
 - arcs::aubo_scope::VariableFactory, [520](#)
- createGripperNode
 - arcs::aubo_scope::ProgramNodeFactory, [387](#)
- createHaltNode
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createIfNode
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createInstallationNode
 - arcs::aubo_scope::InstallationNodeService, [287](#)
- createLoopNode
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createMoveNode
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createMoveNodeNoTemplate
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createNode
 - arcs::aubo_scope::ProgramNodeService, [391](#)
- createOfflineTrackNode
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createPalletNode
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createPopupNode
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createProgramHomeNode
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createRecordTrackNode
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createScrewdrivingNode
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createScriptNode
 - arcs::aubo_scope::ProgramNodeFactory, [388](#)
- createSeekNode
 - arcs::aubo_scope::ProgramNodeFactory, [389](#)
- createSetNode
 - arcs::aubo_scope::ProgramNodeFactory, [389](#)
- createSetPayloadNode
 - arcs::aubo_scope::ProgramNodeFactory, [389](#)
- createSubProgNode
 - arcs::aubo_scope::ProgramNodeFactory, [389](#)
- createSwitchNode
 - arcs::aubo_scope::ProgramNodeFactory, [389](#)
- createThreadNode
 - arcs::aubo_scope::ProgramNodeFactory, [389](#)
- createTimerNode
 - arcs::aubo_scope::ProgramNodeFactory, [389](#)
- createUntilNode
 - arcs::aubo_scope::ProgramNodeFactory, [389](#)
- createUserProgramNode
 - arcs::aubo_scope::ProgramNodeFactory, [389](#)

- createView
 - arcs::aubo_scope::InstallationNodeService, 287
 - arcs::aubo_scope::ProgramNodeService, 392
- createVisualLocalizationProgramNode
 - arcs::aubo_scope::ProgramNodeFactory, 390
- createWaitNode
 - arcs::aubo_scope::ProgramNodeFactory, 390
- createWaypointNode
 - arcs::aubo_scope::ProgramNodeFactory, 390
- CUSTOM
 - arcs::aubo_scope::WaypointNode, 550
- CUSTOM_BLEND
 - arcs::aubo_scope::UntilNode, 496
- CUSTOM_DECELERATION
 - arcs::aubo_scope::UntilNode, 497
- CUSTOM_PARAMETERS
 - arcs::aubo_scope::GripperNode, 266
 - arcs::aubo_scope::SetPayloadNode, 453
- CustomDecelerationParameterType
 - arcs::aubo_scope::UntilNode, 497
- cutChildNode
 - arcs::aubo_scope::TreeNode, 485
- d_
 - arcs::aubo_scope::AnalogIo, 121
 - arcs::aubo_scope::ApplicationApi, 125
 - arcs::aubo_scope::AssemblyNode, 129
 - arcs::aubo_scope::AssignmentNode, 133
 - arcs::aubo_scope::BaseFeature, 137
 - arcs::aubo_scope::BreakNode, 141
 - arcs::aubo_scope::CallNode, 144
 - arcs::aubo_scope::CaseNode, 146
 - arcs::aubo_scope::CircleMoveNode, 149
 - arcs::aubo_scope::CommentNode, 152
 - arcs::aubo_scope::ConveyorTrackingNode, 157
 - arcs::aubo_scope::DataModel, 167
 - arcs::aubo_scope::DefaultCaseNode, 170
 - arcs::aubo_scope::DigitalIo, 177
 - arcs::aubo_scope::DirectionNode, 184
 - arcs::aubo_scope::ElseIfNode, 193
 - arcs::aubo_scope::ElseNode, 196
 - arcs::aubo_scope::EtherNetIP, 200
 - arcs::aubo_scope::Expression, 204
 - arcs::aubo_scope::Feature, 207
 - arcs::aubo_scope::FeatureContributionModel, 211
 - arcs::aubo_scope::FeatureLine, 215
 - arcs::aubo_scope::FeatureModel, 218
 - arcs::aubo_scope::FeaturePlane, 221
 - arcs::aubo_scope::FeaturePoint, 224
 - arcs::aubo_scope::FeatureTool, 227
 - arcs::aubo_scope::FeatureVariable, 231
 - arcs::aubo_scope::FolderNode, 235
 - arcs::aubo_scope::ForceNode, 238
 - arcs::aubo_scope::ForceOptionNode, 240
 - arcs::aubo_scope::Function, 243
 - arcs::aubo_scope::FunctionModel, 246
 - arcs::aubo_scope::GlobalVariable, 249
 - arcs::aubo_scope::GripperNode, 268
 - arcs::aubo_scope::HaltNode, 271
 - arcs::aubo_scope::IfNode, 275
 - arcs::aubo_scope::InstallationApi, 280
 - arcs::aubo_scope::InstallationApiProvider, 282
 - arcs::aubo_scope::InstallationCreationContext, 285
 - arcs::aubo_scope::Io, 298
 - arcs::aubo_scope::IoModel, 302
 - arcs::aubo_scope::KeyboardManager, 307
 - arcs::aubo_scope::LoopNode, 312
 - arcs::aubo_scope::ModbusIo, 317
 - arcs::aubo_scope::MoveNode, 324
 - arcs::aubo_scope::OfflineTrackNode, 327
 - arcs::aubo_scope::PalletNode, 330
 - arcs::aubo_scope::Payload, 335
 - arcs::aubo_scope::PayloadContributionModel, 340
 - arcs::aubo_scope::PayloadModel, 343
 - arcs::aubo_scope::PersistedVariable, 347
 - arcs::aubo_scope::PopupManager, 349
 - arcs::aubo_scope::PopupNode, 353
 - arcs::aubo_scope::Profinet, 358
 - arcs::aubo_scope::ProgramApi, 362
 - arcs::aubo_scope::ProgramApiProvider, 365
 - arcs::aubo_scope::ProgramContributionConfiguration, 370
 - arcs::aubo_scope::ProgramCreationContext, 372
 - arcs::aubo_scope::ProgramHomeNode, 375
 - arcs::aubo_scope::ProgramModel, 378
 - arcs::aubo_scope::ProgramNode, 381
 - arcs::aubo_scope::ProgramNodeFactory, 391
 - arcs::aubo_scope::RecordTrackNode, 397
 - arcs::aubo_scope::RenderInterface, 416
 - arcs::aubo_scope::ScrewdrivingNode, 430
 - arcs::aubo_scope::ScriptNode, 433
 - arcs::aubo_scope::ScriptWriter, 442
 - arcs::aubo_scope::SeekNode, 444
 - arcs::aubo_scope::SetNode, 451
 - arcs::aubo_scope::SetPayloadNode, 455
 - arcs::aubo_scope::SubProgNode, 460
 - arcs::aubo_scope::SwitchNode, 463
 - arcs::aubo_scope::SystemApi, 466
 - arcs::aubo_scope::TCP, 469
 - arcs::aubo_scope::TcpContributionModel, 473
 - arcs::aubo_scope::TcpModel, 477
 - arcs::aubo_scope::ThreadNode, 479
 - arcs::aubo_scope::TimerNode, 482
 - arcs::aubo_scope::TreeNode, 488
 - arcs::aubo_scope::UndoRedoManager, 493
 - arcs::aubo_scope::UntilNode, 503
 - arcs::aubo_scope::UserInteraction, 510
 - arcs::aubo_scope::UserInterfaceApi, 512
 - arcs::aubo_scope::ValueFactory, 514
 - arcs::aubo_scope::Variable, 518
 - arcs::aubo_scope::VariableFactory, 521
 - arcs::aubo_scope::VariableModel, 525
 - arcs::aubo_scope::ViewApiProvider, 528
 - arcs::aubo_scope::VisualLocalizationProgramNode, 530
 - arcs::aubo_scope::WaitNode, 539

- arcs::aubo_scope::Waypoint, 542
- arcs::aubo_scope::WaypointModel, 546
- arcs::aubo_scope::WaypointNode, 556
- DataModel
 - arcs::aubo_scope::DataModel, 159, 160
- DataSwitch
 - arcs::aubo_scope::AnalogIo, 120
 - arcs::aubo_scope::ApplicationApi, 125
 - arcs::aubo_scope::AssemblyNode, 128
 - arcs::aubo_scope::AssignmentNode, 133
 - arcs::aubo_scope::BaseFeature, 136
 - arcs::aubo_scope::BreakNode, 141
 - arcs::aubo_scope::CallNode, 144
 - arcs::aubo_scope::CaseNode, 146
 - arcs::aubo_scope::CircleMoveNode, 149
 - arcs::aubo_scope::CommentNode, 152
 - arcs::aubo_scope::ConveyorTrackingNode, 156
 - arcs::aubo_scope::DataModel, 167
 - arcs::aubo_scope::DefaultCaseNode, 170
 - arcs::aubo_scope::DigitalIo, 177
 - arcs::aubo_scope::DirectionNode, 184
 - arcs::aubo_scope::ElseIfNode, 193
 - arcs::aubo_scope::ElseNode, 196
 - arcs::aubo_scope::EtherNetIP, 200
 - arcs::aubo_scope::Expression, 204
 - arcs::aubo_scope::Feature, 207
 - arcs::aubo_scope::FeatureContributionModel, 211
 - arcs::aubo_scope::FeatureLine, 215
 - arcs::aubo_scope::FeatureModel, 218
 - arcs::aubo_scope::FeaturePlane, 221
 - arcs::aubo_scope::FeaturePoint, 224
 - arcs::aubo_scope::FeatureTool, 227
 - arcs::aubo_scope::FeatureVariable, 231
 - arcs::aubo_scope::FolderNode, 235
 - arcs::aubo_scope::ForceNode, 238
 - arcs::aubo_scope::ForceOptionNode, 240
 - arcs::aubo_scope::Function, 243
 - arcs::aubo_scope::FunctionModel, 246
 - arcs::aubo_scope::GlobalVariable, 249
 - arcs::aubo_scope::GripperNode, 268
 - arcs::aubo_scope::HaltNode, 271
 - arcs::aubo_scope::IfNode, 274
 - arcs::aubo_scope::InstallationApi, 279
 - arcs::aubo_scope::InstallationApiProvider, 282
 - arcs::aubo_scope::InstallationCreationContext, 285
 - arcs::aubo_scope::Io, 298
 - arcs::aubo_scope::IoModel, 302
 - arcs::aubo_scope::KeyboardManager, 307
 - arcs::aubo_scope::LoopNode, 312
 - arcs::aubo_scope::ModbusIo, 316
 - arcs::aubo_scope::MoveNode, 324
 - arcs::aubo_scope::OfflineTrackNode, 327
 - arcs::aubo_scope::PalletNode, 330
 - arcs::aubo_scope::Payload, 334
 - arcs::aubo_scope::PayloadContributionModel, 340
 - arcs::aubo_scope::PayloadModel, 343
 - arcs::aubo_scope::PersistedVariable, 346
 - arcs::aubo_scope::PopupNode, 353
 - arcs::aubo_scope::Profinet, 358
 - arcs::aubo_scope::ProgramApi, 362
 - arcs::aubo_scope::ProgramApiProvider, 365
 - arcs::aubo_scope::ProgramContributionConfiguration, 370
 - arcs::aubo_scope::ProgramCreationContext, 372
 - arcs::aubo_scope::ProgramHomeNode, 375
 - arcs::aubo_scope::ProgramModel, 378
 - arcs::aubo_scope::ProgramNode, 381
 - arcs::aubo_scope::ProgramNodeFactory, 390
 - arcs::aubo_scope::RecordTrackNode, 397
 - arcs::aubo_scope::RenderInterface, 416
 - arcs::aubo_scope::ScrewdrivingNode, 430
 - arcs::aubo_scope::ScriptNode, 433
 - arcs::aubo_scope::ScriptWriter, 441
 - arcs::aubo_scope::SeekNode, 444
 - arcs::aubo_scope::SetNode, 451
 - arcs::aubo_scope::SetPayloadNode, 454
 - arcs::aubo_scope::SubProgNode, 460
 - arcs::aubo_scope::SwitchNode, 463
 - arcs::aubo_scope::SystemApi, 466
 - arcs::aubo_scope::TCP, 469
 - arcs::aubo_scope::TcpContributionModel, 473
 - arcs::aubo_scope::TcpModel, 476
 - arcs::aubo_scope::ThreadNode, 479
 - arcs::aubo_scope::TimerNode, 482
 - arcs::aubo_scope::TreeNode, 488
 - arcs::aubo_scope::UndoRedoManager, 493
 - arcs::aubo_scope::UntilNode, 503
 - arcs::aubo_scope::UserInteraction, 509
 - arcs::aubo_scope::UserInterfaceApi, 511
 - arcs::aubo_scope::ValueFactory, 514
 - arcs::aubo_scope::Variable, 517
 - arcs::aubo_scope::VariableFactory, 521
 - arcs::aubo_scope::VariableModel, 525
 - arcs::aubo_scope::ViewApiProvider, 527
 - arcs::aubo_scope::VisualLocalizationProgramNode, 530
 - arcs::aubo_scope::WaitNode, 538
 - arcs::aubo_scope::Waypoint, 542
 - arcs::aubo_scope::WaypointModel, 546
 - arcs::aubo_scope::WaypointNode, 556
 - DecelerationParameterType
 - arcs::aubo_scope::UntilNode, 497
 - decreaseIndent
 - arcs::aubo_scope::ScriptWriter, 438
 - DefaultCaseNode
 - arcs::aubo_scope::DefaultCaseNode, 169
 - DefaultSingletonTraits< PopupManager >
 - arcs::aubo_scope::PopupManager, 349
 - DEFINED
 - arcs::aubo_scope::WaypointNode, 549, 550
 - defineFunction
 - arcs::aubo_scope::ScriptWriter, 438
 - defineThread
 - arcs::aubo_scope::ScriptWriter, 438
 - DIGITAL

- arcs::aubo_scope::Io, 295
- DIGITAL_INPUT
 - arcs::aubo_scope::UntilNode, 497
 - arcs::aubo_scope::WaitNode, 533
- DIGITAL_OUTPUT
 - arcs::aubo_scope::SetNode, 447
- DigitalIo
 - arcs::aubo_scope::DigitalIo, 175, 176
- DigitalIoFilter
 - arcs::aubo_scope::DigitalIoFilter, 178
- DirectionNode
 - arcs::aubo_scope::DirectionNode, 181, 182
- DirectionSelectionType
 - arcs::aubo_scope::DirectionNode, 180
- disconnectFrom
 - arcs::aubo_scope::TcpModel, 475
 - arcs::aubo_scope::VariableModel, 524
- DISTANCE
 - arcs::aubo_scope::UntilNode, 496
- doc 目录参考, 83
- doc/SDK_overview.md, 559
- DOUBLE
 - arcs::aubo_scope::Io, 295
- DoubleRangeValidator
 - arcs::aubo_scope::DoubleRangeValidator, 185
- elseCondition
 - arcs::aubo_scope::ScriptWriter, 438
- elseifCondition
 - arcs::aubo_scope::ScriptWriter, 438
- ElseIfNode
 - arcs::aubo_scope::ElseIfNode, 191, 192
- ElseNode
 - arcs::aubo_scope::ElseNode, 195
- end
 - arcs::aubo_scope::ScriptWriter, 438
- equals
 - arcs::aubo_scope::Variable, 517
- ERROR
 - arcs::aubo_scope::PopupNode, 351
- ETHERNET_IP
 - arcs::aubo_scope::Io, 295
- EtherNetIP
 - arcs::aubo_scope::EtherNetIP, 198, 199
- EtherNetIPFilter
 - arcs::aubo_scope::EtherNetIPFilter, 201
- EXPRESSION
 - arcs::aubo_scope::AssignmentNode, 130
 - arcs::aubo_scope::DirectionNode, 180
 - arcs::aubo_scope::LoopNode, 309
 - arcs::aubo_scope::UntilNode, 496
- Expression
 - arcs::aubo_scope::Expression, 202, 203
- EXPRESSION_INPUT
 - arcs::aubo_scope::WaitNode, 533
- EXPRESSION_OUTPUT
 - arcs::aubo_scope::SetNode, 447
- FEATURE
 - arcs::aubo_scope::DirectionNode, 181
 - arcs::aubo_scope::MoveNode, 320
 - arcs::aubo_scope::Variable, 515
- Feature
 - arcs::aubo_scope::Feature, 205, 206
- FeatureContributionModel
 - arcs::aubo_scope::FeatureContributionModel, 208, 209
- FeatureLine
 - arcs::aubo_scope::FeatureLine, 214, 215
- FeatureModel
 - arcs::aubo_scope::FeatureModel, 216, 217
- FeaturePlane
 - arcs::aubo_scope::FeaturePlane, 220, 221
- FeaturePoint
 - arcs::aubo_scope::FeaturePoint, 223, 224
- FeatureSelectionType
 - arcs::aubo_scope::DirectionNode, 180
 - arcs::aubo_scope::MoveNode, 320
- FeatureTool
 - arcs::aubo_scope::FeatureTool, 226, 227
- FeatureVariable
 - arcs::aubo_scope::FeatureVariable, 230
- FIXED_POSITION
 - arcs::aubo_scope::WaypointNode, 549
- FLOAT_REGISTER_INPUT
 - arcs::aubo_scope::UntilNode, 497
 - arcs::aubo_scope::WaitNode, 533
- FLOAT_REGISTER_OUTPUT
 - arcs::aubo_scope::SetNode, 447
- FolderNode
 - arcs::aubo_scope::FolderNode, 233, 234
- ForceNode
 - arcs::aubo_scope::ForceNode, 237
- ForceOptionNode
 - arcs::aubo_scope::ForceOptionNode, 239, 240
- forCondition
 - arcs::aubo_scope::ScriptWriter, 439
- FRAME
 - arcs::aubo_scope::ForceNode, 236
- Function
 - arcs::aubo_scope::Function, 241, 242
- function_lock
 - arcs::aubo_scope::IoConfigure, 299
- FunctionModel
 - arcs::aubo_scope::FunctionModel, 244, 245
- functions
 - arcs::aubo_scope::IoConfigure, 299
- GENERAL_PURPOSE
 - arcs::aubo_scope::Io, 295
- generateGripActionScript
 - arcs::aubo_scope::GripperContribution, 261
- generateReleaseActionScript
 - arcs::aubo_scope::GripperContribution, 261
- generateScript
 - arcs::aubo_scope::InstallationNodeContribution, 285
 - arcs::aubo_scope::ProgramNodeContribution, 382

- arcs::aubo_scope::ScriptWriter, 439
- generateStartScrewdriverScript
 - arcs::aubo_scope::ScrewdriverContribution, 424
- generateStopScrewdriverScript
 - arcs::aubo_scope::ScrewdriverContribution, 425
- get
 - arcs::aubo_scope::ContributionProvider< T >, 153
 - arcs::aubo_scope::DataModel, 160–163
 - arcs::aubo_scope::VariableModel, 524
- getAction
 - arcs::aubo_scope::Io, 296
 - arcs::aubo_scope::ProgramNodeContribution, 382
- getAll
 - arcs::aubo_scope::FeatureModel, 217
 - arcs::aubo_scope::IoModel, 301
 - arcs::aubo_scope::TcpModel, 476
 - arcs::aubo_scope::VariableModel, 524
- getAnalogInput
 - arcs::aubo_scope::WaitNode, 534
- getAnalogInputValueToProceed
 - arcs::aubo_scope::UntilNode, 499
- getAnalogOutput
 - arcs::aubo_scope::SetNode, 448
- getAngularDeceleration
 - arcs::aubo_scope::UntilNode, 499
- getArguments
 - arcs::aubo_scope::Function, 242
- getAxis
 - arcs::aubo_scope::DirectionNode, 182
- getBaseFeature
 - arcs::aubo_scope::FeatureModel, 217
- getBitOffset
 - arcs::aubo_scope::EtherNetIP, 199
 - arcs::aubo_scope::Profinet, 357
- getBlend
 - arcs::aubo_scope::MoveNode, 321
 - arcs::aubo_scope::WaypointNode, 551
- getBlendParameterType
 - arcs::aubo_scope::UntilNode, 499
- getBlendType
 - arcs::aubo_scope::WaypointNode, 552
- getBoolRegister
 - arcs::aubo_scope::WaitNode, 534
- getBoolValueToSet
 - arcs::aubo_scope::SetNode, 449
- getBoolValueToWaitFor
 - arcs::aubo_scope::WaitNode, 535
- getBugfixVersion
 - SystemApi(系统信息), 71
- getBuildNumber
 - SystemApi(系统信息), 71
- getByName
 - arcs::aubo_scope::FeatureModel, 217
 - arcs::aubo_scope::WaypointModel, 545
- getByteOffset
 - arcs::aubo_scope::EtherNetIP, 199
 - arcs::aubo_scope::Profinet, 357
- getCartesianDeceleration
 - arcs::aubo_scope::UntilNode, 499
- getCenterOfGravity
 - arcs::aubo_scope::Payload, 333
 - arcs::aubo_scope::SetPayloadNode, 454
- getChildren
 - arcs::aubo_scope::TreeNode, 485
- getChildrenNotSuppressed
 - arcs::aubo_scope::TreeNode, 485
- getComment
 - arcs::aubo_scope::CommentNode, 152
- getCompareOperator
 - arcs::aubo_scope::UntilNode, 499
 - arcs::aubo_scope::WaitNode, 535
- getConfigType
 - arcs::aubo_scope::AssignmentNode, 132
 - arcs::aubo_scope::GripperNode, 267
 - arcs::aubo_scope::LoopNode, 310
 - arcs::aubo_scope::MoveNode, 321
 - arcs::aubo_scope::SetNode, 449
 - arcs::aubo_scope::SetPayloadNode, 454
 - arcs::aubo_scope::UntilNode, 499
 - arcs::aubo_scope::WaitNode, 535
 - arcs::aubo_scope::WaypointNode, 552
- getConfigurableDigitalInputConfigure
 - SystemApi(系统信息), 71
- getConfigurableDigitalOutputConfigure
 - SystemApi(系统信息), 71
- getContributorInfo
 - arcs::aubo_scope::Device, 171
 - arcs::aubo_scope::Function, 242
- getCreatedVersion
 - arcs::aubo_scope::InstallationCreationContext, 284
 - arcs::aubo_scope::ProgramCreationContext, 372
- getCurrent
 - arcs::aubo_scope::SetNode, 449
- getCurrentPayload
 - arcs::aubo_scope::PayloadModel, 342
- getCustomDecelerationParameterType
 - arcs::aubo_scope::UntilNode, 499
- getDecelerationParameters
 - arcs::aubo_scope::UntilNode, 499
- getDecelerationParameterType
 - arcs::aubo_scope::UntilNode, 499
- getDefaultName
 - arcs::aubo_scope::Io, 296
- getDefaultPayload
 - arcs::aubo_scope::PayloadModel, 342
- getDefaultTcp
 - arcs::aubo_scope::TcpModel, 476
- getDeviceManager
 - ApplicationApi(应用程序), 66
- getDigitalInput
 - arcs::aubo_scope::WaitNode, 535
- getDigitalInputValueToProceed
 - arcs::aubo_scope::UntilNode, 499
- getDirectionSelectionType

- arcs::aubo_scope::DirectionNode, 182
- getDisplayNames
 - arcs::aubo_scope::Device, 171
 - arcs::aubo_scope::Feature, 206
 - arcs::aubo_scope::Io, 296
 - arcs::aubo_scope::Payload, 333
 - arcs::aubo_scope::ScrewdriverProgram, 427
 - arcs::aubo_scope::TCP, 468
 - arcs::aubo_scope::Variable, 517
 - arcs::aubo_scope::Waypoint, 541
- getDistance
 - arcs::aubo_scope::UntilNode, 500
- getDoubleRegisterInput
 - arcs::aubo_scope::WaitNode, 535
- getENCConfigure
 - SystemApi(系统信息), 71
- getExpression
 - arcs::aubo_scope::AssignmentNode, 132
 - arcs::aubo_scope::DirectionNode, 182
 - arcs::aubo_scope::ElseIfNode, 192
 - arcs::aubo_scope::IfNode, 274
 - arcs::aubo_scope::LoopNode, 311
 - arcs::aubo_scope::SetNode, 449
- getExpressionConfig
 - arcs::aubo_scope::UntilNode, 500
- getExpressionInputConfig
 - arcs::aubo_scope::WaitNode, 535
- getFeature
 - arcs::aubo_scope::DirectionNode, 182
 - arcs::aubo_scope::FeatureContributionModel, 210
 - arcs::aubo_scope::MoveNode, 322
- getFeatureContributionModel
 - InstallationApi(安装), 68
- getFeatureModel
 - ApplicationApi(应用程序), 66
- getFeatures
 - arcs::aubo_scope::FeatureModel, 217
- getFeatureSelectionType
 - arcs::aubo_scope::DirectionNode, 183
- getFeatureVariable
 - arcs::aubo_scope::VariableModel, 525
- getFloatValueToSet
 - arcs::aubo_scope::SetNode, 449
- getFloatValueToWaitFor
 - arcs::aubo_scope::WaitNode, 535
- getForce
 - arcs::aubo_scope::GripActionParameters, 250
- getFunction
 - arcs::aubo_scope::FunctionModel, 245
- getFunctionModel
 - InstallationApi(安装), 68
- getFunctions
 - arcs::aubo_scope::FunctionModel, 246
- getGeomFeture
 - arcs::aubo_scope::FeatureModel, 218
- getGripForceCapability
 - arcs::aubo_scope::RegisteredCapabilities, 399
- getGripperDevice
 - arcs::aubo_scope::GripperNode, 267
- getGrippers
 - arcs::aubo_scope::DeviceManager, 172
- getGripVacuumCapability
 - arcs::aubo_scope::RegisteredCapabilities, 400
- getHomeDefaultPosition
 - SystemApi(系统信息), 72
- getHomePackagePosition
 - SystemApi(系统信息), 72
- getIcon
 - arcs::aubo_scope::InstallationNodeService, 287
 - arcs::aubo_scope::ProgramNodeService, 392
- getId
 - arcs::aubo_scope::Device, 171
 - arcs::aubo_scope::ProgramNodeService, 392
 - arcs::aubo_scope::ScrewdriverProgram, 427
- getInertiaMatrix
 - arcs::aubo_scope::Payload, 333
- getInstallationApi
 - arcs::aubo_scope::InstallationApiProvider, 281
- getInstallationNode
 - arcs::aubo_scope::ApplicationApi, 124, 125
- getInterfaceType
 - arcs::aubo_scope::Io, 297
- getIo
 - arcs::aubo_scope::IoModel, 301
- getIOInputSelectionType
 - arcs::aubo_scope::UntilNode, 500
- getIOModel
 - arcs::aubo_scope::ScrewdriverApiProvider, 417
- getIoModel
 - ApplicationApi(应用程序), 66
 - arcs::aubo_scope::GripperApiProvider, 254
- getIos
 - arcs::aubo_scope::IoModel, 302
- getIpAddress
 - arcs::aubo_scope::ModbusIo, 315
- getJointAcceleration
 - arcs::aubo_scope::MoveNode, 322
 - arcs::aubo_scope::WaypointNode, 552
- getJointDefaultAcceleration
 - SystemApi(系统信息), 72
- getJointDefaultVelocity
 - SystemApi(系统信息), 72
- getJointMaxAcceleration
 - SystemApi(系统信息), 72
- getJointMaxPositions
 - SystemApi(系统信息), 72
- getJointMaxVelocity
 - SystemApi(系统信息), 72
- getJointMinPositions
 - SystemApi(系统信息), 73
- getJointPositions
 - arcs::aubo_scope::Waypoint, 541
 - arcs::aubo_scope::WaypointNode, 552
- getJointSpeed
 - arcs::aubo_scope::MoveNode, 322
 - arcs::aubo_scope::WaypointNode, 552

- getKeyboardManager
 - arcs::aubo_scope::UserInteraction, 505
- getKeys
 - arcs::aubo_scope::DataModel, 163
- getLastSavedVersion
 - arcs::aubo_scope::InstallationCreationContext, 284
 - arcs::aubo_scope::ProgramCreationContext, 372
- getLocale
 - SystemApi(系统信息), 73
- getLocaleForProgrammingLanguage
 - SystemApi(系统信息), 73
- getLoopCount
 - arcs::aubo_scope::LoopNode, 311
- getLoopCountVariable
 - arcs::aubo_scope::LoopNode, 311
- getMajorVersion
 - SystemApi(系统信息), 73
- getMass
 - arcs::aubo_scope::Payload, 333
 - arcs::aubo_scope::SetPayloadNode, 454
- getMaxPayloadDistance
 - SystemApi(系统信息), 73
- getMaxPayloadMass
 - SystemApi(系统信息), 73
- getMaxRangeValue
 - arcs::aubo_scope::AnalogIo, 120
- getMeshes
 - arcs::aubo_scope::RenderInterface, 412
- getMessage
 - arcs::aubo_scope::DoubleRangeValidator, 186
 - arcs::aubo_scope::InputValidator, 276
 - arcs::aubo_scope::IntegerRangeValidator, 290
 - arcs::aubo_scope::PopupNode, 352
 - arcs::aubo_scope::StringLengthValidator, 457
 - arcs::aubo_scope::UIntegerRangeValidator, 490
- getMinorVersion
 - SystemApi(系统信息), 73
- getMinRangeValue
 - arcs::aubo_scope::AnalogIo, 120
- getModbusInput
 - arcs::aubo_scope::WaitNode, 535
- getMotionType
 - arcs::aubo_scope::WaypointNode, 552
- getName
 - arcs::aubo_scope::AuboCapInfo, 133
 - arcs::aubo_scope::Function, 242
- getNodeCreationType
 - arcs::aubo_scope::InstallationCreationContext, 285
 - arcs::aubo_scope::ProgramCreationContext, 372
- getOffset
 - arcs::aubo_scope::TCP, 468
- getOutput
 - arcs::aubo_scope::SetNode, 449
- getOutputSelectionType
 - arcs::aubo_scope::SetNode, 449
- getParametersWarning
 - arcs::aubo_scope::Payload, 333
- getParent
 - arcs::aubo_scope::TreeNode, 485
- getPayload
 - arcs::aubo_scope::PayloadContributionModel, 338
 - arcs::aubo_scope::PayloadModel, 343
- getPayloadContributionModel
 - InstallationApi(安装), 68
- getPayloadModel
 - ApplicationApi(应用程序), 66
- getPayloads
 - arcs::aubo_scope::PayloadModel, 343
- getPopupManager
 - arcs::aubo_scope::UserInteraction, 505
- getPose
 - arcs::aubo_scope::Feature, 206
 - arcs::aubo_scope::Waypoint, 541
 - arcs::aubo_scope::WaypointNode, 552
- getPositionDefinition
 - arcs::aubo_scope::WaypointNode, 552
- getProgramApi
 - arcs::aubo_scope::ProgramApiProvider, 364
- getProgramModel
 - ProgramApi(程序), 69
- getProgramNode
 - arcs::aubo_scope::TreeNode, 485
- getProgramNodeContribution
 - arcs::aubo_scope::ProgramNode, 381
- getProgramNodeFactory
 - arcs::aubo_scope::ProgramModel, 377
- getPulseTime
 - arcs::aubo_scope::SetNode, 449
- getRenderInterface
 - arcs::aubo_scope::UserInteraction, 505
- getResolvedVariable
 - arcs::aubo_scope::ScriptWriter, 439
- getRobotDhParams
 - SystemApi(系统信息), 74
- getRobotProxy
 - arcs::aubo_scope::ApplicationApi, 125
- getRobotThumb
 - arcs::aubo_scope::UserInteraction, 506
- getScrewdriverOperationType
 - arcs::aubo_scope::ProgramSelectionParameters, 394
 - arcs::aubo_scope::ScrewdriverParameters, 426
- getScrewdriverProgram
 - arcs::aubo_scope::ProgramSelectionParameters, 394
 - arcs::aubo_scope::ScrewdriverParameters, 426
- getScrewDrivers
 - arcs::aubo_scope::DeviceManager, 172
- getSerialNumber
 - SystemApi(系统信息), 74
- getSettingType
 - arcs::aubo_scope::GripperNode, 267
- getSignalAddress
 - arcs::aubo_scope::ModbusIo, 316

- getSpeed
 - arcs::aubo_scope::GripActionParameters, 251
 - arcs::aubo_scope::ReleaseActionParameters, 401
- getSpeedCapability
 - arcs::aubo_scope::RegisteredCapabilities, 400
- getStandardAnalogInputConfigure
 - SystemApi(系统信息), 74
- getStandardAnalogOutputConfigure
 - SystemApi(系统信息), 74
- getStandardDigitalInputConfigure
 - SystemApi(系统信息), 74
- getStandardDigitalOutputConfigure
 - SystemApi(系统信息), 74
- getStaticLinkInputConfigure
 - SystemApi(系统信息), 75
- getStaticLinkOutputConfigure
 - SystemApi(系统信息), 75
- getSymbolicName
 - arcs::aubo_scope::AuboCapInfo, 133
- getSystemAPI
 - arcs::aubo_scope::ScrewdriverApiProvider, 417
- getSystemApi
 - arcs::aubo_scope::GripperApiProvider, 254
 - arcs::aubo_scope::InstallationApiProvider, 282
 - arcs::aubo_scope::ProgramApiProvider, 364
 - arcs::aubo_scope::ViewApiProvider, 527
- getTcp
 - arcs::aubo_scope::TcpContributionModel, 471
 - arcs::aubo_scope::TcpModel, 476
- getTcpContributionModel
 - InstallationApi(安装), 68
- getTcpDefaultAcceleration
 - SystemApi(系统信息), 75
- getTcpDefaultVelocity
 - SystemApi(系统信息), 75
- getTcpMaxAcceleration
 - SystemApi(系统信息), 75
- getTcpMaxDistance
 - SystemApi(系统信息), 75
- getTcpMaxVelocity
 - SystemApi(系统信息), 75
- getTcpModel
 - ApplicationApi(应用程序), 66
- getTcpOffset
 - arcs::aubo_scope::Waypoint, 541
 - arcs::aubo_scope::WaypointNode, 553
- getTCPSelection
 - arcs::aubo_scope::MoveNode, 322
- getTcpSelectionType
 - arcs::aubo_scope::MoveNode, 322
- getTime
 - arcs::aubo_scope::WaypointNode, 553
- getTitle
 - arcs::aubo_scope::GripperContribution, 262
 - arcs::aubo_scope::InstallationNodeService, 288
 - arcs::aubo_scope::ProgramNodeContribution, 383
 - arcs::aubo_scope::ProgramNodeService, 392
 - arcs::aubo_scope::ScrewdriverContribution, 425
- getToolAcceleration
 - arcs::aubo_scope::DirectionNode, 183
 - arcs::aubo_scope::MoveNode, 322
 - arcs::aubo_scope::WaypointNode, 553
- getToolAnalogInputConfigure
 - SystemApi(系统信息), 76
- getToolAnalogOutputConfigure
 - SystemApi(系统信息), 76
- getToolDigitalIOConfigure
 - SystemApi(系统信息), 76
- getToolFeature
 - arcs::aubo_scope::FeatureModel, 218
- getToolSpeed
 - arcs::aubo_scope::DirectionNode, 183
 - arcs::aubo_scope::MoveNode, 322
 - arcs::aubo_scope::WaypointNode, 553
- getTreeNode
 - arcs::aubo_scope::ProgramModel, 377
- getType
 - arcs::aubo_scope::DirectionNode, 183
 - arcs::aubo_scope::Io, 297
 - arcs::aubo_scope::PopupNode, 352
 - arcs::aubo_scope::Variable, 517
- getUndoRedoManager
 - ProgramApi(程序), 69
- getUnitType
 - SystemApi(系统信息), 76
- getUseJoints
 - arcs::aubo_scope::MoveNode, 322
- getUserDefinedRobotPosition
 - arcs::aubo_scope::UserInteraction, 506, 507
- getUserInteraction
 - UserInterfaceApi(用户界面), 77
- getUserInterfaceApi
 - arcs::aubo_scope::InstallationApiProvider, 282
 - arcs::aubo_scope::ProgramApiProvider, 364
 - arcs::aubo_scope::ViewApiProvider, 527
- getVacuum
 - arcs::aubo_scope::GripActionParameters, 251
- getValue
 - arcs::aubo_scope::AnalogIo, 120
 - arcs::aubo_scope::DigitalIo, 177
 - arcs::aubo_scope::EtherNetIP, 199
 - arcs::aubo_scope::ModbusIo, 316
 - arcs::aubo_scope::Profinet, 357
- getValueFactory
 - ApplicationApi(应用程序), 67
- getValueStr
 - arcs::aubo_scope::Io, 297
- getVariable
 - arcs::aubo_scope::AssignmentNode, 132
 - arcs::aubo_scope::WaypointNode, 553
- getVariableDefinition
 - arcs::aubo_scope::WaypointNode, 553
- getVariableFactory
 - arcs::aubo_scope::VariableModel, 525
- getVariableModel
 - ApplicationApi(应用程序), 67

- getVendor
 - arcs::aubo_scope::AuboCapInfo, 134
- getVoltage
 - arcs::aubo_scope::SetNode, 450
- getWaitTime
 - arcs::aubo_scope::WaitNode, 535
- getWaypointBlendParameter
 - arcs::aubo_scope::UntilNode, 500
- getWaypointModel
 - ProgramApi(程序), 69
- getWaypoints
 - arcs::aubo_scope::WaypointModel, 545
- getWidth
 - arcs::aubo_scope::GripActionParameters, 251
 - arcs::aubo_scope::ReleaseActionParameters, 401
- getWidthCapability
 - arcs::aubo_scope::RegisteredCapabilities, 400
- GLOBAL
 - arcs::aubo_scope::Variable, 515
- GlobalVariable
 - arcs::aubo_scope::GlobalVariable, 248, 249
- GREATER_THAN
 - arcs::aubo_scope::UntilNode, 496
 - arcs::aubo_scope::WaitNode, 533
- GRIP_ACTION
 - arcs::aubo_scope::GripperNode, 265
- GripperNode
 - arcs::aubo_scope::GripperNode, 266, 267
- Ground
 - arcs::aubo_scope::RenderInterface, 405
- HaltNode
 - arcs::aubo_scope::HaltNode, 270, 271
- hasCoordinate
 - arcs::aubo_scope::RenderInterface, 412
- hashCode
 - arcs::aubo_scope::Feature, 206
 - arcs::aubo_scope::Io, 297
 - arcs::aubo_scope::Payload, 334
 - arcs::aubo_scope::TCP, 468
 - arcs::aubo_scope::Variable, 517
 - arcs::aubo_scope::Waypoint, 541
- hasPointCloud
 - arcs::aubo_scope::RenderInterface, 412
- hasWaypoint
 - arcs::aubo_scope::RenderInterface, 413
- HOLE_SEARCH
 - arcs::aubo_scope::AssemblyNode, 127
- ifCondition
 - arcs::aubo_scope::ScriptWriter, 439
- IfNode
 - arcs::aubo_scope::IfNode, 273, 274
- ifNotCondition
 - arcs::aubo_scope::ScriptWriter, 439
- IGNORE_ACTIVE_TCP
 - arcs::aubo_scope::MoveNode, 320
- IMPERIAL
 - arcs::aubo_scope, 105
- include 目录参考, 86
- include/aubo_caps 目录参考, 79
- include/aubo_caps/class_forward.h, 559, 561
- include/aubo_caps/contribution 目录参考, 81
- include/aubo_caps/contribution/contribution_provider.h, 562, 563
- include/aubo_caps/contribution/driver 目录参考, 84
- include/aubo_caps/contribution/driver/gripper 目录参考, 86
- include/aubo_caps/contribution/driver/gripper/capability 目录参考, 81
- include/aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h, 564, 565
- include/aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h, 566, 567
- include/aubo_caps/contribution/driver/gripper/capability/registered_capabilities.h, 568
- include/aubo_caps/contribution/driver/gripper/capability/speed_capability.h, 570, 571
- include/aubo_caps/contribution/driver/gripper/capability/width_capability.h, 572, 573
- include/aubo_caps/contribution/driver/gripper/grip_action_parameters.h, 574, 575
- include/aubo_caps/contribution/driver/gripper/gripper_api_provider.h, 577, 578
- include/aubo_caps/contribution/driver/gripper/gripper_configuration.h, 579, 580
- include/aubo_caps/contribution/driver/gripper/gripper_contribution.h, 583, 584
- include/aubo_caps/contribution/driver/gripper/release_action_parameters.h, 587, 588
- include/aubo_caps/contribution/driver/screw_driver 目录参考, 90
- include/aubo_caps/contribution/driver/screw_driver/screw_driver_api_provider.h, 590, 591
- include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h, 591, 593
- include/aubo_caps/contribution/driver/screw_driver/screw_driver_contribution.h, 599, 600
- include/aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h, 602, 604
- include/aubo_caps/contribution/installation 目录参考, 87
- include/aubo_caps/contribution/installation/installation_api_provider.h, 605, 606
- include/aubo_caps/contribution/installation/installation_contribution_configuration.h, 608, 609
- include/aubo_caps/contribution/installation/installation_creation_context.h, 609, 610
- include/aubo_caps/contribution/installation/installation_node_contribution.h, 611, 612
- include/aubo_caps/contribution/installation/installation_node_service.h, 613, 614
- include/aubo_caps/contribution/installation/installation_node_view.h, 616, 617
- include/aubo_caps/contribution/program 目录参

- 考, 89
- include/aubo_caps/contribution/program/program_api_provider.h, 618, 619
- include/aubo_caps/contribution/program/program_contribution_model.h, 620, 621
- include/aubo_caps/contribution/program/program_creation_context.h, 623, 624
- include/aubo_caps/contribution/program/program_model.h, 624, 625
- include/aubo_caps/contribution/program/program_model_loader.h, 627, 628
- include/aubo_caps/contribution/view_api_provider.h, 630, 631
- include/aubo_caps/contribution/view_api_provider.h, 631, 632
- include/aubo_caps/domain 目录参考, 83
- include/aubo_caps/domain/application_api.h, 633, 634
- include/aubo_caps/domain/data 目录参考, 82
- include/aubo_caps/domain/data/data_model.h, 637, 638
- include/aubo_caps/domain/device 目录参考, 82
- include/aubo_caps/domain/device/device.h, 647, 648
- include/aubo_caps/domain/device/device_manager.h, 650, 651
- include/aubo_caps/domain/device/device_registration_manager.h, 652, 653
- include/aubo_caps/domain/device/gripper_device.h, 654, 655
- include/aubo_caps/domain/extension_info.h, 656, 658
- include/aubo_caps/domain/feature 目录参考, 85
- include/aubo_caps/domain/feature/base_feature.h, 659
- include/aubo_caps/domain/feature/feature.h, 660, 661
- include/aubo_caps/domain/feature/feature_contribution_model.h, 663, 664
- include/aubo_caps/domain/feature/feature_line.h, 666, 667
- include/aubo_caps/domain/feature/feature_model.h, 668, 669
- include/aubo_caps/domain/feature/feature_plane.h, 671
- include/aubo_caps/domain/feature/feature_point.h, 672, 673
- include/aubo_caps/domain/feature/feature_tool.h, 674
- include/aubo_caps/domain/function 目录参考, 85
- include/aubo_caps/domain/function/function.h, 675, 676
- include/aubo_caps/domain/function/function_model.h, 678, 679
- include/aubo_caps/domain/installation 目录参考, 87
- include/aubo_caps/domain/installation/installation_api.h, 681, 682
- include/aubo_caps/domain/io 目录参考, 88
- include/aubo_caps/domain/io/analog_io.h, 684, 686
- include/aubo_caps/domain/io/boolean_register.h, 687, 689
- include/aubo_caps/domain/io/configuration.h, 687, 689
- include/aubo_caps/domain/io/digital_io.h, 689, 691
- include/aubo_caps/domain/io/double_register.h, 692, 694
- include/aubo_caps/domain/io/ethernet_ip.h, 694, 695
- include/aubo_caps/domain/io/integer_register.h, 697
- include/aubo_caps/domain/io/io.h, 698, 699
- include/aubo_caps/domain/io/io_model.h, 701, 702
- include/aubo_caps/domain/io/modbus_io.h, 703, 705
- include/aubo_caps/domain/io/profinet.h, 707
- include/aubo_caps/domain/io/register.h, 709, 710
- include/aubo_caps/domain/payload 目录参考, 89
- include/aubo_caps/domain/payload/payload.h, 711, 712
- include/aubo_caps/domain/payload/payload_contribution_model.h, 715, 716
- include/aubo_caps/domain/payload/payload_model.h, 721, 723
- include/aubo_caps/domain/program 目录参考, 90
- include/aubo_caps/domain/program/nodes 目录参考, 88
- include/aubo_caps/domain/program/nodes/builtin 目录参考, 80
- include/aubo_caps/domain/program/nodes/builtin/assembly_node.h, 724, 725
- include/aubo_caps/domain/program/nodes/builtin/assignment_node.h, 726, 727
- include/aubo_caps/domain/program/nodes/builtin/break_node.h, 729, 730
- include/aubo_caps/domain/program/nodes/builtin/call_node.h, 731, 732
- include/aubo_caps/domain/program/nodes/builtin/case_node.h, 733, 734
- include/aubo_caps/domain/program/nodes/builtin/circle_move_node.h, 735, 736
- include/aubo_caps/domain/program/nodes/builtin/comment_node.h, 737, 738
- include/aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h, 739, 740
- include/aubo_caps/domain/program/nodes/builtin/default_case_node.h, 741, 742
- include/aubo_caps/domain/program/nodes/builtin/direction_node.h, 743, 744
- include/aubo_caps/domain/program/nodes/builtin/else_if_node.h, 746, 748
- include/aubo_caps/domain/program/nodes/builtin/else_node.h, 749, 750
- include/aubo_caps/domain/program/nodes/builtin/folder_node.h, 751, 752
- include/aubo_caps/domain/program/nodes/builtin/force_node.h, 753, 754

[illegible]

- incrementVariable
 - arcs::aubo_scope::ScriptWriter, [439](#)
- INPUT
 - arcs::aubo_scope::WaypointNode, [550](#)
- INPUT_BOOL
 - arcs::aubo_scope::AssignmentNode, [131](#)
- INPUT_FLOAT
 - arcs::aubo_scope::AssignmentNode, [131](#)
- INPUT_INTEGER
 - arcs::aubo_scope::AssignmentNode, [131](#)
- INPUT_STRING
 - arcs::aubo_scope::AssignmentNode, [131](#)
- InputMode
 - arcs::aubo_scope::AssignmentNode, [130](#)
- insertChildAfter
 - arcs::aubo_scope::TreeNode, [485](#)
- insertChildBefore
 - arcs::aubo_scope::TreeNode, [486](#)
- installation_node_view.h
 - WidgetHandle, [617](#)
- InstallationApi
 - arcs::aubo_scope::InstallationApi, [278](#), [279](#)
- InstallationApi(安装), [67](#)
 - getFeatureContributionModel, [68](#)
 - getFunctionModel, [68](#)
 - getPayloadContributionModel, [68](#)
 - getTcpContributionModel, [68](#)
- InstallationApiProvider
 - arcs::aubo_scope::InstallationApiProvider, [280](#), [281](#)
- InstallationCreationContext
 - arcs::aubo_scope::InstallationCreationContext, [283](#), [284](#)
- INTEGER
 - arcs::aubo_scope::Io, [295](#)
- IntegerRangeValidator
 - arcs::aubo_scope::IntegerRangeValidator, [290](#)
- interface_
 - arcs::aubo_scope::AnalogIoFilter, [121](#)
 - arcs::aubo_scope::DigitalIoFilter, [178](#)
- InterfaceType
 - arcs::aubo_scope::Io, [294](#)
- Io
 - arcs::aubo_scope::Io, [295](#), [296](#)
- IO_INPUT
 - arcs::aubo_scope::UntilNode, [496](#)
- IOInputSelectionType
 - arcs::aubo_scope::UntilNode, [497](#)
- IoModel
 - arcs::aubo_scope::IoModel, [300](#), [301](#)
- IoType
 - arcs::aubo_scope::Io, [295](#)
- is_input_
 - arcs::aubo_scope::AnalogIoFilter, [121](#)
 - arcs::aubo_scope::DigitalIoFilter, [178](#)
 - arcs::aubo_scope::EtherNetIPFilter, [201](#)
 - arcs::aubo_scope::ModbusIoFilter, [317](#)
 - arcs::aubo_scope::ProfinetFilter, [359](#)
- is_max_inclusive_
 - arcs::aubo_scope::DoubleRangeValidator, [186](#)
- is_min_inclusive_
 - arcs::aubo_scope::DoubleRangeValidator, [186](#)
- isCurrent
 - arcs::aubo_scope::AnalogIo, [120](#)
- isDefined
 - arcs::aubo_scope::Feature, [207](#)
 - arcs::aubo_scope::ProgramNodeContribution, [383](#)
 - arcs::aubo_scope::Waypoint, [541](#)
- isEvaluateContinously
 - arcs::aubo_scope::LoopNode, [311](#)
- isGripDetectionEnabled
 - arcs::aubo_scope::GripActionParameters, [251](#)
- isInProgramTree
 - arcs::aubo_scope::TreeNode, [486](#)
- isInput
 - arcs::aubo_scope::Io, [297](#)
- isJoggable
 - arcs::aubo_scope::Feature, [207](#)
- isPopupCountZero
 - arcs::aubo_scope::PopupManager, [349](#)
- isRealRobot
 - SystemApi(系统信息), [76](#)
- isReleaseDetectionEnabled
 - arcs::aubo_scope::ReleaseActionParameters, [402](#)
- isResolvable
 - arcs::aubo_scope::Device, [171](#)
 - arcs::aubo_scope::Feature, [207](#)
 - arcs::aubo_scope::Io, [297](#)
 - arcs::aubo_scope::Payload, [334](#)
 - arcs::aubo_scope::PersistedVariable, [346](#)
 - arcs::aubo_scope::TCP, [469](#)
- isSet
 - arcs::aubo_scope::DataModel, [163](#)
- isSlave
 - arcs::aubo_scope::EtherNetIP, [199](#)
 - arcs::aubo_scope::ModbusIo, [316](#)
 - arcs::aubo_scope::Profinet, [358](#)
- isSuppressed
 - arcs::aubo_scope::TreeNode, [487](#)
- isUsable
 - arcs::aubo_scope::Payload, [334](#)
- isValid
 - arcs::aubo_scope::DoubleRangeValidator, [186](#)
 - arcs::aubo_scope::InputValidator, [276](#)
 - arcs::aubo_scope::IntegerRangeValidator, [290](#)
 - arcs::aubo_scope::StringLengthValidator, [457](#)
 - arcs::aubo_scope::UIntegerRangeValidator, [490](#)
- isVariable
 - arcs::aubo_scope::Feature, [207](#)
- isVoltage
 - arcs::aubo_scope::AnalogIo, [120](#)
- JOINT_MOTION
 - arcs::aubo_scope::WaypointNode, [549](#)
- KeyboardManager
 - arcs::aubo_scope::KeyboardManager, [304](#), [305](#)

- killThread
 - arcs::aubo_scope::ScriptWriter, [440](#)
- label
 - arcs::aubo_scope::IoConfigure, [299](#)
- LESS_THAN
 - arcs::aubo_scope::UntilNode, [496](#)
 - arcs::aubo_scope::WaitNode, [533](#)
- lineFeed
 - arcs::aubo_scope::ScriptWriter, [440](#)
- LINKIO
 - arcs::aubo_scope::Io, [295](#)
- LOAD
 - arcs::aubo_scope::InstallationCreationContext, [283](#)
 - arcs::aubo_scope::ProgramCreationContext, [371](#)
- locateDescendantTreeNode
 - arcs::aubo_scope::TreeNode, [487](#)
- LoopNode
 - arcs::aubo_scope::LoopNode, [309](#), [310](#)
- LOOSEN
 - arcs::aubo_scope, [105](#)
- MASS_0KG
 - arcs::aubo_scope::Payload, [331](#)
- MASS_AND_CENTER_OF_GRAVITY_EXCEEDED
 - arcs::aubo_scope::Payload, [331](#)
- max_length_
 - arcs::aubo_scope::StringLengthValidator, [458](#)
- max_value_
 - arcs::aubo_scope::DoubleRangeValidator, [186](#)
 - arcs::aubo_scope::IntegerRangeValidator, [291](#)
 - arcs::aubo_scope::UIntegerRangeValidator, [490](#)
- maxs
 - arcs::aubo_scope::IoConfigure, [299](#)
- MESSAGE
 - arcs::aubo_scope::PopupNode, [351](#)
- METRIC
 - arcs::aubo_scope, [105](#)
- min_length_
 - arcs::aubo_scope::StringLengthValidator, [458](#)
- min_value_
 - arcs::aubo_scope::DoubleRangeValidator, [186](#)
 - arcs::aubo_scope::IntegerRangeValidator, [291](#)
 - arcs::aubo_scope::UIntegerRangeValidator, [490](#)
- mins
 - arcs::aubo_scope::IoConfigure, [299](#)
- MODBUS
 - arcs::aubo_scope::Io, [295](#)
- ModbusIo
 - arcs::aubo_scope::ModbusIo, [314](#), [315](#)
- ModbusIoFilter
 - arcs::aubo_scope::ModbusIoFilter, [317](#)
- MOTION
 - arcs::aubo_scope::ForceNode, [236](#)
- MotionType
 - arcs::aubo_scope::WaypointNode, [549](#)
- MOVE_J
 - arcs::aubo_scope::MoveNode, [319](#)
- MOVE_L
 - arcs::aubo_scope::MoveNode, [319](#)
- MOVE_S
 - arcs::aubo_scope::MoveNode, [320](#)
- MOVE_TCP
 - arcs::aubo_scope::MoveNode, [320](#)
- MoveNode
 - arcs::aubo_scope::MoveNode, [320](#), [321](#)
- names
 - arcs::aubo_scope::IoConfigure, [299](#)
- NEW
 - arcs::aubo_scope::InstallationCreationContext, [283](#)
 - arcs::aubo_scope::ProgramCreationContext, [371](#)
- NEW_WORKPIECE
 - arcs::aubo_scope::ConveyorTrackingNode, [155](#)
- NO_ACTION
 - arcs::aubo_scope::SetNode, [447](#)
- NO_BLEND
 - arcs::aubo_scope::UntilNode, [496](#)
 - arcs::aubo_scope::WaypointNode, [548](#)
- NO_OUTPUT
 - arcs::aubo_scope::SetNode, [447](#)
- NO_TYPE
 - arcs::aubo_scope::SetNode, [446](#)
 - arcs::aubo_scope::WaitNode, [533](#)
- NO_WAIT
 - arcs::aubo_scope::WaitNode, [533](#)
- NO_WARNING
 - arcs::aubo_scope::Payload, [331](#)
- NodeCreationType
 - arcs::aubo_scope::InstallationCreationContext, [283](#)
 - arcs::aubo_scope::ProgramCreationContext, [370](#)
- NONE
 - arcs::aubo_scope::AssemblyNode, [127](#)
- None
 - arcs::aubo_scope::RenderInterface, [405](#)
- NONE_CONFIG_TYPE
 - arcs::aubo_scope::UntilNode, [496](#)
- NONE_INPUT_SELECT
 - arcs::aubo_scope::UntilNode, [497](#)
- note
 - arcs::aubo_scope::ScriptWriter, [440](#)
- notifyPopupHidden
 - arcs::aubo_scope::PopupManager, [349](#)
- notifyPopupShown
 - arcs::aubo_scope::PopupManager, [349](#)
- num
 - arcs::aubo_scope::IoConfigure, [299](#)
- Obstacle
 - arcs::aubo_scope::RenderInterface, [405](#)
- OfflineTrackNode
 - arcs::aubo_scope::OfflineTrackNode, [326](#), [327](#)
- openView
 - arcs::aubo_scope::InstallationNodeContribution, [286](#)

- arcs::aubo_scope::ProgramNodeContribution, 383
- OPERATOR
 - arcs::aubo_scope::AssignmentNode, 130
- operator()
 - arcs::aubo_scope::AnalogIoFilter, 121
 - arcs::aubo_scope::DigitalIoFilter, 178
 - arcs::aubo_scope::EtherNetIPFilter, 201
 - arcs::aubo_scope::FeatureDefinedFilter, 212
 - arcs::aubo_scope::FeatureJoggableFilter, 212
 - arcs::aubo_scope::FeatureVariableFilter, 231
 - arcs::aubo_scope::GlobalVariableFilter, 250
 - arcs::aubo_scope::InputIoFilter, 275
 - arcs::aubo_scope::ModbusIoFilter, 317
 - arcs::aubo_scope::OutputIoFilter, 327
 - arcs::aubo_scope::PersistedVariableFilter, 347
 - arcs::aubo_scope::ProfinetFilter, 359
- OUTPUT
 - arcs::aubo_scope::SetNode, 447
- OutputSelectionType
 - arcs::aubo_scope::SetNode, 447
- PalletNode
 - arcs::aubo_scope::PalletNode, 329, 330
- ParametersWarning
 - arcs::aubo_scope::Payload, 331
- ParameterType
 - arcs::aubo_scope::DirectionNode, 181
- Payload
 - arcs::aubo_scope::Payload, 332
- PAYLOAD_SELECTION
 - arcs::aubo_scope::GripperNode, 266
 - arcs::aubo_scope::SetPayloadNode, 452
- PayloadContributionModel
 - arcs::aubo_scope::PayloadContributionModel, 336
- PayloadModel
 - arcs::aubo_scope::PayloadModel, 341, 342
- PersistedVariable
 - arcs::aubo_scope::PersistedVariable, 345, 346
- POINT
 - arcs::aubo_scope::ForceNode, 236
- Point
 - arcs::aubo_scope::RenderInterface, 405
- PointCloud
 - arcs::aubo_scope::RenderInterface, 405
- popupCount
 - arcs::aubo_scope::PopupManager, 349
- PopupManager
 - arcs::aubo_scope::PopupManager, 348, 349
- PopupNode
 - arcs::aubo_scope::PopupNode, 351, 352
- PositionDefinition
 - arcs::aubo_scope::WaypointNode, 549
- postAction
 - arcs::aubo_scope::ProgramNodeContribution, 383
- PREVIOUS_WORKPIECE
 - arcs::aubo_scope::ConveyorTrackingNode, 155
- PROFINET
 - arcs::aubo_scope::Io, 295
- Profinet
 - arcs::aubo_scope::Profinet, 356, 357
 - ProfinetFilter
 - arcs::aubo_scope::ProfinetFilter, 359
 - ProgramApi
 - arcs::aubo_scope::ProgramApi, 361, 362
 - ProgramApi(程序), 68
 - getProgramModel, 69
 - getUndoRedoManager, 69
 - getWaypointModel, 69
 - ProgramApiProvider
 - arcs::aubo_scope::ProgramApiProvider, 363, 364
 - ProgramContributionConfiguration
 - arcs::aubo_scope::ProgramContributionConfiguration, 366, 367
 - ProgramCreationContext
 - arcs::aubo_scope::ProgramCreationContext, 371, 372
 - ProgramHomeNode
 - arcs::aubo_scope::ProgramHomeNode, 374, 375
 - ProgramModel
 - arcs::aubo_scope::ProgramModel, 376, 377
 - ProgramNode
 - arcs::aubo_scope::ProgramNode, 380, 381
 - ProgramNodeFactory
 - arcs::aubo_scope::ProgramNodeFactory, 385, 386
 - range_func_
 - arcs::aubo_scope::DoubleRangeValidator, 187
 - arcs::aubo_scope::IntegerRangeValidator, 291
 - arcs::aubo_scope::StringLengthValidator, 458
 - arcs::aubo_scope::UIntegerRangeValidator, 491
 - range_func_with_flags_
 - arcs::aubo_scope::DoubleRangeValidator, 187
 - RATED_MASS_EXCEEDED
 - arcs::aubo_scope::Payload, 331
 - REACHED_WAYPOINT
 - arcs::aubo_scope::UntilNode, 496
 - RealRobot
 - arcs::aubo_scope::RenderInterface, 405
 - recordChanges
 - arcs::aubo_scope::UndoRedoManager, 492
 - RecordTrackNode
 - arcs::aubo_scope::RecordTrackNode, 396, 397
 - registerCodeEditorKeyboard
 - arcs::aubo_scope::KeyboardManager, 305
 - registerDoubleKeyboard
 - arcs::aubo_scope::KeyboardManager, 305
 - registerDriveScrewNotOKCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, 419
 - registerDriveScrewOKCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, 420
 - registerExpressionKeyboard
 - arcs::aubo_scope::KeyboardManager, 305
 - registerFeedScrewCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, 420
 - registerGripDetectedCapability
 - arcs::aubo_scope::GripperConfiguration, 255
 - registerGripperContribution

- arcs::aubo_scope::DeviceRegistrationManager, 173
- registerGrippingForceCapability
 - arcs::aubo_scope::GripperConfiguration, 255
- registerGrippingVacuumCapability
 - arcs::aubo_scope::GripperConfiguration, 256
- registerIntegerKeyboard
 - arcs::aubo_scope::KeyboardManager, 305
- registerIPAddressKeyboard
 - arcs::aubo_scope::KeyboardManager, 305
- registerNumberKeyboard
 - arcs::aubo_scope::KeyboardManager, 305
- registerOperationTypeCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, 421
- registerPasswordKeyboard
 - arcs::aubo_scope::KeyboardManager, 306
- registerPositiveDoubleKeyboard
 - arcs::aubo_scope::KeyboardManager, 306
- registerPositiveIntegerKeyboard
 - arcs::aubo_scope::KeyboardManager, 306
- registerPrepareToStartScrewdriverCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, 421
- registerProgramSelectionCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, 422
- registerReleaseDetectedCapability
 - arcs::aubo_scope::GripperConfiguration, 257
- registerScrewDriverContribution
 - arcs::aubo_scope::DeviceRegistrationManager, 173
- registerScrewdriverReadyCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, 423
- registerSpeedCapability
 - arcs::aubo_scope::GripperConfiguration, 258
- registerStandardKeyboard
 - arcs::aubo_scope::KeyboardManager, 306
- registerStringKeyboard
 - arcs::aubo_scope::KeyboardManager, 307
- registerWidthCapability
 - arcs::aubo_scope::GripperConfiguration, 259
- RELATIVE_POSITION
 - arcs::aubo_scope::WaypointNode, 549
- RelativeOffsetType
 - arcs::aubo_scope::WaypointNode, 549
- RelativePositionType
 - arcs::aubo_scope::WaypointNode, 550
- RELEASE_ACTION
 - arcs::aubo_scope::GripperNode, 266
- remove
 - arcs::aubo_scope::DataModel, 163
- removeChild
 - arcs::aubo_scope::TreeNode, 487
- removeCoordinate
 - arcs::aubo_scope::RenderInterface, 413
- removeCube
 - arcs::aubo_scope::RenderInterface, 413
- removeFeature
 - arcs::aubo_scope::FeatureContributionModel, 210
- removeFunction
 - arcs::aubo_scope::FunctionModel, 246
- removeLine
 - arcs::aubo_scope::RenderInterface, 413
- removeMesh
 - arcs::aubo_scope::RenderInterface, 414
- removePayload
 - arcs::aubo_scope::PayloadContributionModel, 339
- removePlane
 - arcs::aubo_scope::RenderInterface, 414
- removePointCloud
 - arcs::aubo_scope::RenderInterface, 414
- removeTcp
 - arcs::aubo_scope::TcpContributionModel, 472
- removeWaypoint
 - arcs::aubo_scope::RenderInterface, 414
 - arcs::aubo_scope::WaypointModel, 545
- renameFeature
 - arcs::aubo_scope::FeatureContributionModel, 210
- renameWaypoint
 - arcs::aubo_scope::WaypointModel, 545
- RenderInterface
 - arcs::aubo_scope::RenderInterface, 405, 406
- requestUserToMoveJoint
 - arcs::aubo_scope::UserInteraction, 507, 508
- requestUserToMoveLine
 - arcs::aubo_scope::UserInteraction, 508, 509
- RESERVE_QUEUE
 - arcs::aubo_scope::ConveyorTrackingNode, 155
- returnMethod
 - arcs::aubo_scope::ScriptWriter, 440
- RobotMovementCallback
 - arcs::aubo_scope, 104
- RobotPositionCallback
 - arcs::aubo_scope, 104
- runThread
 - arcs::aubo_scope::ScriptWriter, 440
- safety_functions
 - arcs::aubo_scope::IoConfigure, 299
- safety_functions_fixed
 - arcs::aubo_scope::IoConfigure, 299
- SafetyPlane
 - arcs::aubo_scope::RenderInterface, 405
- ScrewdriverOperationType
 - arcs::aubo_scope, 105
- ScrewdrivingNode
 - arcs::aubo_scope::ScrewdrivingNode, 429, 430
- ScriptNode
 - arcs::aubo_scope::ScriptNode, 432, 433
- ScriptWriter
 - arcs::aubo_scope::ScriptWriter, 435, 436
- SeekNode
 - arcs::aubo_scope::SeekNode, 443, 444
- sequences
 - arcs::aubo_scope::IoConfigure, 299
- set
 - arcs::aubo_scope::DataModel, 164–167
- setAllowBreakpointOnChildNodesInSubtree

- arcs::aubo_scope::ProgramContributionConfiguration, 367
- setAllowBreakpointOnNode
 - arcs::aubo_scope::ProgramContributionConfiguration, 367
- setAllowInsertOnlyIntoNodes
 - arcs::aubo_scope::ProgramContributionConfiguration, 367
- setAllowStartFromChildNodesInSubtree
 - arcs::aubo_scope::ProgramContributionConfiguration, 368
- setAllowStartFromNode
 - arcs::aubo_scope::ProgramContributionConfiguration, 368
- setAnalogInput
 - arcs::aubo_scope::WaitNode, 535
- setAnalogInputValueToProceed
 - arcs::aubo_scope::UntilNode, 500
- setAnalogOutput
 - arcs::aubo_scope::SetNode, 450
- setAngularDeceleration
 - arcs::aubo_scope::UntilNode, 500
- setAxis
 - arcs::aubo_scope::DirectionNode, 183
- setBlend
 - arcs::aubo_scope::MoveNode, 322
- setBlendParameterType
 - arcs::aubo_scope::UntilNode, 501
- setBlendTrack
 - arcs::aubo_scope::RenderInterface, 414, 415
- setBoolRegister
 - arcs::aubo_scope::WaitNode, 536
- setBoolValueToSet
 - arcs::aubo_scope::SetNode, 450
- setBoolValueToWaitFor
 - arcs::aubo_scope::WaitNode, 536
- setCartesianDeceleration
 - arcs::aubo_scope::UntilNode, 501
- setCenterOfGravity
 - arcs::aubo_scope::SetPayloadNode, 454
- setChildrenAllowed
 - arcs::aubo_scope::ProgramContributionConfiguration, 368
- setChildrenAllowedOnly
 - arcs::aubo_scope::ProgramContributionConfiguration, 369
- setChildSequenceLocked
 - arcs::aubo_scope::TreeNode, 487
- setComment
 - arcs::aubo_scope::CommentNode, 152
- setCompareOperator
 - arcs::aubo_scope::UntilNode, 501
 - arcs::aubo_scope::WaitNode, 536
- setConfigType
 - arcs::aubo_scope::AssignmentNode, 132
 - arcs::aubo_scope::GripperNode, 267
 - arcs::aubo_scope::LoopNode, 311
 - arcs::aubo_scope::MoveNode, 322
- arcs::aubo_scope::SetNode, 450
 - arcs::aubo_scope::SetPayloadNode, 454
 - arcs::aubo_scope::UntilNode, 501
 - arcs::aubo_scope::WaitNode, 536
- setCurrent
 - arcs::aubo_scope::SetNode, 450
- setCustomDecelerationParameterType
 - arcs::aubo_scope::UntilNode, 501
- setDecelerationParameters
 - arcs::aubo_scope::UntilNode, 501
- setDecelerationParameterType
 - arcs::aubo_scope::UntilNode, 501
- setDeprecated
 - arcs::aubo_scope::ProgramContributionConfiguration, 369
- setDigitalInput
 - arcs::aubo_scope::WaitNode, 537
- setDigitalInputValueToProceed
 - arcs::aubo_scope::UntilNode, 501
- setDirectionSelectionType
 - arcs::aubo_scope::DirectionNode, 183
- setDisplayOrderId
 - arcs::aubo_scope::ProgramContributionConfiguration, 369
- setDistance
 - arcs::aubo_scope::UntilNode, 502
- setDoubleRegisterInput
 - arcs::aubo_scope::WaitNode, 537
- setEvaluateContinously
 - arcs::aubo_scope::LoopNode, 311
- setExpression
 - arcs::aubo_scope::AssignmentNode, 132
 - arcs::aubo_scope::DirectionNode, 183
 - arcs::aubo_scope::ElseIfNode, 192
 - arcs::aubo_scope::IfNode, 274
 - arcs::aubo_scope::LoopNode, 311
 - arcs::aubo_scope::SetNode, 450
- setExpressionConfig
 - arcs::aubo_scope::UntilNode, 502
- setExpressionInputConfig
 - arcs::aubo_scope::WaitNode, 537
- setFeature
 - arcs::aubo_scope::DirectionNode, 183
 - arcs::aubo_scope::MoveNode, 322, 323
- setFeatureSelectionType
 - arcs::aubo_scope::DirectionNode, 183
- setFixedPositionConfig
 - arcs::aubo_scope::WaypointNode, 553, 554
- setFloatValueToSet
 - arcs::aubo_scope::SetNode, 450
- setFloatValueToWaitFor
 - arcs::aubo_scope::WaitNode, 538
- setFolderName
 - arcs::aubo_scope::FolderNode, 234
- setIOInputSelectionType
 - arcs::aubo_scope::UntilNode, 502
- setJointAcceleration
 - arcs::aubo_scope::MoveNode, 323

- setJointMotionParameters
 - arcs::aubo_scope::WaypointNode, 554
- setJointSpeed
 - arcs::aubo_scope::MoveNode, 323
- setLabel
 - arcs::aubo_scope::ScriptWriter, 440
- setLoopCount
 - arcs::aubo_scope::LoopNode, 311
- setLoopCountVariable
 - arcs::aubo_scope::LoopNode, 312
- setMass
 - arcs::aubo_scope::SetPayloadNode, 454
- setMessage
 - arcs::aubo_scope::PopupNode, 352
- setModbusInput
 - arcs::aubo_scope::WaitNode, 538
- setMoveTrack
 - arcs::aubo_scope::RenderInterface, 415
- SetNode
 - arcs::aubo_scope::SetNode, 447, 448
- setOperationLogDisplayName
 - arcs::aubo_scope::UserInteraction, 509
- setOutput
 - arcs::aubo_scope::SetNode, 450
- setPayload
 - arcs::aubo_scope::ScriptWriter, 440
- SetPayloadNode
 - arcs::aubo_scope::SetPayloadNode, 453, 454
- setPopupCountChangedCallback
 - arcs::aubo_scope::PopupManager, 349
- setPulseTime
 - arcs::aubo_scope::SetNode, 450
- setRenderEmbedded
 - arcs::aubo_scope::RenderInterface, 416
- setRobotIndex
 - arcs::aubo_scope::ScriptWriter, 440
- setSharedMotionParameters
 - arcs::aubo_scope::WaypointNode, 554
- setShowOptions
 - arcs::aubo_scope::RenderInterface, 416
- setTcp
 - arcs::aubo_scope::ScriptWriter, 440
- setTCPSelection
 - arcs::aubo_scope::MoveNode, 323
- setTcpSelectionType
 - arcs::aubo_scope::MoveNode, 323
- setTimeMotionParameters
 - arcs::aubo_scope::WaypointNode, 555
- SettingType
 - arcs::aubo_scope::GripperNode, 266
- setToolAcceleration
 - arcs::aubo_scope::DirectionNode, 183
 - arcs::aubo_scope::MoveNode, 324
- setToolMotionParameters
 - arcs::aubo_scope::WaypointNode, 555
- setToolSpeed
 - arcs::aubo_scope::DirectionNode, 183
 - arcs::aubo_scope::MoveNode, 324
- setType
 - arcs::aubo_scope::DirectionNode, 184
 - arcs::aubo_scope::PopupNode, 353
- setUseJoints
 - arcs::aubo_scope::MoveNode, 324
- setUserInsertable
 - arcs::aubo_scope::ProgramContributionConfiguration, 369
- setValue
 - arcs::aubo_scope::AnalogIo, 120
 - arcs::aubo_scope::DigitalIo, 177
 - arcs::aubo_scope::EtherNetIP, 200
 - arcs::aubo_scope::ModbusIo, 316
 - arcs::aubo_scope::Profinet, 358
- setVariable
 - arcs::aubo_scope::AssignmentNode, 132
- setVariablePositionConfig
 - arcs::aubo_scope::WaypointNode, 555, 556
- setVoltage
 - arcs::aubo_scope::SetNode, 450
- setWaitTime
 - arcs::aubo_scope::WaitNode, 538
- setWaypointBlendParameter
 - arcs::aubo_scope::UntilNode, 502
- SHARED
 - arcs::aubo_scope::DirectionNode, 181
- SHARED_BLEND
 - arcs::aubo_scope::WaypointNode, 548
- SHARED_DECELERATION
 - arcs::aubo_scope::UntilNode, 497
- SHARED_MOTION
 - arcs::aubo_scope::DirectionNode, 181
 - arcs::aubo_scope::WaypointNode, 549
- ShowOptions
 - arcs::aubo_scope::RenderInterface, 404
- SIMPLE
 - arcs::aubo_scope::ForceNode, 236
- SINGLE_DIGITAL_PULSE
 - arcs::aubo_scope::SetNode, 447
- sleep
 - arcs::aubo_scope::ScriptWriter, 440
- STANDARD
 - arcs::aubo_scope::Io, 294
- StringLengthValidator
 - arcs::aubo_scope::StringLengthValidator, 457
- SubProgNode
 - arcs::aubo_scope::SubProgNode, 459, 460
- SwitchNode
 - arcs::aubo_scope::SwitchNode, 462, 463
- sync
 - arcs::aubo_scope::ScriptWriter, 441
- SystemApi
 - arcs::aubo_scope::SystemApi, 465, 466
- SystemApi(系统信息), 69
 - getBugfixVersion, 71
 - getBuildNumber, 71
 - getConfigurableDigitalInputConfigure, 71
 - getConfigurableDigitalOutputConfigure, 71

- getENCConfigure, 71
- getHomeDefaultPosition, 72
- getHomePackagePosition, 72
- getJointDefaultAcceleration, 72
- getJointDefaultVelocity, 72
- getJointMaxAcceleration, 72
- getJointMaxPositions, 72
- getJointMaxVelocity, 72
- getJointMinPositions, 73
- getLocale, 73
- getLocaleForProgrammingLanguage, 73
- getMajorVersion, 73
- getMaxPayloadDistance, 73
- getMaxPayloadMass, 73
- getMinorVersion, 73
- getRobotDhParams, 74
- getSerialNumber, 74
- getStandardAnalogInputConfigure, 74
- getStandardAnalogOutputConfigure, 74
- getStandardDigitalInputConfigure, 74
- getStandardDigitalOutputConfigure, 74
- getStaticLinkInputConfigure, 75
- getStaticLinkOutputConfigure, 75
- getTcpDefaultAcceleration, 75
- getTcpDefaultVelocity, 75
- getTcpMaxAcceleration, 75
- getTcpMaxDistance, 75
- getTcpMaxVelocity, 75
- getToolAnalogInputConfigure, 76
- getToolAnalogOutputConfigure, 76
- getToolDigitalIOConfigure, 76
- getUnitType, 76
- isRealRobot, 76
- TargetRobot
 - arcs::aubo_scope::RenderInterface, 405
- TCP
 - arcs::aubo_scope::TCP, 467, 468
- TcpContributionModel
 - arcs::aubo_scope::TcpContributionModel, 470, 471
- TcpModel
 - arcs::aubo_scope::TcpModel, 474, 475
- TcpSelectionType
 - arcs::aubo_scope::MoveNode, 320
- TEACHING
 - arcs::aubo_scope::WaypointNode, 550
- ThreadNode
 - arcs::aubo_scope::ThreadNode, 478, 479
- TIGHTEN
 - arcs::aubo_scope, 105
- TIME
 - arcs::aubo_scope::WaitNode, 533
- TIME_MOTION
 - arcs::aubo_scope::WaypointNode, 549
- TimerNode
 - arcs::aubo_scope::TimerNode, 481, 482
- TOOL
 - arcs::aubo_scope::Io, 294
- TOOL_CONTACT
 - arcs::aubo_scope::UntilNode, 496
- TOOL_MOTION
 - arcs::aubo_scope::DirectionNode, 181
 - arcs::aubo_scope::WaypointNode, 549
- ToolPosition
 - arcs::aubo_scope::RenderInterface, 405
- toString
 - arcs::aubo_scope::Expression, 204
 - arcs::aubo_scope::Variable, 517
- Track
 - arcs::aubo_scope::RenderInterface, 405
- traverse
 - arcs::aubo_scope::TreeNode, 488
- TreeNode
 - arcs::aubo_scope::TreeNode, 483, 484
- Type
 - arcs::aubo_scope::PopupNode, 351
 - arcs::aubo_scope::Variable, 515
- type_
 - arcs::aubo_scope::EtherNetIPFilter, 201
 - arcs::aubo_scope::ModbusIoFilter, 317
 - arcs::aubo_scope::ProfinetFilter, 359
- types
 - arcs::aubo_scope::IoConfigure, 299
- UIntegerRangeValidator
 - arcs::aubo_scope::UIntegerRangeValidator, 490
- UNDEFINED
 - arcs::aubo_scope::GripperNode, 265
 - arcs::aubo_scope::WaypointNode, 549, 550
- UNDEFINED0
 - arcs::aubo_scope::GripperNode, 266
- UndoRedoManager
 - arcs::aubo_scope::UndoRedoManager, 491, 492
- UnitType
 - arcs::aubo_scope, 105
- Unknown
 - arcs::aubo_scope::UntilNode, 497
- UNRESOLVED_FEATURE
 - arcs::aubo_scope::DirectionNode, 181
 - arcs::aubo_scope::MoveNode, 320
- UNRESOLVED_OUTPUT
 - arcs::aubo_scope::SetNode, 447
- UntilNode
 - arcs::aubo_scope::UntilNode, 498
- updateCapability
 - arcs::aubo_scope::GripForceCapability, 253
 - arcs::aubo_scope::GripVacuumCapability, 268
 - arcs::aubo_scope::SpeedCapability, 455
 - arcs::aubo_scope::WidthCapability, 557
- updateFeature
 - arcs::aubo_scope::FeatureContributionModel, 210
- updatePayload
 - arcs::aubo_scope::PayloadContributionModel, 339, 340
- updateTcp
 - arcs::aubo_scope::TcpContributionModel, 472
- updateValue

- arcs::aubo_scope::DoubleRangeValidator, 186
- arcs::aubo_scope::IntegerRangeValidator, 290
- arcs::aubo_scope::StringLengthValidator, 457
- arcs::aubo_scope::UIntegerRangeValidator, 490
- USE_ACTIVE_TCP
 - arcs::aubo_scope::MoveNode, 320
- UserInteraction
 - arcs::aubo_scope::UserInteraction, 504, 505
- UserInterfaceApi
 - arcs::aubo_scope::UserInterfaceApi, 510, 511
- UserInterfaceApi(用户界面), 76
 - getUserInteraction, 77
- VALUE_PERSISTED
 - arcs::aubo_scope::Variable, 515
- ValueFactory
 - arcs::aubo_scope::ValueFactory, 512, 513
- VARIABLE
 - arcs::aubo_scope::WaypointNode, 550
- Variable
 - arcs::aubo_scope::Variable, 516
- VARIABLE_JOINT
 - arcs::aubo_scope::WaypointNode, 550
- VARIABLE_POSE
 - arcs::aubo_scope::WaypointNode, 550
- VARIABLE_POSITION
 - arcs::aubo_scope::WaypointNode, 549
- VariableDefinition
 - arcs::aubo_scope::WaypointNode, 550
- VariableFactory
 - arcs::aubo_scope::VariableFactory, 518, 519
- VariableModel
 - arcs::aubo_scope::VariableModel, 522, 523
- VariableType
 - arcs::aubo_scope::WaypointNode, 550
- View
 - arcs::aubo_scope::RenderInterface, 405
- ViewApiProvider
 - arcs::aubo_scope::ViewApiProvider, 526, 527
- VisualLocalizationProgramNode
 - arcs::aubo_scope::VisualLocalizationProgramNode, 529, 530
- WaitNode
 - arcs::aubo_scope::WaitNode, 533, 534
- WARNING
 - arcs::aubo_scope::PopupNode, 351
- Waypoint
 - arcs::aubo_scope::Waypoint, 540
- WAYPOINT_BLEND
 - arcs::aubo_scope::WaypointNode, 548
- WaypointModel
 - arcs::aubo_scope::WaypointModel, 543
- WaypointNode
 - arcs::aubo_scope::WaypointNode, 550, 551
- whileCondition
 - arcs::aubo_scope::ScriptWriter, 441
- whileNot
 - arcs::aubo_scope::ScriptWriter, 441
- whileTrue
 - arcs::aubo_scope::ScriptWriter, 441
- WidgetHandle
 - arcs::aubo_scope, 104
 - installation_node_view.h, 617
- writeChildren
 - arcs::aubo_scope::ScriptWriter, 441
- X_MINUS
 - arcs::aubo_scope::DirectionNode, 180
- X_PLUS
 - arcs::aubo_scope::DirectionNode, 180
- Y_MINUS
 - arcs::aubo_scope::DirectionNode, 180
- Y_PLUS
 - arcs::aubo_scope::DirectionNode, 180
- Z_MINUS
 - arcs::aubo_scope::DirectionNode, 180
- Z_PLUS
 - arcs::aubo_scope::DirectionNode, 180