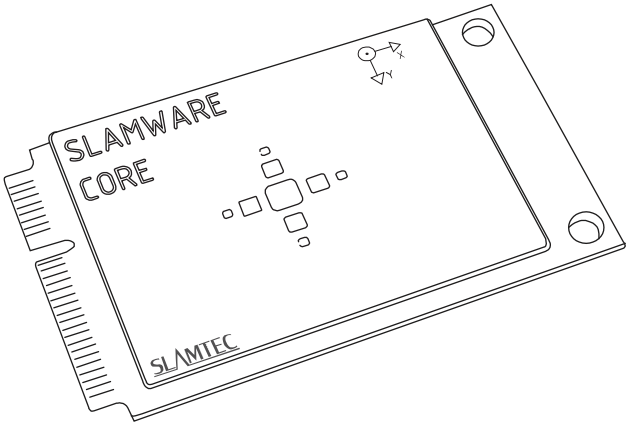
SLAMTEC

SLAMWARE

模块化自主定位导航解决方案

API 参考手册



2016-11-23 rev.1.8

www.slamtec.com

上海思岚科技有限公司

目录结构

Slamware SDK 包含了大量您开发过程中可能会用到的资源、代码、和项目文件，其目录结构组织如下：

目录	说明
bin	预编译的工具
include	SDK 相关的头文件
lib	预编译的库文件
samples	样例程序
workspaces	项目文件

图表 1-1 SLAMWARE SDK 目录结构

头文件结构

在 include 目录下，我们包含了 Slamware SDK 以及它所依赖的所有库的头文件：

目录	说明
boost	Boost 1.53.0
Eigen	Eigen 矩阵库
rpos	Slamware SDK 相关的头文件

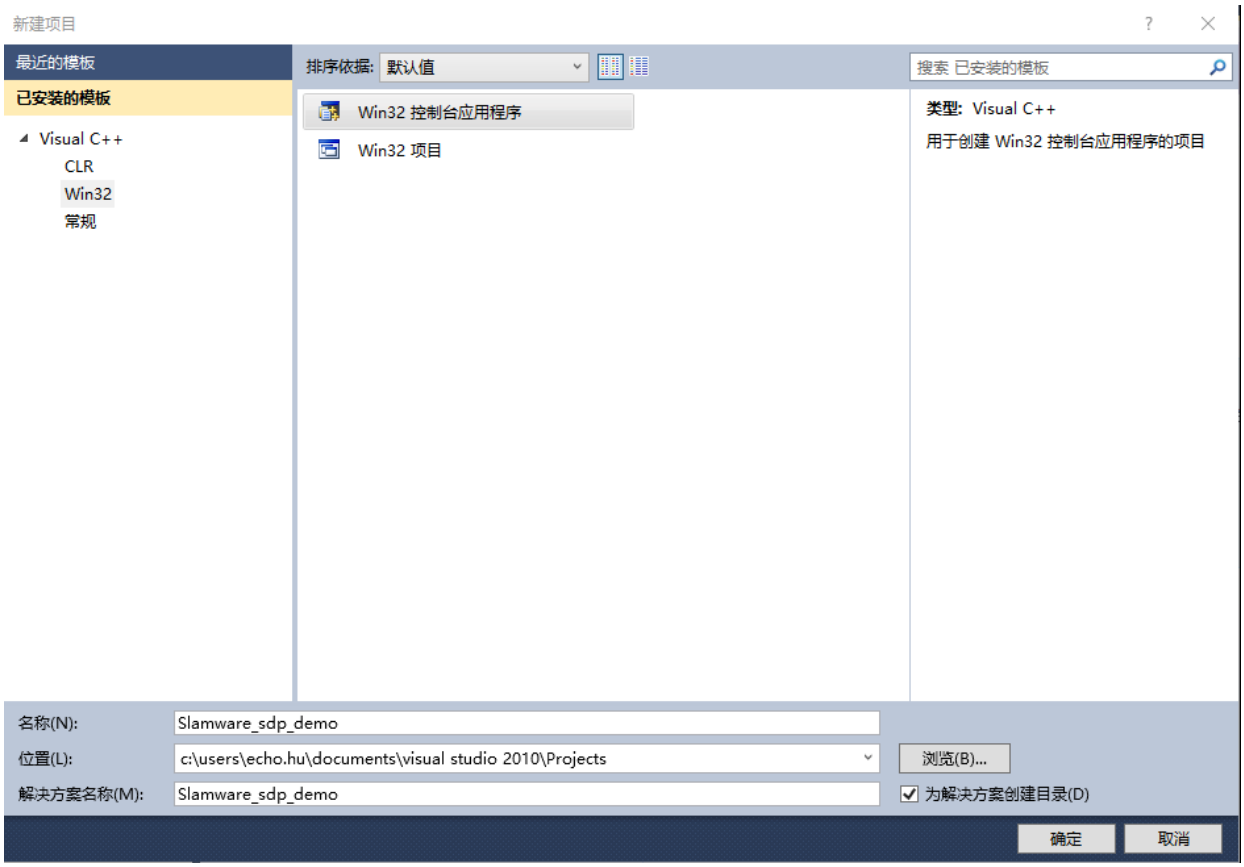
图表 1-2 SLAMWARE SDK 头文件组织结构

要基于 Slamware SDK 进行应用开发，需要您的开发环境满足如下条件：

- 您的计算机应当安装 Visual Studio 2010 SP1（由于我们提供的预编译库采用 Visual Studio 2010 SP1 进行编译，所以不可使用 Visual Studio 2012 或者 2013 进行开发）

新建工程

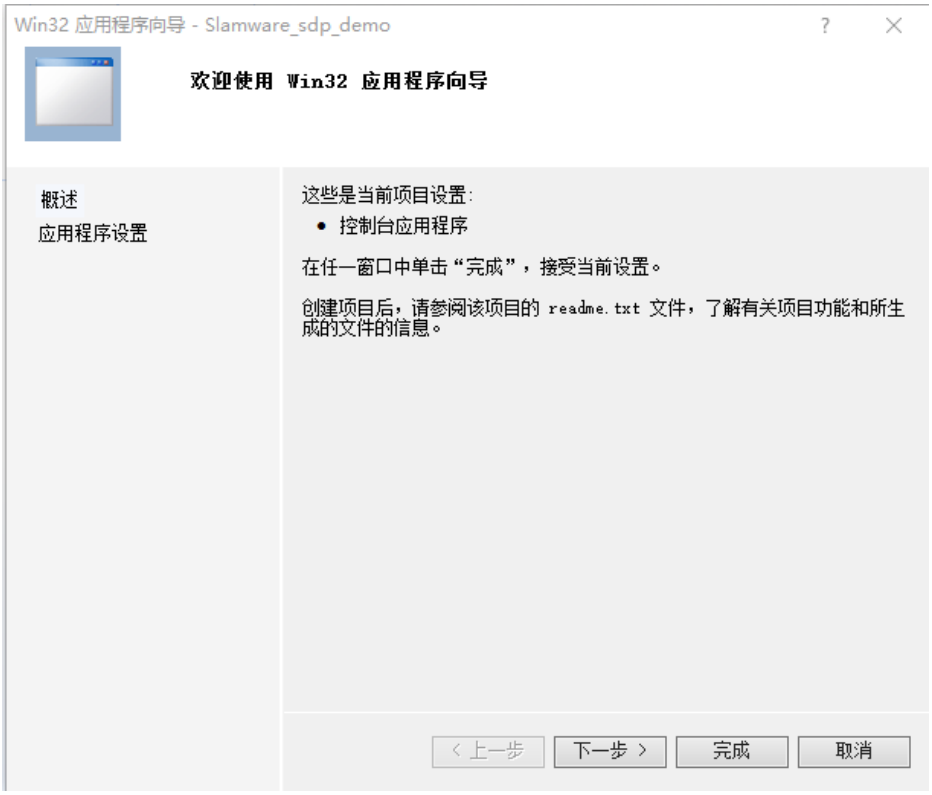
步骤 1 打开 Visual Studio 2010 并新建项目



图表 3-1 新建项目

1. 选择 Visual C++项目，并选择 Win32 Console Application（Win32 控制台应用程序）项目类型
2. 在 Name（名称）中输入项目名称
3. 点击 OK（确定）

步骤 2 设定应用程序选项



图表 3-2 向导概览

点击 Next（下一步）

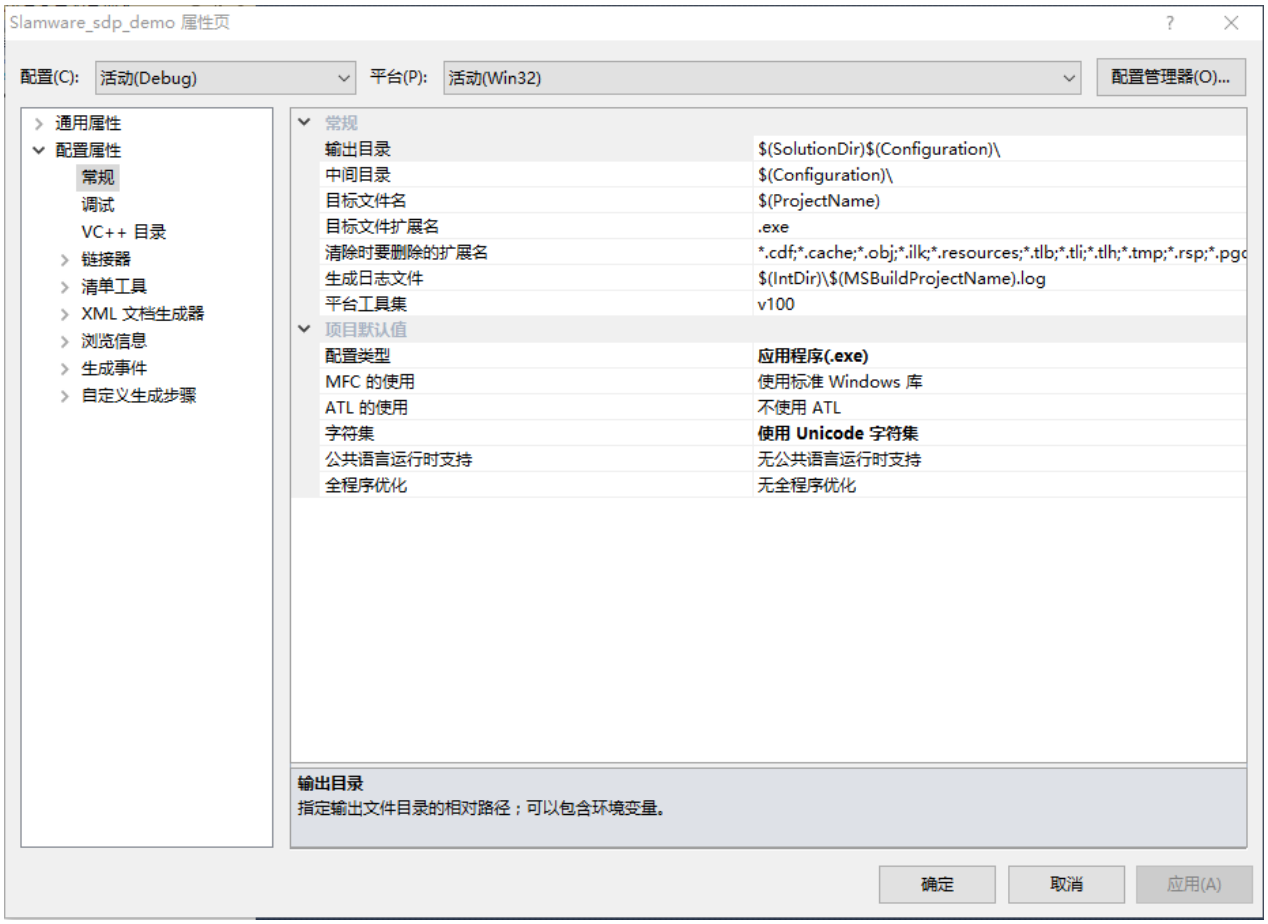
图表 3-3 项目设置

1. Application Type（应用程序类型）选择 Console Application（命令行应用程序）
2. Additional options（附加选项）勾选 Empty Project（空项目）
3. 点击 Finish 完成项目创建

配置编译选项

步骤 1 打开项目属性面板

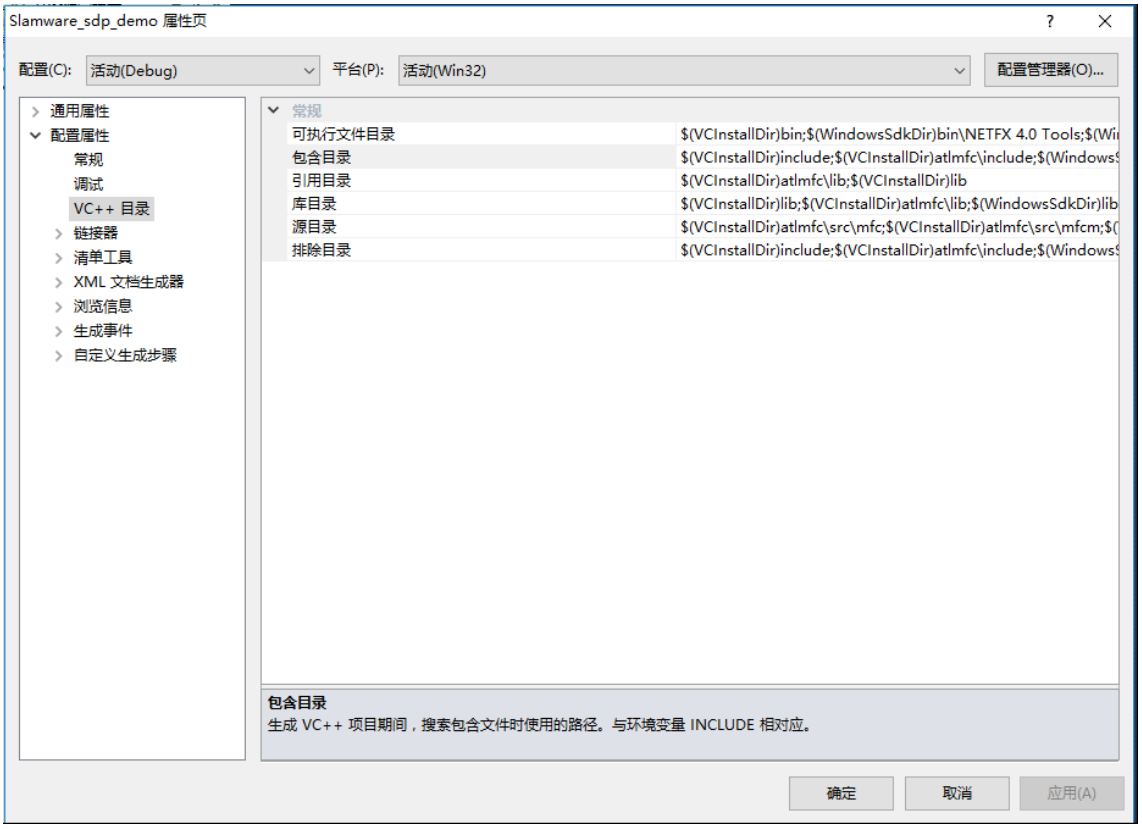
在 Solution Explorer（解决方案）中右键单击您刚刚创建的项目，并单击 Properties（属性）菜单，打开属性面板：



图表 3-4 项目属性页

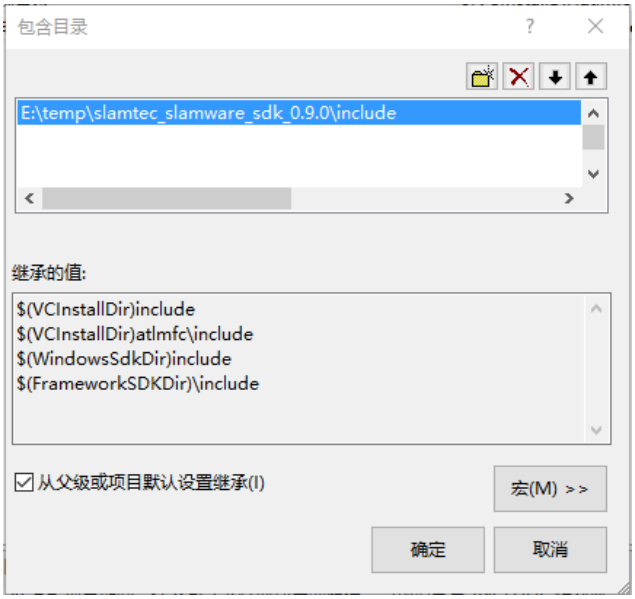
步骤 2 配置 VC++目录

在左侧列表中，选择 VC++ Directories（VC++目录）



图表 3-5 VC++目录

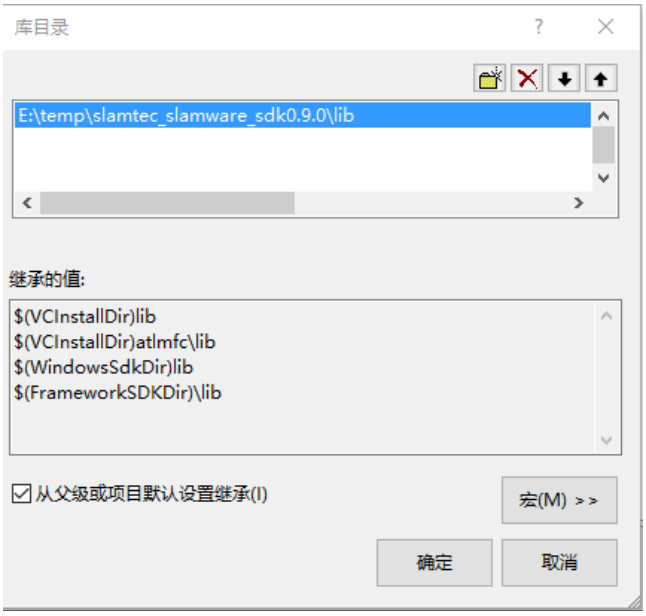
1. 选中右侧的 Include Directories（包含目录），并点击下拉按钮
2. 选择<Edit...>（<编辑...>）
3. 将第一种中提到的 SDK 中的 include 目录加入到列表中



图表 3-6 包含目录

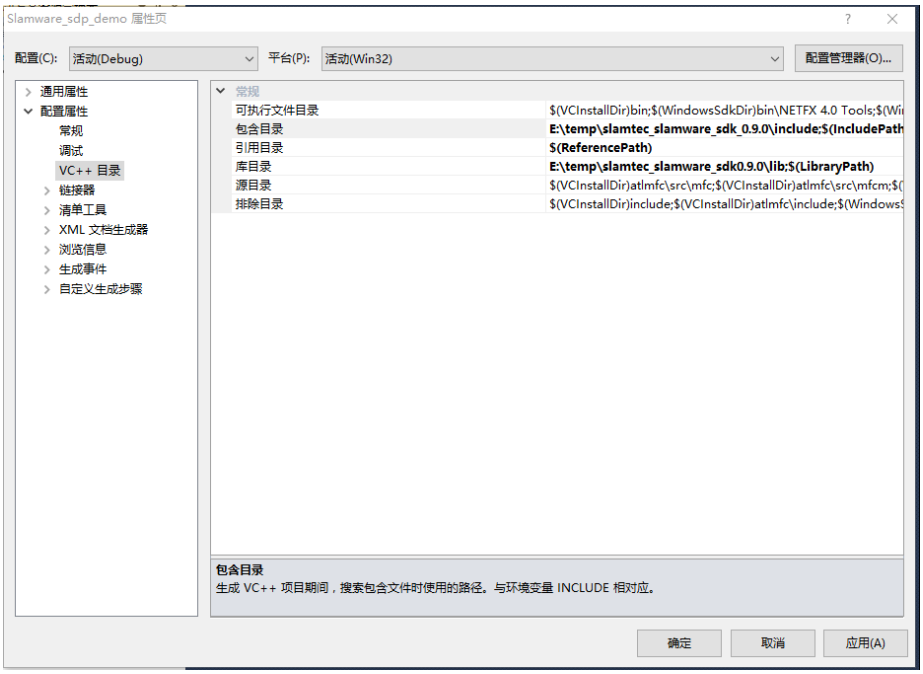
1. 选中右侧的 Library Directories（库目录），并点击下拉按钮
2. 选择<Edit...>（<编辑...>）

3. 将第一种中提到的 SDK 中的 lib 目录加入到列表中



图表 3-7 库目录

完成后，您的项目属性页应当与下图相似：



图表 3-8 完成配置的属性页

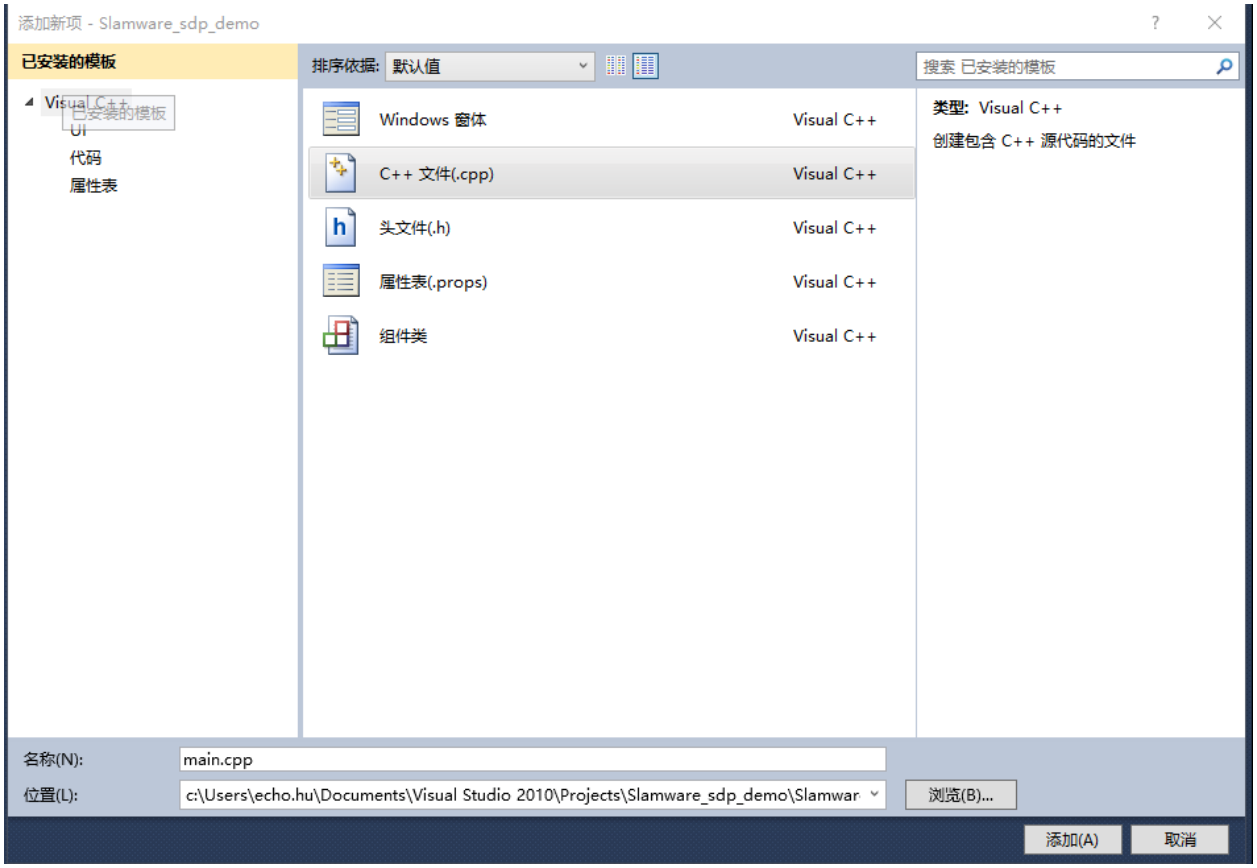
点击 OK（确定）完成配置。

Hello World

步骤 1 创建源文件

在 Solution Explorer 中您的项目下的 Source Files 目录上，右键单机，并在

菜单中选择 Add（添加）->New Item（新项目）



图表 3-9 新建源文件

选择 C++ File (.cpp)，并将文件命名成 main.cpp

步骤 2 添加代码

在弹出的编辑器中输入如下代码：

```
#include <rpos/robot_platforms/slamware_core_platform.h>
#include <iostream>
using namespace std;
using namespace rpos::robot_platforms;
int main(int argc, char* argv[])
{
	SlamwareCorePlatform platform =
SlamwareCorePlatform::connect("192.168.111.1", 1445);
	cout << "Base version: " << platform.getSDPVersion() << endl;
	return 0;
}
```

步骤 3 编译运行

在 Visual Studio 的主菜单中，单击 Debug（调试）->Start Debugging（开始调试）即可编译运行您的程序。

概览

对象	说明
rpos::core::Location 类	位置
rpos::core::Rotation 类	旋转姿态
rpos::core::Pose 类	姿态
rpos::core::Action 类	动作
rpos::core::ActionStatus 枚举	动作执行状态
rpos::core::Feature 类	特征类
rpos::core::RectangleF 类	矩形（float 型）
rpos::core::Vector2f 类	二维向量（float 型）
rpos::core::Vector2i 类	二维向量（int 型）
rpos::core::LaserPoint 类	激光雷达扫描点
rpos::core::RobotPlatform 类	机器人平台基类
rpos::robot::heading::HeadingMode 枚举	机器人移动时头朝向的方式
rpos::robot::heading::RobotHeading 类	机器人头朝向某个物体或者方向的设置
rpos::robot::option::MoveOption 结构体	存储机器人运行时的设置。
rpos::actions::MoveAction 类	移动动作
rpos::features::ArtifactProvider 类	器物功能特征，包括了虚拟墙功能。
rpos::features::LocationProvider 类	定位功能特征。
rpos::features::MotionPlanner 类	路径规划功能特征。
rpos::features::SweepMotionPlanner 类	清扫路径规划功能特征类。
rpos::features::SystemResource 类	系统资源功能特征类。
rpos::features::location_provider::Map 类	地图基类。
rpos::features::location_provider::MapType 枚举	MapType 枚举表示地图的类型。
rpos::features::location_provider::BitmapMap 类	位图地图。
rpos::features::location_provider::BitmapMapPixelFormat 枚举	BitmapMapPixelFormat 枚举表示位图地图的像素格式。
rpos::features::motion_planner::Path 类	代表一条路径。
rpos::features::system_resource::LaserScan 类	代表一次激光扫描数据。
rpos::robot_platforms::SlamwareCorePlatform 类	Slamware CORE 对象

图表 4-1 API 概览

rpos::core::Location 类

概览

Location 类用于表示三维空间中的一个坐标，由 x，y，z 三个成员组成，遵循右手坐标系。

头文件

rpos/core/pose.h

构造器

- Location()
- Location(double x, double y, double z)
- Location(const Location&)

运算符

- Location& operator=(const Location&)

方法

- double x() const
- double& x()
- double y() const
- double& y()
- double z() const
- double& z()

Location()构造器

创建一个 Location 对象，并自动将 x、y、z 都设为 0。

Location(double x, double y, double z)构造器

创建一个 Location 对象，将 x、y、z 设定为相应的值。

Location(const Location&)构造器

拷贝构造函数。

Location& operator=(const Location&)运算符

赋值运算符。

double x() const、double& x()属性

x 属性。

示例

```
Location location;
std::cout<<location.x()<<std::endl; // output 0
location.x() = 10;
std::cout<<location.x()<<std::endl; // output 10
```

double y() const、double& y()属性

y 属性，用法请参见 x 属性。

double z() const、double& z()属性

z 属性，用法请参见 x 属性。

rpos::core::Rotation 类

概览

Rotation 类用以表示物体在三维坐标系中的旋转姿态。Rotation 使用弧度作为角度单位。

头文件

```
rpos/core/pose.h
```

构造器

```
Rotation()
Rotation(double yaw, double pitch, double roll)
Rotation(const Rotation&)
```

运算符

```
Rotation& operator=(const Rotation&)
```

方法

```
double yaw() const
double& yaw()
double pitch() const
double& pitch()
double roll() const
double& roll()
```

Rotation()构造器

创建一个 Rotation 对象，并将 yaw，pitch，roll 都设定为 0。

Rotation(double yaw, double pitch, double roll)构造器

创建一个 Rotation 对象，并将 yaw，pitch，roll 都设定为相应值。

Rotation(const Rotation&)构造器

拷贝构造函数。

Rotation& operator=(const Rotation&)运算符

赋值运算符。

double yaw() const、double& yaw()属性

摇摆角（单位：弧度），遵循 Tait-Bryan angles 规则，请参见维基百科内容：

http://en.wikipedia.org/wiki/Euler_angles#Tait.E2.80.93Bryan_angles

用法请参见 Location::x()的示例。

double pitch() const、double& pitch()属性

俯仰角。

double roll() const, double& roll()属性

翻滚角。

rpos::core::Pose 类

概览

Pose 包含了 Location 和 Rotation 数据，即对象在三维空间中的完整姿态。

头文件

```
rpos/core/pose.h
```

构造器

```
Pose()
Pose(const Location&, const Rotation&)
Pose(const Location&)
Pose(const Rotation&)
Pose(const Pose&)
```

运算符

```
Pose& operator=(const Pose&)
```

方法

```
const Location& location() const
```

> Location& location()
> const Rotation& rotation() const
> Rotation& rotation()
> double x() const
> double& x()
> double y() const
> double& y()
> double z() const
> double& z()
> double yaw() const
> double& yaw()
> double pitch() const
> double& pitch()
> double roll() const
> double& roll()

Pose()构造器

构造一个 x、y、z、yaw、pitch、roll 都为 0 的 Pose 对象。

Pose(const Location&, const Rotation&)构造器

构造一个 location 和 rotation 为对应值的 Pose 对象。

Pose(const Location&)构造器

构造一个 location 为对应值，yaw、pitch、roll 皆为 0 的 Pose 对象。

Pose(const Rotation&)构造器

构造一个 rotation 为对应值，x、y、z 皆为 0 的 Pose 对象。

Pose(const Pose&)构造器

拷贝构造函数。

Pose& operator=(const Pose&)运算符

赋值运算符。

const Location& location() const、Location& location()属性

位置，详见 Location 类。

const Rotation& rotation() const、Rotation& rotation()属性

旋转，详见 Rotation 类。

double x() const、double& x()属性

x 属性。

示例

```
Location location;
std::cout<<location.x()<<std::endl; // output 0
location.x() = 10;
std::cout<<location.x()<<std::endl; // output 10
```

double y() const、double& y()属性

y 属性，用法请参见 x 属性。

double z() const、double& z()属性

z 属性，用法请参见 x 属性。

double yaw() const、double& yaw()属性

摇摆角（单位：弧度），遵循 Tait-Bryan angles 规则，请参见维基百科内容：

http://en.wikipedia.org/wiki/Euler_angles#Tait.E2.80.93Bryan_angles

用法请参见 Location::x()的示例。

double pitch() const、double& pitch()属性

俯仰角。

double roll() const, double& roll()属性

翻滚角。

rpos::core::Action 类

概览

Action 类表示一个动作。

头文件

```
rpos/core/action.h
```

构造器

```
Action(const Action&)
```

运算符

> Action& operator=(const Action&)

方法

> ActionStatus getStatus()
> void cancel()
> ActionStatus waitUntilDone()
> template<class ActionT> ActionT cast()

Action(const Action&)构造器

拷贝构造函数。

Action& operator=(const Action&)运算符

赋值运算符。

ActionStatus getStatus()

获得当前动作状态，返回值详见 ActionStatus 枚举。

void cancel()

取消当前动作。

ActionStatus waitUntilDone()

等待动作完成或出错，返回值为动作结束时的动作状态，详见 ActionStatus 枚举。

template<class ActionT> ActionT cast()

将 rpos::core::Action 的对象转换成子类的对象

示例

```
rpos::core::Action                    someAction                    =
robotPlatform.startSomeAction();
rpos::actions::MoveAction                moveAction                 =
someAction.cast<rpos::actions::MoveAction>;
```

rpos::core::ActionStatus 枚举

概览

ActionStatus 枚举表示动作的状态。

头文件

rpos/core/action.h

枚举项

- ActionStatusWaitingForStart
- ActionStatusRunning
- ActionStatusFinished
- ActionStatusPaused
- ActionStatusStopped
- ActionStatusError

ActionStatusWaitingForStart

动作正在等待开始

ActionStatusRunning

动作正在进行

ActionStatusFinished

动作已经完成

ActionStatusPaused

动作已经暂停

ActionStatusStopped

动作已经停止（取消）

ActionStatusError

动作执行过程中出现错误

rpos::core::Feature 类

概览

Feature 类表示一个特征，即一个特定的功能集合。

头文件

rpos/core/feature.h

构造器

- Feature(const Feature&)

运算符

> Feature& operator=(const Feature&)

Feature(const Feature&)构造器

拷贝构造函数

Feature& operator=(const Feature&)运算符

赋值运算符

rpos::core::RectangleF 类

概览

RectangleF 类表示一个矩形，其坐标参数的类型为 float。

头文件

rpos/core/geometory.h

构造器

> RectangleF()
> RectangleF(Vector2f position, Vector2f size)
> RectangleF(float x, float y, float width, float height)
> RectangleF(const RectangleF&)

运算符

> RectangleF& operator=(const RectangleF&)

方法

> const Vector2f& position()
> Vector2f& position()
> const Vector2f& size()
> Vector2f& size()
> float x() const
> float& x()
> float y() const
> float& y()
> float width() const
> float& width()
> float height() const
> float& height()

> float left() const
> float right() const
> float top() const
> float bottom() const
> bool contains(const Vector2i& point)
> bool empty()
> bool contains(const RectangleF& dest)
> void unionOf(const RectangleF& dest)

RectangleF()构造器

创建一个 x、y、width、height 都为 0 的矩形。

RectangleF(Vector2f position, Vector2f size)构造器

创建一个位置和大小为指定值的矩形。

RectangleF(float x, float y, float width, float height)构造器

创建一个位置和大小为指定值的矩形。

RectangleF(const RectangleF&)构造器

拷贝构造函数

RectangleF& operator=(const RectangleF&)运算符

赋值运算符

const Vector2f& position()、Vector2f& position()属性

矩形的位置（左上角）

const Vector2f& size()、Vector2f& size()属性

矩形的大小

float x() const、float& x()属性

矩形左上角的 x 坐标

float y() const、float& y()属性

矩形左上角的 y 坐标

float width() const、float& width()属性

矩形的宽度

float height() const、float& height()属性

矩形的高度

float left() const 属性

矩形左侧的 x 坐标

float right() const 属性

矩形右侧的 x 坐标（right=x+width）

float top() const 属性

矩形顶部的 y 坐标

float bottom() const 属性

矩形底部的 y 坐标（bottom=y+height）

bool contains(const Vector2i& point)

判断点是否在矩形的范围内

bool empty()

判断矩形是否是全空的（即 width() < epsilon 或 height()<epsilon）

bool contains(const RectangleF& dest)

判断目标矩形是否完全在本矩形的区域内

void unionOf(const RectangleF& dest)

计算本矩形和目标矩形重合的部分，并将本矩形设定为该重合部分矩形。

rpos::core::Vector2f 类

概览

二维向量，元素数据类型为 float 型。

头文件

`rpos/core/geometry.h`

构造器

> `Vector2f()`
> `Vector2f(float x, float y)`
> `Vector2f(const Vector2f&)`

运算符

> Vector2f& operator=(const Vector2f&)

方法

> float x() const
> float& x()
> float y() const
> float& y()

Vector2f()构造器

构造一个新的向量，其 x、y 的值是不确定的。

Vector2f(float x, float y)构造器

构造一个 x、y 为指定的值的向量。

Vector2f(const Vector2f&)构造器

拷贝构造函数。

Vector2f& operator=(const Vector2f&)运算符

赋值运算符。

float x() const、float& x()属性

二维向量的 x 分量。

float y() const、float& y()属性

二维向量的 y 分量。

rpos::core::Vector2i 类

概览

二维向量，元素数据类型为 int 型。

头文件

`rpos/core/geometry.h`

构造器

```
Vector2i()
Vector2i(float x, float y)
Vector2i(const Vector2i&)
```

运算符

```
Vector2i& operator=(const Vector2i&)
```

方法

```
int x() const
int& x()
int y() const
int& y()
```

与 rpos::core::Vector2f 类类似，不再赘述。

rpos::core::LaserPoint 类

概览

激光雷达测距的单点数据，包括了距离、角度、是否有效等信息。

头文件

rpos/core/laser_point.h

构造器

```
LaserPoint()
LaserPoint(float distance, float angle, bool valid)
LaserPoint(const LaserPoint&)
```

运算符

```
LaserPoint& operator=(const LaserPoint&)
```

方法

```
float distance() const
float& distance()
float angle() const
float& angle()
bool valid() const
bool& valid()
```

LaserPoint()构造器

创建一个新的 LaserPoint 对象。

LaserPoint(float distance, float angle, bool valid)构造器

创建一个距离、角度、有效性为指定值的 LaserPoint 对象。

LaserPoint(const LaserPoint&)构造器

拷贝构造函数

LaserPoint& operator=(const LaserPoint&)运算符

赋值运算符

float distance() const、float& distance()属性

距离数据（单位：米）

float angle() const、float& angle()属性

本次测量的角度（单位：弧度）

bool valid() const、bool& valid()属性

本次测量是否有效

rpos::core::RobotPlatform 类

概览

机器人平台是一系列设备组合而成的整体，提供一系列的特征从而提供功能。RobotPlatform 类是所有机器人平台的基类。

头文件

`rpos/core/robot_platform.h`

构造器

```
RobotPlatform(const RobotPlatform&)
```

运算符

```
RobotPlatform& operator=(const RobotPlatform&)
```

方法

```
std::vector<Feature> getFeatures()
template<class RobotPlatformT> RobotPlatformT cast()
```

RobotPlatform(const RobotPlatform&)构造器

拷贝构造函数。

RobotPlatform& operator=(const RobotPlatform&)运算符

赋值运算符。

std::vector<Feature> getFeatures()

获得该机器人平台提供的所有特征。

template<class RobotPlatformT> RobotPlatformT cast()

将 RobotPlatform 对象转换成子类对象，示例请参考 `rpos::core::Action::cast<>`。

rpos::robot::heading::HeadingMode 枚举

概览

HeadingMode 枚举表示机器人移动时头朝向的方式

头文件

`rpos/features/motion_planner/move_heading.h`

枚举项

> HeadingModeAuto,
> HeadingModeFixAngle,
> HeadingModeCircleMotion,
> HeadingModeDirection

HeadingModeAuto

机器人按照自己的方式随意行走。

HeadingModeFixAngle

机器人行走时头与前进方向成固定角度。

HeadingModeCircleMotion

机器人行走时头始终朝向某个物体或者某点。

HeadingModeDirection

机器人行走时头始终朝向某个固定的方向。

rpos::robot::heading::RobotHeading 类

概览

RobotHeading 类表示机器人在行走的时候的头朝向某个物体或者方向的设置

头文件

rpos/features/motion_planner/move_heading.h

构造器

RobotHeading()

RobotHeading(HeadingMode headingMode, rpos::core::Pose pose)

运算符

RobotHeading& operator=(const RobotHeading&)

方法

const rpos::core::Pose& pose() const

const HeadingMode& headingMode() const

RobotHeading()构造器

构造函数。

RobotHeading(HeadingMode headingMode, rpos::core::Pose pose)构造器

构造函数。

RobotHeading& operator=(const RobotHeading&)运算符

赋值运算符。

const rpos::core::Pose& RobotHeading::pose() const

获取机器人行走时头朝向的角度或者物体的位置

头朝向与位置或者角度的对应关系。

朝向	值
HeadingModeAuto	Pose 值不可用
HeadingModeFixAngle 或者 HeadingModeDirection	Pose 的 Rotation 参数可用
HeadingModeCircleMotion	Pose 的 Location 参数可用

const HeadingMode& RobotHeading::headingMode() const

获取机器人行走时头朝向设置。

可以参考 rpos::robot::heading::HeadingMode。

rpos::robot::option::MoveOption 结构体

概览

MoveOption 结构体存储机器人运行时的设置。

头文件

`rpos/features/motion_planner/move_option.h`

结构体说明

> appending
> isMilestone
> robotHeading

appending

如果机器人正在执行其他的移动动作，该参数决定新的点是追加或是替换既有节点。

isMilestone

机器人是否通过路径搜索的方式前往目的地。

robotHeading

机器人行走时头朝向设置。

参考 rpos::robot::heading::RobotHeading。

rpos::actions::MoveAction 类

概览

MoveAction 类表示一个移动的动作，它包含了当前机器人规划的路径、检查点列表、移动的进程。

头文件

`rpos/features/motion_planner/move_action.h`

父类

继承自 rpos::core::Action 类

构造器

> MoveAction(boost::shared_ptr<rpos::actions::detail::MoveActionImpl>)
> MoveAction(const MoveAction&)

运算符

MoveAction& operator=(const MoveAction&)

方法

rpos::features::motion_planner::Path getRemainingPath()

rpos::features::motion_planner::Path getRemainingMilestones()

继承自 rpos::core::Action 类的方法

ActionStatus getStatus()

void cancel()

ActionStatus waitUntilDone()

template<class ActionT> ActionT cast()

MoveAction(boost::shared_ptr<rpos::actions::detail::MoveActionImpl>) 构造器

该构造器仅限 SDK 内部使用。

MoveAction(const MoveAction&)构造器

拷贝构造函数。

MoveAction& operator=(const MoveAction&)运算符

赋值运算符。

rpos::features::motion_planner::Path getRemainingPath()

获得已经规划好的，剩余的路径。

rpos::features::motion_planner::Path getRemainingMilestones()

获得剩余的里程碑。

rpos::features::ArtifactProvider 类

概览

器物功能特征，包括了虚拟墙功能。

头文件

rpos/features/artifact_provider.h

父类

继承自 rpos::core::Feature 类

构造器

- ArtifactProvider(boost::shared_ptr<detail::ArtifactProviderImpl>)
- ArtifactProvider(const ArtifactProvider&)

运算符

- ArtifactProvider& operator=(const ArtifactProvider&)

方法

- std::vector<rpos::core::Line> getWalls()
- bool addWall(const rpos::core::Line&)
- bool addWalls(const std::vector<rpos::core::Line>&)
- bool clearWallById(const rpos::core::SegmentID&)
- bool clearWalls()

ArtifactProvider(boost::shared_ptr<detail::ArtifactProviderImpl>)构造器

该构造器仅限 SDK 内部使用。

ArtifactProvider(const ArtifactProvider&)构造器

拷贝构造函数。

ArtifactProvider& operator=(const ArtifactProvider&)运算符

赋值运算符。

std::vector<rpos::core::Line> getWalls()

获取系统中所有的虚拟墙。

bool addWall(const rpos::core::Line&)

添加虚拟墙。

bool addWalls(const std::vector<rpos::core::Line>&)

添加虚拟墙。

bool clearWallById(const rpos::core::SegmentID&)

清除指定的虚拟墙。

bool clearWalls()

清除所有的虚拟墙。

rpos::features::LocationProvider 类

概览

定位功能特征，包括了自动建图和定位的功能（亦即 SLAM 功能）。

头文件

rpos/features/location_provider.h

父类

继承自 rpos::core::Feature 类

构造器

LocationProvider(boost::shared_ptr<detail::LocationProviderImpl>)
LocationProvider(const LocationProvider&)

运算符

LocationProvider& operator=(const LocationProvider&)

方法

std::vector<rpos::features::location_provider::MapType> getAvailableMaps()
rpos::features::location_provider::Map getMap(
  rpos::features::location_provider::MapType,
  rpos::core::RectangleF,
  rpos::features::location_provider::MapKind)
bool setMap(
  const rpos::features::location_provider::Map&,
  rpos::features::location_provider::MapType,
  rpos::features::location_provider::MapKind)
rpos::core::RectangleF getKnownArea(
  rpos::features::location_provider::MapType,
  rpos::features::location_provider::MapKind)
bool clearMap()
rpos::core::Location getLocation()
rpos::core::Pose getPose()
bool setPose(const rpos::core::Pose&)
bool getMapLocalization()

> bool setMapLocalization(bool)
> bool getMapUpdate()
> bool setMapUpdate(bool)

LocationProvider(boost::shared_ptr<detail::LocationProviderImpl>)构造器

该构造器仅限 SDK 内部使用。

LocationProvider(const LocationProvider&)构造器

拷贝构造函数。

LocationProvider& operator=(const LocationProvider&)运算符

赋值运算符。

std::vector< rpos::features::location_provider::MapType> getAvailableMaps()

获得该定位功能特征提供的所有地图类型。

rpos::features::location_provider::Map getMap(
rpos::features::location_provider::MapType,
rpos::core::RectangleF,
rpos::features::location_provider::MapKind)

获得该定位功能特征提供的指定地图类型指定区域的地图数据。

bool setMap(
const rpos::features::location_provider::Map&,
rpos::features::location_provider::MapType,
rpos::features::location_provider::MapKind)

上载指定地图类型指定区域的地图数据到该定位功能特征，返回是否成功。

rpos::core::RectangleF getKnownArea(
rpos::features::location_provider::MapType,
rpos::features::location_provider::MapKind)

获得指定地图类型的地图中，已经完成建图的区域。

bool clearMap()

清除地图数据。

rpos::core::Location getLocation()

获得机器人在上述地图坐标系统中的坐标。

rpos::core::Pose getPose()

获得机器人在上述地图坐标系统中的姿态。

bool setPose(const rpos::core::Pose&)

上载当前机器人的姿态到上述地图坐标系统中，返回是否成功。

bool getMapLocalization()

获得机器人是否启用定位功能。

bool setMapLocalization(bool)

设置机器人是否启用定位功能。

bool getMapUpdate()

获取机器人是否启用地图更新功能。

bool setMapUpdate(bool)

设置机器人是否启用地图更新功能。

rpos::features::MotionPlanner 类

概览

路径规划功能特征，包括了动态路径规划功能和自动壁障功能。

头文件

`rpos/features/motion_planner.h`

父类

继承自 `rpos::core::Feature 类`

构造器

> `MotionPlanner(boost::shared_ptr<detail::MotionPlannerImpl>)`
> `MotionPlanner(const MotionPlanner&)`

运算符

> `MotionPlanner& operator=(const MotionPlanner&)`

方法

> `rpos::actions::MoveAction moveTo(`

> const std::vector<rpos::core::Location>&,
> bool, bool)
> rpos::actions::MoveAction moveTo(const rpos::core::Location&, bool, bool)
> rpos::actions::MoveAction getCurrentAction()
> rpos::features::motion_planner::Path searchPath(const rpos::core::Location&)

MotionPlanner(boost::shared_ptr<detail::MotionPlannerImpl>)构造器

该构造器仅限 SDK 内部使用。

MotionPlanner(const MotionPlanner&)构造器

拷贝构造函数。

MotionPlanner& operator=(const MotionPlanner&)运算符

赋值运算符。

rpos::actions::MoveAction moveTo(const std::vector<rpos::core::Location>&, bool, bool)

让机器人沿着路径移动（机器人会逐一走过路径中的节点，在节点之间尽量走圆滑的曲线，如果遇到障碍物，机器人会自动避开）。

参数

名称	类型	说明
locations	const std::vector <rpos::core::Location>&	期望机器人经过的点
appending	bool	如果机器人正在执行其他的移动动作，该参数决定新的点是追加或是替换既有节点
isMilestone	bool	当这个参数为 true 时，机器人会将上述点视作关键点，通过路径搜索的方式前往目的地；当参数为 false 时，会被视作普通点，不会启用路径搜索功能。

rpos::actions::MoveAction moveTo(const rpos::core::Location&, bool, bool)

让机器人移动到目标位置。

参数

名称	类型	说明
location	const rpos::core::Location&	期望机器人经过的点
appending	bool	如果机器人正在执行其他的移动动作，该参数决定新的点是追加或是替换既有节点
isMilestone	bool	当这个参数为true时，机器人会将上述点视作关键点，通过路径搜索的方式前往目的地；当参数为 false 时，会被视作普通点，不会启用路径搜索功能。

rpos::actions::MoveAction getCurrentAction()

获得机器人当前正在进行的移动动作。

您可以使用 Action::isEmpty()方法判断它是否存在。当机器人当前没有正在进行的动作时，它 Action::isEmpty()将会返回 true。

rpos::features::motion_planner::Path searchPath(const rpos::core::Location&)

使用机器人内置的寻路算法寻找前往指定目的地的路径。

rpos::features::SweepMotionPlanner 类

概览

清扫路径规划功能特征类。该类中的功能主要针对扫地机专属版本 Slamware Core 所提供的清扫和自动回充功能。

头文件

`rpos/features/sweep_motion_planner.h`

父类

继承自 `rpos::core::Feature` 类

构造器

`SweepMotionPlanner(boost::shared_ptr<detail::SweepMotionPlan`

nerImpl>)
SweepMotionPlanner(const SweepMotionPlanner&)

运算符

SweepMotionPlanner& operator=(const SweepMotionPlanner&)

方法

rpos::actions::SweepMoveAction startSweep()
rpos::actions::SweepMoveAction sweepSpot(const rpos::core::Location& location)
rpos::actions::MoveAction goHome()

SweepMotionPlanner(boost::shared_ptr<detail::SweepMotionPlannerImpl>)构造器

该构造器仅限 SDK 内部使用。

SweepMotionPlanner(const SweepMotionPlanner&)构造器

拷贝构造函数。

SweepMotionPlanner& operator=(const SweepMotionPlanner&)运算符

赋值运算符。

rpos::actions::SweepMoveAction startSweep()

让机器人进行清扫。（此接口仅适用于扫地机版本）

rpos::actions::SweepMoveAction sweepSpot(const rpos::core::Location& location)

让机器人定点打扫。（此接口仅适用于扫地机版本）

rpos::actions::MoveAction goHome()

让机器人回家充电。

rpos::features::system_resource::DeviceInfo 类

概览

获取设备信息。设备信息包括设备 ID，制造商，型号，硬件版本，软件版本。

头文件

rpos/features/device_info.h

构造器

DeviceInfo()
DeviceInfo(const DeviceInfo&)

运算符

DeviceInfo& operator=(const DeviceInfo&)

方法

std::string deviceID() const、std::string& deviceID();
int manufacturerID() const、int& manufacturerID();
std::string manufacturerName() const 、std::string& manufacturerName();
int modelID() const、int& modelID();
std::string modelName() const、std::string& modelName();
std::string hardwareVersion() const 、std::string& hardwareVersion();
std::string softwareVersion() const 、std::string& softwareVersion();

DeviceInfo()

构造函数。

DeviceInfo(const DeviceInfo&)

构造一个特定设备信息的函数。

DeviceInfo& operator=(const DeviceInfo&)

赋值运算符。

std::string deviceID() const、std::string& deviceID();

deviceID 属性。

int manufacturerID() const、int& manufacturerID();

manufacturerID 属性。

std::string manufacturerName() const、std::string& manufacturerName();

manufacturerName 属性。

int modelID() const、int& modelID();

modelID 属性。

std::string modelName() const、std::string& modelName();

modelName 属性。

std::string hardwareVersion() const、std::string& hardwareVersion();

硬件版本属性。

std::string softwareVersion() const、std::string& softwareVersion();

软件版本属性。

rpos::features::SystemResource 类

概览

系统资源功能特征类。该类提供了对原始激光扫描数据、电源管理系统等相关资源的访问 API。

头文件

rpos/features/system_resource.h

父类

继承自 rpos::core::Feature 类

构造器

SystemResource(boost::shared_ptr<detail::SystemResourceImpl>)

SystemResource(const SystemResource&)

运算符

SystemResource& operator=(const SystemResource&)

方法

int getBatteryPercentage()

bool getBatteryIsCharging()

bool getDCIsConnected()

int getBoardTemperature()

std::string getSDPVersion()

rpos::features::system_resource::LaserScan getLaserScan()

rpos::features::system_resource::DeviceInfo getDeviceInfo()

SystemResource(boost::shared_ptr<detail::SystemResourceImpl>)构造器

该构造器仅限 SDK 内部使用。

SystemResource(const SystemResource&)构造器

拷贝构造函数。

SystemResource& operator=(const SystemResource&)运算符

赋值运算符。

int getBatteryPercentage()

获得电池电量，返回值的单位是百分比。比如电池剩余为 56%，则该返回值为 56。

bool getBatteryIsCharging()

电池是否处于充电状态。

bool getDCIsConnected()

外部电源是否连接。

int getBoardTemperature()

系统温度。

std::string getSDPVersion()

底盘的版本号。

rpos::features::system_resource::LaserScan getLaserScan()

获取原始激光扫描数据。

rpos::features::location_provider::Map 类

概览

地图基类，泛指定位功能获得的地图。

头文件

`rpos/features/location_provider.h`

构造器

> `Map(boost::shared_ptr<detail::MapImpl>)`
> `Map(const Map&)`

运算符

> Map& operator=(const Map&)

方法

> rpos::core::RectangleF getMapArea()
> rpos::core::Vector2f getMapPosition()
> rpos::core::Vector2i getMapDimension()
> rpos::core::Vector2f getMapResolution()
> rpos::system::types::timestamp_t getMapTimestamp()
> void setMapData(float, float, int, int, float, const std::vector< u8>&, rpos::system::types:: u64)
> std::vector< u8>& getMapData()
> template<class MapT> MapT cast()

Map(boost::shared_ptr<detail::MapImpl>)构造器

该构造器仅限 SDK 内部使用。

Map(const Map&)构造器

拷贝构造函数。

Map& operator=(const Map&)运算符

赋值运算符。

rpos::core::RectangleF getMapArea()

获得这张地图所包含的区域。

rpos::core::Vector2f getMapPosition()

获得这张地图左上角的坐标。

rpos::core::Vector2i getMapDimension()

获得地图的尺寸（两个维度的像素个数）

rpos::core::Vector2f getMapResolution()

获得地图的分辨率（在各个维度上，每个像素代表多少米）

rpos::system::types::timestamp_t getMapTimestamp()

获得地图生成的时间。

void setMapData(float, float, int, int, float, const std::vector<_u8>&, rpos::system::types::_u64)

设置地图数据。

std::vector<_u8>& getMapData()

获得地图数据。

template<class MapT> MapT cast()

将地图转换成特定类型的子类对象。

rpos::features::location_provider::MapType 枚举

概览

MapType 枚举表示地图的类型。

头文件

`rpos/features/location_provider.h`

枚举项

MapTypeBitmap8Bit

MapTypeBitmap8Bit

每像素 8 位的位图。

rpos::features::location_provider::BitmapMap 类

概览

位图地图。

头文件

`rpos/features/location_provider.h`

父类

继承自 rpos::features::location_provider::Map 类

构造器

BitmapMap(boost::shared_ptr<detail::BitmapMapImpl>)
BitmapMap(const BitmapMap&)

运算符

BitmapMap& operator=(const BitmapMap&)

方法

> BitmapMapPixelFormat getMapFormat()

继承自 rpos::features::location_provider::Map 类的方法

> rpos::core::RectangleF getMapArea()
> rpos::core::Vector2f getMapPosition()
> rpos::core::Vector2i getMapDimension()
> rpos::core::Vector2f getMapResolution()
> rpos::system::types::timestamp_t getMapTimestamp()
> void setMapData(float, float, int, int, float, const std::vector<_u8>&, rpos::system::types::_u64)
> std::vector<_u8>& getMapData()
> template<class MapT> MapT cast()

BitmapMap(boost::shared_ptr<detail::BitmapMapImpl>)构造器

该构造器仅限 SDK 内部使用。

BitmapMap(const BitmapMap&)构造器

拷贝构造函数。

BitmapMap& operator=(const BitmapMap&)运算符

赋值运算符。

BitmapMapPixelFormat getMapFormat()

获得地图的像素格式。

rpos::features::location_provider::BitmapMapPixelFormat 枚举

概览

BitmapMapPixelFormat 枚举表示位图地图的像素格式。

头文件

`rpos/features/location_provider.h`

枚举项

> BitmapMapPixelFormat8Bit

BitmapMapPixelFormat8Bit

位图中的每个像素占用 1 个字节。

rpos::features::motion_planner::Path 类

概览

Path 对象是一系列 Location 对象的集合，代表一条路径。

头文件

`rpos/features/motion_planner.h`

构造器

> `Path(const std::vector<rpos::core::Location>&)`
> `Path(const Path&)`

运算符

> `Path& operator=(const Path&)`

方法

> `std::vector<rpos::core::Location>& getPoints()`

Path(const std::vector<rpos::core::Location>&)构造器

创建一个由一系列点组成的路径。

Path(const Path&)构造器

拷贝构造函数。

Path& operator=(const Path&)运算符

赋值运算符。

std::vector<rpos::core::Location>& getPoints()

获得路径中的所有点。

rpos::features::system_resource::LaserScan 类

概览

LaserScan 对象是一系列 LaserPoint 对象的集合，代表一次激光扫描的数据。

头文件

`rpos/features/system_resource.h`

构造器

LaserScan(const std::vector<rpos::core::LaserPoint>&)
LaserScan(const LaserScan&)

运算符

LaserScan& operator=(const LaserScan&)

方法

std::vector<rpos::core::LaserPoint>& getLaserPoints()

LaserScan(const std::vector<rpos::core::LaserPoint>&)构造器

创建一个由一系列激光扫描点组成的扫描数据。

LaserScan(const LaserScan&)构造器

拷贝构造函数。

LaserScan& operator=(const LaserScan&)运算符

赋值运算符。

std::vector<rpos::core::LaserPoint>& getLaserPoints()

获取激光扫描数据。

rpos::robot_platforms::SlamwareCorePlatform 类

概览

SlamwareCorePlatform 对象代表一个基于 Slamware 的机器人，用以获取设备的状态、控制设备的行为。

头文件

`rpos/robot_platforms/slamware_core_platform.h`

父类

继承自 `rpos::core::RobotPlatform` 类

构造器

SlamwareCorePlatform(boost::shared_ptr<detail::SlamwareCorePlatformImpl>)
SlamwareCorePlatform(const SlamwareCorePlatform&)

运算符

SlamwareCorePlatform& operator=(const SlamwareCorePlatform&)

静态方法

SlamwareCorePlatform connect(const std::string&, int, int)

方法

void disconnect()
std::vector<rpos::core::Line> getWalls()
bool addWall(const rpos::core::Line&)
bool addWalls(const std::vector<rpos::core::Line>&)
bool clearWallById(const rpos::core::SegmentID&)
bool clearWalls()
std::vector<rpos::features::location_provider::MapType> getAvailableMaps()
rpos::features::location_provider::Map getMap(
  rpos::features::location_provider::MapType,
  rpos::core::RectangleF,
  rpos::features::location_provider::MapKind)
bool setMap(
  const rpos::features::location_provider::Map&,
  rpos::features::location_provider::MapType,
  rpos::features::location_provider::MapKind , bool partially=false)
bool setMap(const core::Pose&,
  const rpos::features::location_provider::Map&,
  rpos::features::location_provider::MapType,
  rpos::features::location_provider::MapKind, bool partially = false)
rpos::core::RectangleF getKnownArea(
  rpos::features::location_provider::MapType,
  rpos::features::location_provider::MapKind)
bool clearMap()
bool clearMap(rpos::features::location_provider::MapKind kind)
rpos::core::Location getLocation()
rpos::core::Pose getPose()
bool setPose(const rpos::core::Pose&)
bool getMapLocalization()

bool setMapLocalization(bool)
bool getMapUpdate()
bool setMapUpdate(bool)
int getLocalizationQuality()
rpos::actions::MoveAction moveTo(const std::vector<rpos::core::Location>&, bool, bool)
rpos::actions::MoveAction moveTo(const rpos::core::Location&, bool, bool)
rpos::actions::MoveAction moveTo(const std::vector<rpos::core::Location>&, rpos::robot::option::MoveOption&)
rpos::actions::MoveAction moveTo(const rpos::core::Location&, rpos::robot::option::MoveOption&)
rpos::actions::MoveAction moveBy(const rpos::core::Direction& direction)
rpos::actions::MoveAction rotateTo(const rpos::core::Rotation&)
rpos::actions::MoveAction rotate(const rpos::core::Rotation&)
rpos::actions::MoveAction getCurrentAction()
rpos::features::motion_planner::Path searchPath(const rpos::core::Location& location)
rpos::actions::SweepMoveAction startSweep()
rpos::actions::SweepMoveAction sweepSpot(const rpos::core::Location& location)
rpos::actions::MoveAction goHome()
int getBatteryPercentage()
bool getBatteryIsCharging()
bool getDCIsConnected()
int getBoardTemperature()
std::string getSDPVersion()
std::string getSDKVersion()
rpos::features::system_resource::LaserScan getLaserScan()
bool restartModule(rpos::features::system_resource::RestartMode mode = rpos::features::system_resource::RestartModeSoft)
bool setSystemParameter(const std::string& param, const

std::string& value)
std::string getSystemParameter(const std::string& param)
_rpos::features::system_resource::DeviceInfo getDeviceInfo()
void
startCalibration(_rpos::features::system_resource::CalibrationType type)
void stopCalibration()
rpos::features::system_resource::BaseHealthInfo
getRobotHealth()
void clearRobotHealth(int errorCode)
bool
configurateNetwork(rpos::features::system_resouce::NetworkMode mode, const std::map<std::string, std::string>& options)
std::map<std::string, std::string> getNetworkStatus()
bool
getSensors(std::vector<features::impact_sensor::ImpactSensorInfo>& sensors)
bool
getSensorValues(std::map<features::impact_sensor::impact_sensor_id_t, features::impact_sensor::ImpactSensorValue>& values)
bool getSensorValues(const
std::vector<features::impact_sensor::impact_sensor_id_t>&
sensorIds,
std::vector<features::impact_sensor::ImpactSensorValue>&
values)
bool
getSensorValue(features::impact_sensor::impact_sensor_id_t
sensorId, features::impact_sensor::ImpactSensorValue& value)
void setCompositeMap(const
rpos::robot_platforms::objects::CompositeMap& , const
core::Pose&)
rpos::robot_platforms::objects::CompositeMap getCompositeMap()

继承自 rpos::core::RobotPlatform 类的方法

> std::vector<Feature> getFeatures()
> template<class RobotPlatformT> RobotPlatformT cast()

SlamwareCorePlatform(boost::shared_ptr<detail::SlamwareCorePlatformImpl>)构造器

该构造器仅限 SDK 内部使用。

SlamwareCorePlatform(const SlamwareCorePlatform&)构造器

拷贝构造函数。

SlamwareCorePlatform& operator=(const SlamwareCorePlatform&)运算符

赋值运算符。

SlamwareCorePlatform connect(const std::string&, int, int)

连接到指定的 Slamware 设备。

参数

名称	类型	说明
host	const std::string&	Slamware Core 的 IP 地址
port	int	Slamware Core 的端口（通常为 1445）
timeout_in_ms	int	连接超时时间，单位为毫秒

void disconnect()

断开与 CORE 之间的连接。

std::vector<rpos::core::Line> getWalls()

获取系统中所有的虚拟墙。

bool addWall(const rpos::core::Line&)

添加虚拟墙。

bool addWalls(const std::vector<rpos::core::Line>&)

添加多个虚拟墙。

bool clearWallById(const rpos::core::SegmentID&)

清除指定的虚拟墙。

bool clearWalls()

清除所有的虚拟墙。

std::vector<rpos::features::location_provider::MapType> getAvailableMaps()

获得该 Slamware CORE 提供的所有地图类型。

rpos::features::location_provider::Map getMap(
rpos::features::location_provider::MapType,
rpos::core::RectangleF,
rpos::features::location_provider::MapKind)

获得该 Slamware CORE 提供的指定地图类型指定区域的地图数据。

参数

名称	类型	说明
type	rpos::features::location_provider::MapType	地图的数据类型
area	core::RectangleF	地图的区域
kind	rpos::features::location_provider::MapKind	地图类型

示例

```
rpos::feature::location_provider:MapType mapType =
rpos::feature::location_provider:MapType::MapTypeBitmap8Bit;
rpos::feature::location_provider:Mapkind mapKind =
rpos::feature::location_provider:MapKind::EXPLORERMAP;
rpos::core::Rectangle knownArea = robotPlatform.getKnownArea(mapType,
mapKind);
rpos::feature::location_provider:Map map =
robotPlatform.getMap(mapType, knownArea, mapKind);
```

注： 扫地机版本 `mapkind` 可以使用 `SWEEPERMAP`

bool setMap(
const rpos::features::location_provider::Map&,
rpos::features::location_provider::MapType,
rpos::features::location_provider::MapKind, bool partially)

上载指定地图类型指定区域的地图数据到该 Slamware CORE。

参数

名称	类型	说明
map	rpos::features::location_provider::Map	地图
type	rpos::features::location_provider::MapType	地图数据类型
kind	rpos::features::location_provider::MapKind	地图类型
partially	bool	是否部分更新地图

示例

```
rpos::feature::location_provider:MapType mapType =
rpos::feature::location_provider:MapType::MapTypeBitmap8Bit;
rpos::feature::location_provider:Mapkind mapKind =
rpos::feature::location_provider:MapKind::EXPLORERMAP;
rpos::core::Rectangle knownArea = robotPlatform.getKnownArea(mapType, mapKind);
rpos::feature::location_provider:Map map = robotPlatform.getMap(mapType, knownArea,
mapKind);
bool bRet = robotPlatform.setMap(map, mapType, mapKind);
```

bool setMap(const core::Pose& pose, const rpos::features::location_provider::Map&, rpos::features::location_provider::MapType, rpos::features::location_provider::MapKind, bool partially)

上载指定地图类型指定区域的地图数据到该 SIamware CORE。

参数

名称	类型	说明
pose	core::Pose	机器人的 pose 信息
map	rpos::features::location_provider::Map	地图
type	rpos::features::location_provider::MapType	地图数据类型
kind	rpos::features::location_provider::MapKind	地图类型
partially	bool	是否部分更新地图

示例

```
rpos::core::Pose pose;
rpos::feature::location_provider:MapType mapType =
rpos::feature::location_provider:MapType::MapTypeBitmap8Bit;
```

```
rpos::feature::location_provider:Mapkind mapKind =
rpos::feature::location_provider:MapKind::EXPLORERMAP;
rpos::core::Rectangle knownArea = robotPlatform.getKnownArea(mapType, mapKind);
rpos::feature::location_provider:Map map = robotPlatform.getMap(mapType, knownArea,
mapKind);
bool bRet = robotPlatform.setMap(pose, map, mapType, mapKind);
```

rpos::core::RectangleF getKnownArea(rpos::features::location_provider::MapType, rpos::features::location_provider::MapKind)

获得指定地图类型的地图中，已经完成建图的区域。

示例

```
rpos::feature::location_provider:MapType mapType =
rpos::feature::location_provider:MapType::MapTypeBitmap8Bit;
rpos::feature::location_provider:Mapkind mapKind =
rpos::feature::location_provider:MapKind::EXPLORERMAP;
rpos::core::Rectangle knownArea = robotPlatform.getKnownArea(mapType, mapKind);
```

bool clearMap()

清除地图数据。

bool clearMap(rpos::features::location_provider::MapKind kind)

清除指定地图类型的地图数据。

rpos::core::Location getLocation()

获得机器人在上述地图坐标系统中的坐标。

rpos::core::Pose getPose()

获得机器人在上述地图坐标系统中的姿态。

bool setPose(const core::Pose&)

获得机器人在上述地图坐标系统中的姿态。

bool getMapLocalization()

获取地图定位。

bool setMapLocalization(bool)

设定是否启用定位功能。

bool getMapUpdate()

获得是否启用地图更新功能。

bool setMapUpdate(bool)

设定是否启用地图更新功能。

int getLocalizationQuality()

获取雷达定位点的可信度(返回一个 0 到 100 的数值,数值越高 当前雷达点的定位越可信，建议取 50 以上的定位点)

rpos::actions::MoveAction moveTo(const std::vector< rpos::core::Location>&, bool, bool)

让机器人沿着路径移动（机器人会逐一走过路径中的节点，在节点之间尽量走圆滑的曲线，如果遇到障碍物，机器人会自动避开）。

参数详细参考 rpos::action::MoveAction MoveTo(const std::vector<rpos::core::Location>&, bool, bool)

rpos::actions::MoveAction moveTo(const rpos::core::Location&, bool, bool)

让机器人移动到目标位置。

参数详细参考 rpos::action::MoveAction MoveTo(const rpos::core::Location&, bool, bool)

rpos::actions::MoveAction moveTo(const std::vector< rpos::core::Location>&, rpos::robot::option::MoveOption &)

让机器人沿着路径移动（机器人会逐一走过路径中的节点，并根据 MoveAction 参数的变量决定机器人行走的方式和移动过朝向）。

参数

名称	类型	说明
locations	const std::vector <rpos::core::Location>&	期望机器人经过的点
moveOption	rpos::robot::option::MoveOption	如果机器人正在执行其他的移动动作，该参数决定新的点是追加或是替换既有节点

moveOption 详细可参考 rpos::robot::option::MoveOption

示例

```
std::vector<rpos::core::Location> locations;
rpos::core::Location location(1,1);
locations.push
rpos::robot::option::MoveAction moveOption;
moveOption.appending = false;
moveOption.isMilestone = true;
rpos::core::Pose pose(Rotation(2));
rpos::robot::heading::RobotHeading
robotHeading(rpos::robot::heading::HeadingMode::HeadingModeFixAngle, pose);
moveOption.robotHeading = robotHeading;
rpos::actions::MoveAction moveInfo = robotPlatform.moveTo(locations, moveOption);
```

rpos::actions::MoveAction moveTo(const rpos::core::Location&, rpos::robot::option::MoveOption&)

让机器人沿着路径移动（机器人会逐一走过路径中的节点，并根据 MoveAction 参数的变量决定机器人行走的方式和移动过朝向）。

详细参考 rpos::actions::MoveAction moveTo(const std::vector<rpos::core::Location>&,rpos::robot::option::MoveAction&)

rpos::actions::MoveAction moveBy(const rpos::core::Direction&)

控制机器人的运行方向

参数

名称	类型	说明
direction	const rpos::core::Direction&	机器人运行的方向

示例

```
rpos::core::ACTION_DIRECTION actionDirection = rpos::core::ACTION_DIRECTION::FORWARD;
rpos::core::Direction direction(actionDirection);
rpos::actions::MoveAction moveBy = platform.moveBy(direction);
```

注：direction 参数 `rpos::core::ACTION_DIRECTION` 取值说明如下

值	单位	说明
FORWARD		向前
BACKWARD		向后
TURNRIGHT		向右
TURNLEFT		向左

rpos::actions::MoveAction rotateTo(const rpos::core::Rotation&)

使机器人水平转动到特定的角度。

rpos::actions::MoveAction rotate(const rpos::core::Rotation&)

使机器人水平转动一定的角度。

rpos::actions::MoveAction getCurrentAction()

获得机器人当前正在进行的移动动作。

您可以使用 rpos::core::Action::isEmpty()方法判断它是否存在。当机器人当前没有正在进行的动作时，它 rpos::core::Action::isEmpty()将会返回 true。

rpos::features::motion_planner::Path searchPath(const rpos::core::Location& location)

使用机器人内置的寻路算法寻找前往指定目的地的路径。

rpos::actions::SweepMoveAction startSweep()

命令机器人开始打扫。（此接口仅适用于扫地机版本）

rpos::actions::SweepMoveAction sweepSpot(const rpos::core::Location& location)

在机器人处于扫地状态下，可以命令机器人定点清扫。（此接口仅适用于扫地机版本）

rpos::actions::MoveAction goHome()

在机器人处于扫地状态下，可命令机器人返回充电位置。（此接口仅适用于扫地机版本）

int getBatteryPercentage()

获得机器人电池剩余电量（0 表示完全没电，100 表示电量全满）。

bool getBatteryIsCharging()

获得机器人是否正在充电。

bool getDCIsConnected()

获得直流电源是否插上。

int getBoardTemperature()

获得机器人核心温度。单位是 0.1℃，比如该函数的返回结果为 452，则表示机器人的核心温度为 45.2℃。

std::string getSDPVersion()

获得机器人底盘的版本号。

std::string getSDKVersion()

获得 SDK 的版本号。

rpos::features::system_resource::LaserScan getLaserScan()

获得上一次激光扫描的原始数据。

bool restartModule(rpos::features::system_resource::RestartMode mode = rpos::features::system_resource::RestartModeSoft)

根据指定的重启模式（默认软复位）执行重启操作。

重启模式：

RestartModeSoft(软复位)，重启 SDK，速度较快。建议使用。

RestartModeHard(硬复位)，速度较慢，需要几分钟时间。不建议经常使用。

bool setSystemParameter(const std::string& param, const std::string& value)

调整系统参数

参数

名称	类型	说明
param	const std::string&	调整的参数名
value	Const std::string &	调整的参数值

注：目前只支持调整系统速度的设置

param 只能取值为 SYSPARAM_ROBOT_SPEED

value 对应可取如下三种

1.SYSVAL_ROBOT_SPEED_HIGH（高）

2.SYSVAL_ROBOT_SPEED_MEDIUM（中）

3.SYSVAL_ROBOT_SPEED_LOW（低）

示例

```
Bool bRet = platform.setSystemParameter(SYSPARAM_ROBOT_SPEED,
SYSVAL_ROBOT_SPEED_HIGH);
```

std::string getSystemParameter(const std::string& param)

获取系统参数

参数

名称	类型	说明
param	const std::string&	要获取的系统参数名

注：目前只支持调整系统速度的设置

param 只能取值为 SYSPARAM_ROBOT_SPEED

示例

```
std::string robotSpeed = platform.getSystemParameter(SYSPARAM_ROBOT_SPEED);
```

rpos::features::system_resource::DeviceInfo getDeviceInfo()

获取设备信息。设备信息包括设备 ID，制造商 ID，制造商名称，型号 ID，型号名称，硬件版本，软件版本。

具体返回值信息请参考：rpos::features::system_resource::DeviceInfo 类

Void startCalibration(rpos::features::system_resource::CalibrationType type)

机器人开始进行磁罗盘校正

Void stoptCalibration()

机器人停止进行磁罗盘校正

rpos::features::system_resource::BaseHealthInfo getRobotHealth()

获取机器人当前的状态信息

void clearRobotHealth(int errorCode)

清除机器人当前出错的状态信息

bool configureNetwork(rpos::features::system_resource::NetworkMode mode, const std::map<std::string, std::string>& options)

配置机器人的网络信息

参数

网络状况	ssid	password	channel

NetworkModeAp	可选	可选	可选
NetworkModeStation	必选	可选	--
NetworkModeWifiDisabled	--	--	--

注：目前暂只支持以上三中 mode 形式，ssid，password，channel 三个选项的信息如上表（--表示不可用）

示例

```
std::map<std::string, std::string> options;
options[“ssid”] = “Slamtec”;
options[“password”] = “slamtect”;
Bool bRet =
platform.configureNetwork(rpos::features::system_resource::NetworkMode::NetworkModeStation,options);
```

std::map<std::string, std::string> getNetworkStatus()

获取机器人当前的网络信息。

注：目前返回的结果只包含 mode，ssid 和 ip 三个选项的值

bool getSensors(std::vector<ImpactSensorInfo>& sensors)

用于获取机器人上已安装的所有碰撞传感器，返回值为 ImpactSensorInfo 列表。

ImpactSensorInfo 数据结构如下：

```
struct ImpactSensorInfo {
        impact_sensor_id_t id;
        rpos::core::Pose pose;
        ImpactSensorType type;
        float refreshFreq;
};
```

字段说明：

字段名称	单位	说明
Id		Id 字段为后续 API 中会用到的数值。
Pose		Pose 字段表示传感器的安装姿态，即传感器相对于机器人中间的位置和方向。
Type		如果机器人正在执行其他的移动动作，该参数表示传感器类型，为 ImpactSensorTypeDigital 或 ImpactSensorTypeAnalog 中的一种。前者表示普通的碰撞传感器，只有发生碰撞和不发生碰撞两种状态。后者表示距离传感器，如超声波测距传感器，红外测距传感器等。
refreshFreq	Hz	表示传感器刷新频率，Hz 表示每秒刷新次数。

bool getSensorValues(std::map<impact_sensor_id_t, ImpactSensorValue>& values)

用于获取当前碰撞传感器状态，返回值为 map，key 字段为上述 API 中获得的 id，value 字段是 ImpactSensorValue 类型，数据结构如下：

```
struct ImpactSensorValue {
        impact_sensor_timestamp_t time;
        float value;
};
```

字段说明：

字段名称	类型	单位	说明
Time	Long	微秒	表示获得该数据的时间。
Value	Float	米	表示该碰撞传感器检测到的与障碍物之间的距离。若传感器为数字传感器，则 0~FLT_EPSILON 表示发生了碰撞，FLT_MAX 表示没有发生碰撞（建议用“<FLT_EPSILON”来判断）；若传感器为模拟传感器，则 value 表示传感器和障碍物之间的距离，FLT_MAX 表示没有检测到障碍物（建议用“<1000”来判断，1000 为场景中最长轴长度的两倍）。

bool getSensorValues(const std::vector<features::impact_sensor::impact_sensor_id_t>& sensorIds, std::vector<features::impact_sensor::ImpactSensorValue>& values)

获取指定传感器的数据。返回值为 ImpactSensorValue 数组。

bool getSensorValue(features::impact_sensor::impact_sensor_id_t sensorId, features::impact_sensor::ImpactSensorValue& value)

获取指定传感器的数据。返回值为 ImpactSensorValue。

void setCompositeMap(const rpos::robot_platforms::objects::CompositeMap& , const core::Pose&)

设置地图信息。

示例

```
auto pose = platform.getPose();

rpos::robot_platforms::objects::Metadata metadata;
std::vector< boost::shared_ptr<rpos::robot_platforms::objects::MapLayer> > maps;

auto map_layer_v_walls =
boost::make_shared<rpos::robot_platforms::objects::LineMapLayer>();
maps.push_back(map_layer_v_walls);
map_layer_v_walls->setUsage("virtual_walls");
map_layer_v_walls->setType(rpos::robot_platforms::objects::LineMapLayer::Type);
rpos::robot_platforms::objects::Line line(Point(0, 0), Point(10, 10));
line.name =”1”;
map_layer_v_walls->lines()[line.name] = line;

rpos::robot_platforms::objects::CompositeMap compositeMap(metadata, maps);
  platform. setCompositeMap (compositeMap, pose);
```

rpos::robot_platforms::objects::CompositeMap getCompositeMap()

获取地图信息。

日期	说明
2014-9-2	初始版本
2015-6-8	更名为 Slamware SDK Api Reference，并增加大量内容
2015-12-30	增加 SDK 中与碰撞传感器相关的 2 个 API 的介绍 补充内容 API: rpos::features::location_provider::BitmapMap::setMapData 移除 RoboPeak logo
2016-04-12	增加 SDK 中 start sweep 与 spot cleaning 的 API 介绍
2016-04-19	更换封面图片
2016-05-26	更新文档模板
2016-06-14	删除 SlamwareCorePlatform 中重复的 feature 方法介绍。
2016-07-05	增加 DeviceInfo 类解释
2016-11-21	增加 setCompositeMap 和 getCompositeMap 接口。

图表索引