KBSW180105 SLAMWARE SDK API (Windows)

WindowsSLAMWARE SDKAPI

WindowsSlamware SDK

- Visual Studio 2010 SP1Visual Studio 2010 SP1Visual Studio 20122013

SDK

Slamware SDK

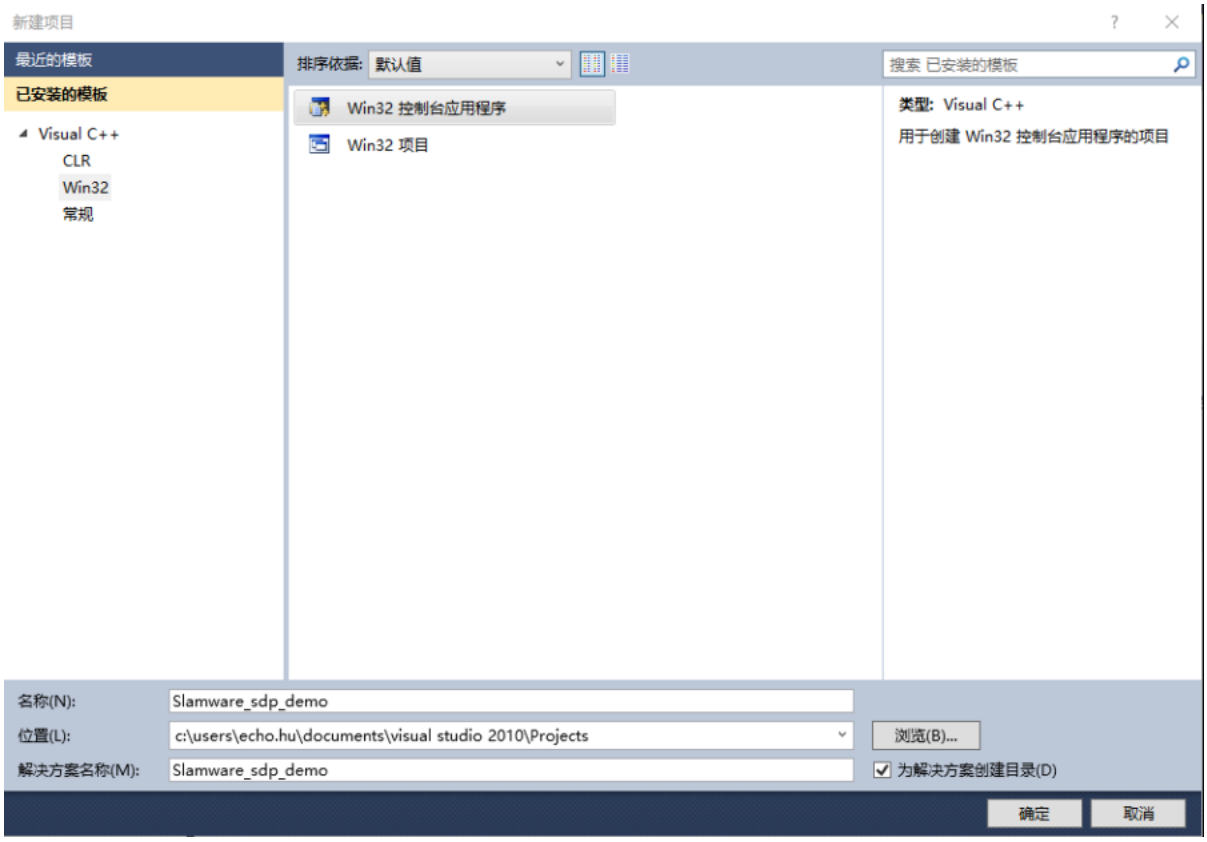
bin	
dll	
docs	
include	SDK

lib	
samples	
workspaces	

includeSlamware SDK

boost	Boost 1.53.0
Eigen	Eigen
json	JsonCpp
rpos	Slamware SDK

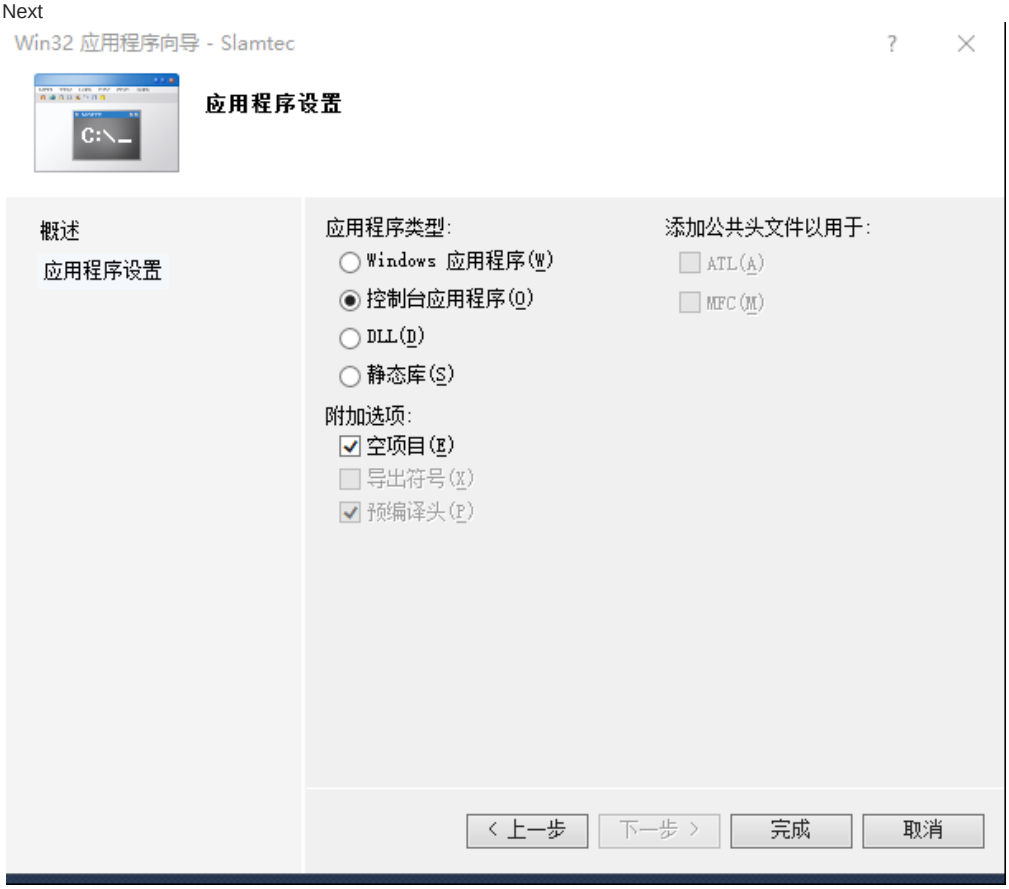
1 Visual Studio 2010



1. Visual C++Win32 Console ApplicationWin32
2. Name
3. OK

2

Next

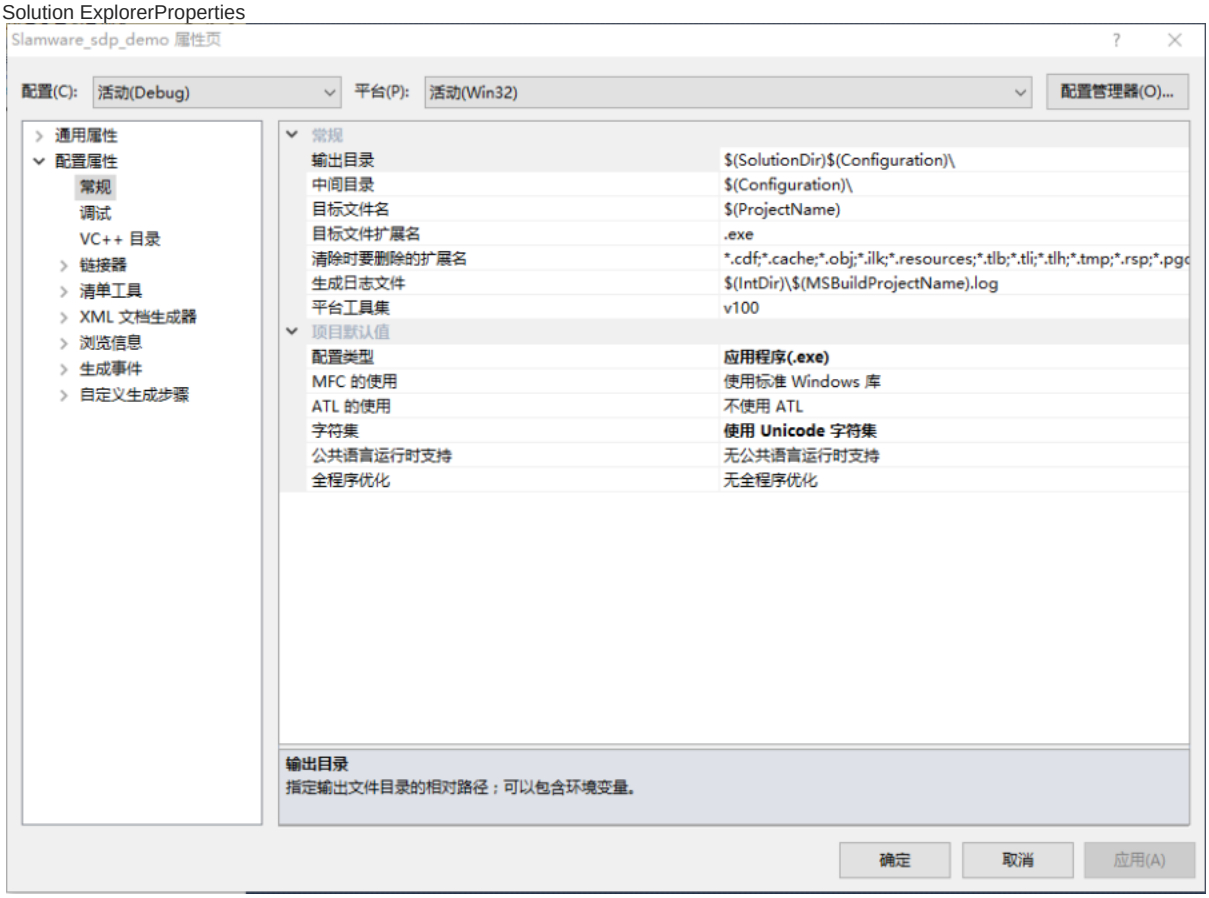


1. Application TypeConsole Application
2. Additional optionsEmpty Project

3. Finish

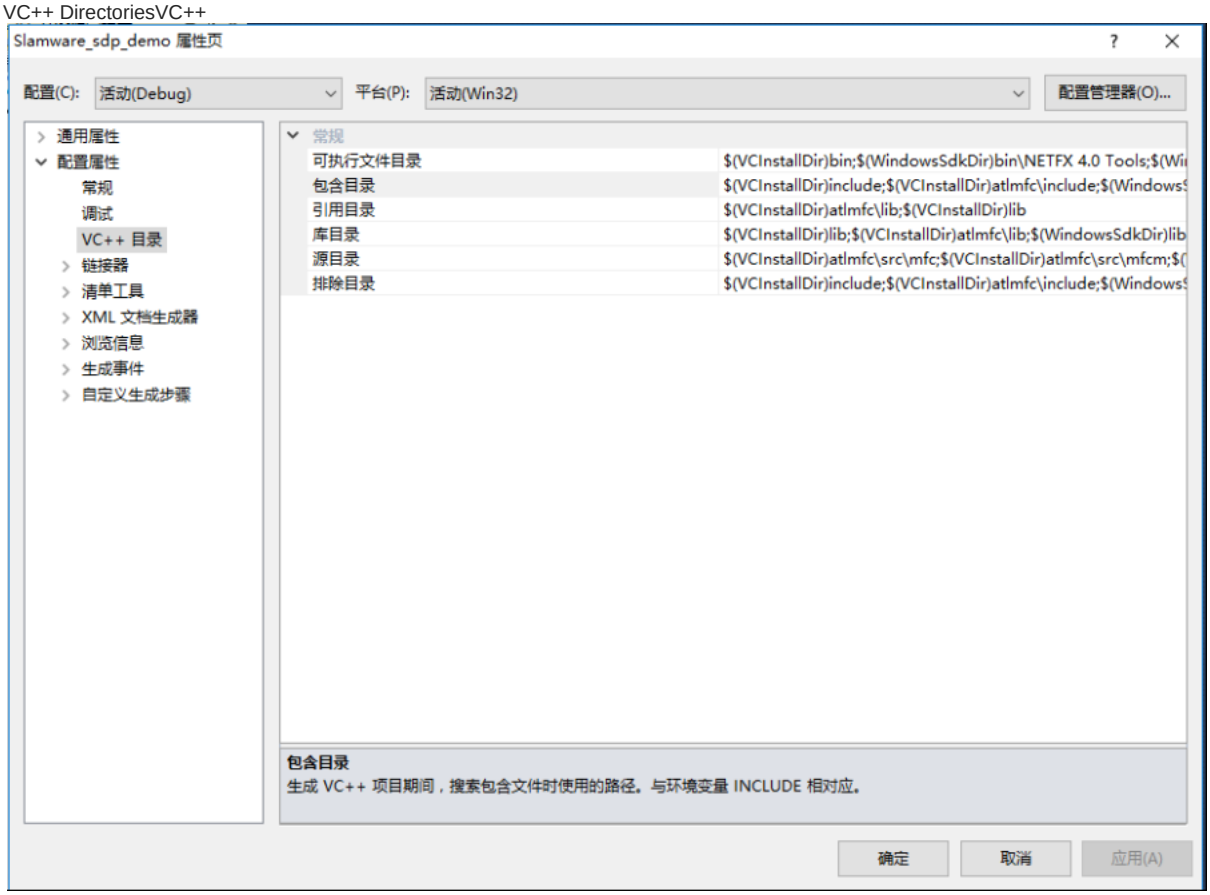
1

Solution ExplorerProperties

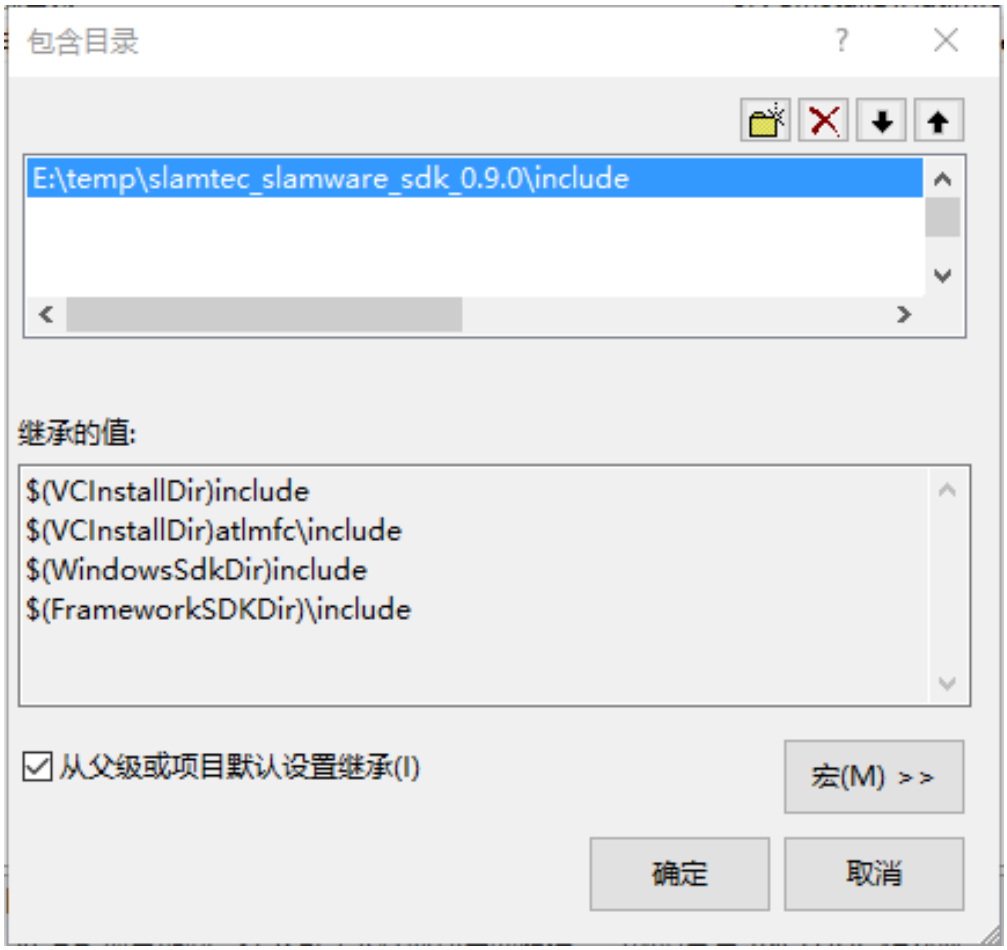


2 VC++

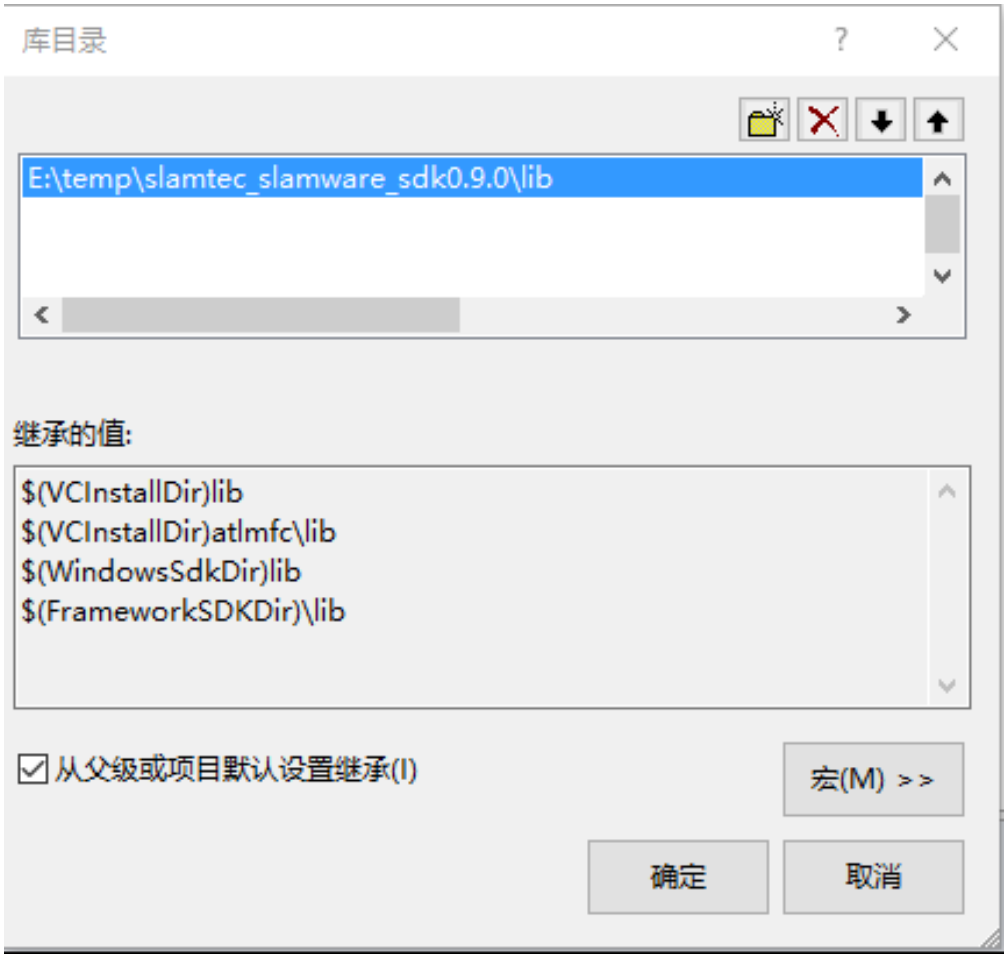
VC++ DirectoriesVC++

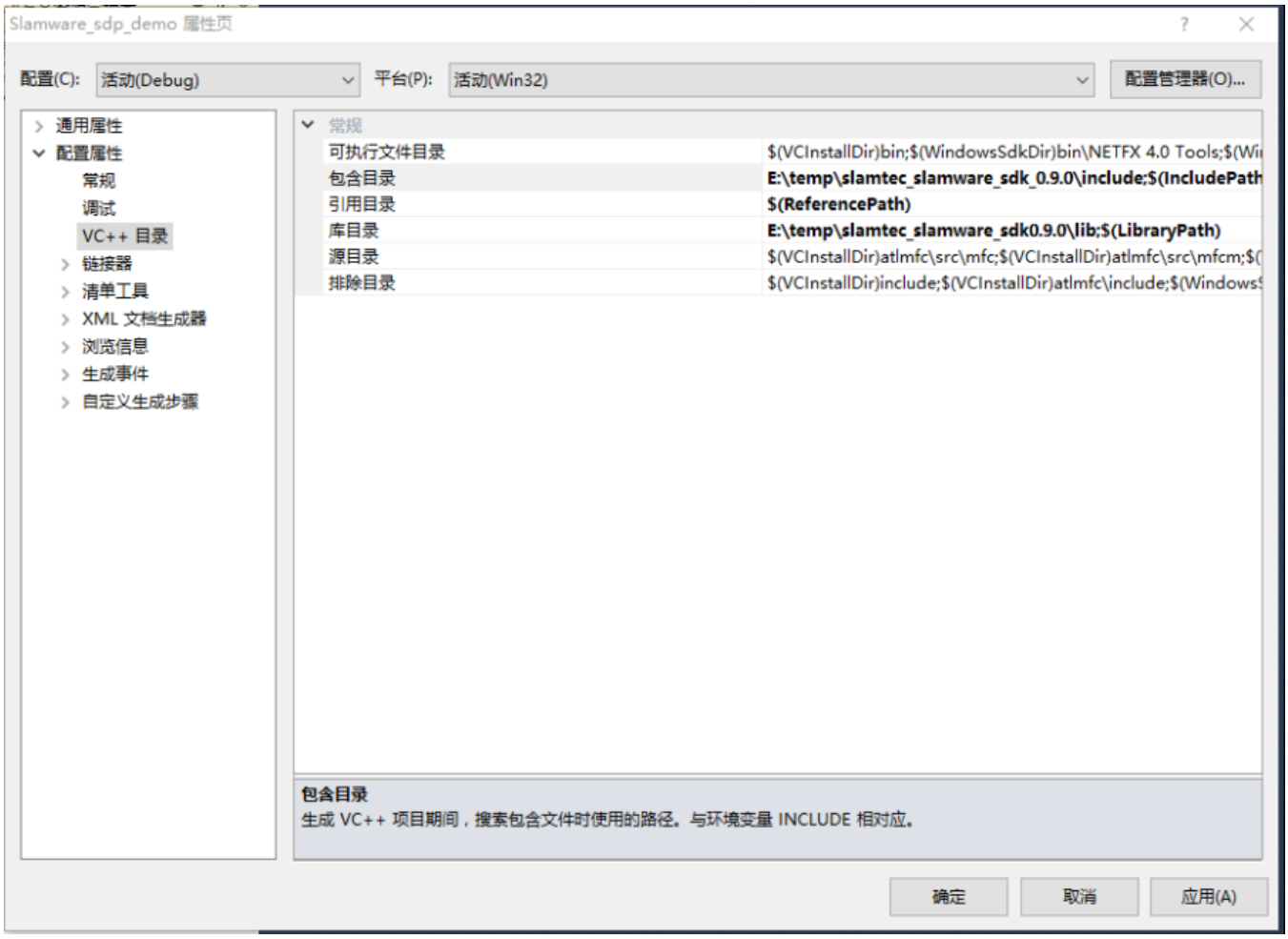


1. Include Directories
2. <Edit…><…>
3. SDKinclude



1. Library Directories
2. <Edit…><…>
3. SDKlib


库目录
E:\temp\slamtec_slamware_sdk0.9.0\lib
继承的值:
$(VCInstallDir)lib
$(VCInstallDir)atlmfc\lib
$(WindowsSdkDir)lib
$(FrameworkSDKDir)\lib
从父级或项目默认设置继承(I)
宏(M) >>
确定
取消

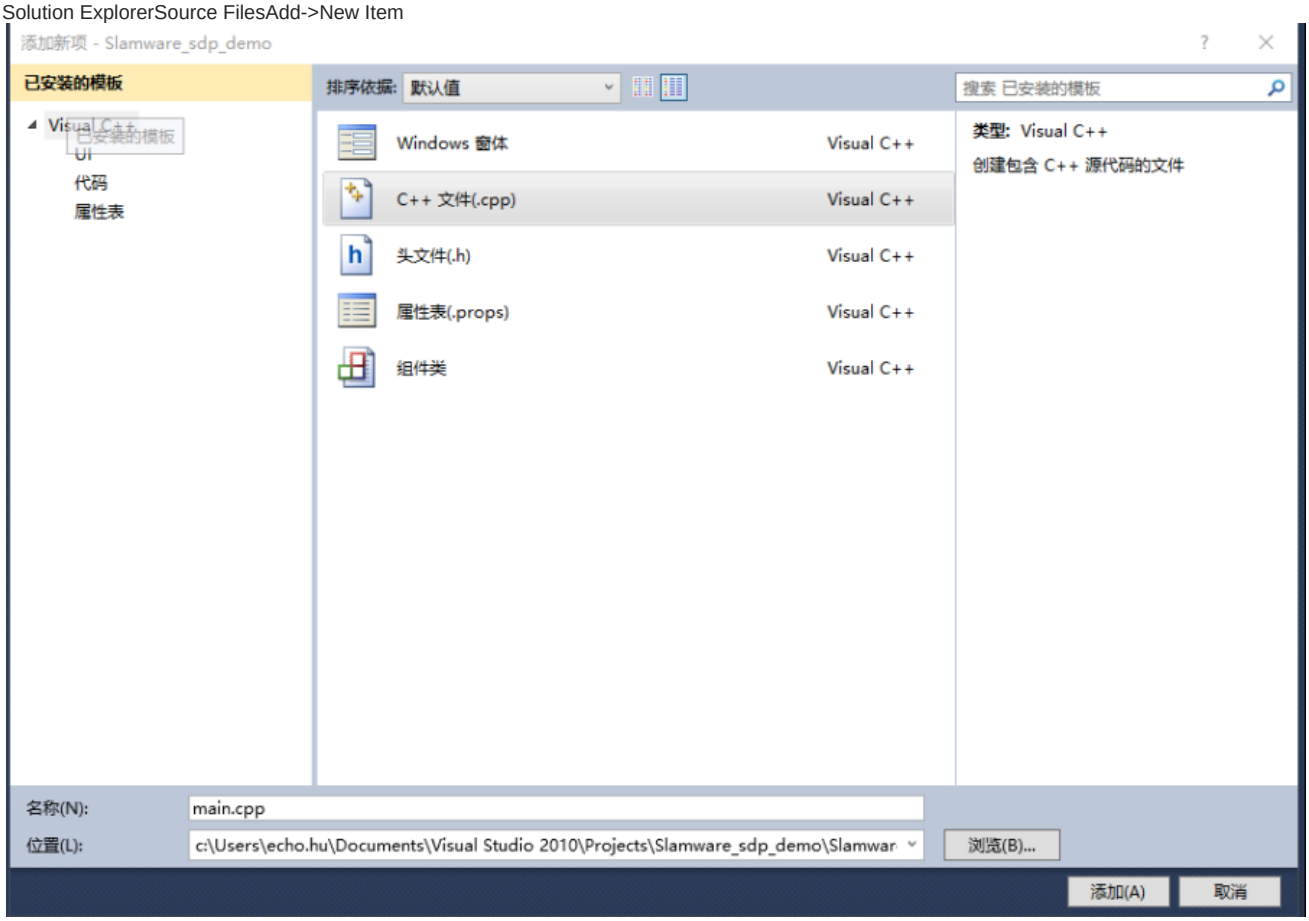


OK

Hello World

1

Solution ExplorerSource FilesAdd->New Item



C++ File (.cpp)main.cpp

2

```
#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("192.168.11.1", 1445);
        cout << "Base version: " << platform.getSDPVersion() << endl;
        return 0;
}
```

3

Visual StudioDebug->Start Debugging

API

rpos::core::Location	
rpos::core::Rotation	

rpos::core::Pose	
rpos::core::Action	
rpos::core::ActionStatus	
rpos::core::Feature	
rpos::core::RectangleF	float
rpos::core::Vector2f	float
rpos::core::Vector2i	int
rpos::core::LaserPoint	
rpos::core::RobotPlatform	
rpos::actions::MoveAction	
rpos::feature::motion_planner::MoveOptionFlag	
rpos::feature::motion_planner::MoveOptions	
rpos::features::ArtifactProvider	
rpos::features::LocationProvider	
rpos::features::MotionPlanner	
rpos::features::SweepMotionPlanner	
rpos::features::SystemResource	
rpos::features::location_provider::Map	
rpos::features::location_provider::MapType	MapType
rpos::features::location_provider::BitmapMap	
rpos::features::location_provider::BitmapMapPixelFormat	BitmapMapPixelFormat
rpos::features::motion_planner::Path	
rpos::features::system_resource::LaserScan	
rpos::robot_platforms::SlamwareCorePlatform	Slamware CORE

rpos::core::Location

Locationxyz

`rpos/core/pose.h`

Location()

Locationxyz0

Location(double x, double y, double z)

Locationxyz

Location(const Location&)

Location& operator=(const Location&)

double x() constdouble& x()

x

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

double y() constdouble& y()

yx

double z() constdouble& z()

zx

rpos::core::Rotation

Rotation

`rpos/core/pose.h`

Rotation()

Rotationyawpitchroll0

Rotation(double yaw, double pitch, double roll)

Rotationyawpitchroll

Rotation(const Rotation&)

Rotation& operator=(const Rotation&)

double yaw() constdouble& yaw()

Tait-Bryan anglesLocation::x()

double pitch() constdouble& pitch()

double roll() const, double& roll()

rpos::core::Pose

PoseLocationRotation

`pos/core/pose.h`

Pose()

xyzyawpitchroll0Pose

Pose(const Location&, const Rotation&)

locationrotationPose

Pose(const Location&)

locationyawpitchroll0Pose

Pose(const Rotation&)

rotationxyz0Pose

Pose(const Pose&)

Pose& operator=(const Pose&)

const Location& location() constLocation& location()

Location

const Rotation& rotation() constRotation& rotation()

Rotation

double x() constdouble& x()

x

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

double y() constdouble& y()

yx

double z() constdouble& z()

zx

double yaw() constdouble& yaw()

Tait-Bryan angles

Location::x()

double pitch() constdouble& pitch()

double roll() const, double& roll()

rpos::core::Action

`rpos/core/action.h`

Action(const Action&)

Action& operator=(const Action&)

ActionStatus getStatus()

ActionStatus

std::string Action::getReason()

Action::getStatus() ActionStatusErrorgetReason

action

“failed” action

“aborted” actioncancelactionaction

moveToaction

“unreachable”

moveToMoveOptionFlagKeyPointsflagmoveto

“blocked[lidar;wall;contact;depth_camera;sonar]” ”[]””;”

lidarwallcontactdepth_camerasonar

void cancel()

ActionStatus waitUntilDone()

ActionStatus

template<class ActionT> ActionT cast()

rpos::core::Action

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

rpos::core::ActionStatus

ActionStatus

`rpos/core/action.h`

ActionStatusWaitingForStart

ActionStatusRunning

ActionStatusFinished

ActionStatusPaused

ActionStatusStopped

ActionStatusError

rpos::core::Feature

Feature

`rpos/core/feature.h`

Feature(const Feature&)

Feature& operator=(const Feature&)

rpos::core::RectangleF

RectangleFfloat

`rpos/core/geometory.h`

RectangleF()

xywidthheight0

RectangleF(Vector2f position, Vector2f size)

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

RectangleF(const RectangleF&)

RectangleF& operator=(const RectangleF&)

const Vector2f& position()Vector2f& position()

const Vector2f& size()Vector2f& size()

float x() constfloat& x()

x

float y() constfloat& y()

y

float width() constfloat& width()

float height() constfloat& height()

float left() const

x

float right() const

xright=x+width

float top() const

y

float bottom() const

ybottom=y+height

bool contains(const Vector2i& point)

bool empty()

width() < epsilonheight()<epsilon

bool contains(const RectangleF& dest)

void unionOf(const RectangleF& dest)

rpos::core::Vector2f

float

`rpos/core/geometry.h`

Vector2f()

xy

Vector2f(float x, float y)

xy

Vector2f(const Vector2f&)

Vector2f& operator=(const Vector2f&)

float x() constfloat& x()

x

float y() constfloat& y()

y

rpos::core::Vector2i

int

`rpos/core/geometry.h`

Vector2i()

xy

Vector2i(float x, float y)

xy

Vector2i(const Vector2i&)

Vector2i& operator=(const Vector2i&)

int x() constint& x()

x

int y() constint& y()

y

rpos::core::LaserPoint

`rpos/core/laser_point.h`

LaserPoint()

LaserPoint

LaserPoint(float distance, float angle, bool valid)

LaserPoint

LaserPoint(const LaserPoint&)

LaserPoint& operator=(const LaserPoint&)

float distance() constfloat& distance()

float angle() constfloat& angle()

bool valid() constbool& valid()

rpos::core::RobotPlatform

RobotPlatform

`rpos/core/robot_platform.h`

RobotPlatform(const RobotPlatform&)

RobotPlatform& operator=(const RobotPlatform&)

std::vector<Feature> getFeatures()

template<class RobotPlatformT> RobotPlatformT cast()

RobotPlatformrpos::core::Action::cast<>

rpos::actions::MoveAction

MoveAction

`rpos/features/motion_planner/move_action.h`

rpos::core::Action

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

SDK

MoveAction(const MoveAction&)

MoveAction& operator=(const MoveAction&)

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

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

rpos::features::motion_planner::MoveOptionFlag

MoveOptionFlagNone

MoveOptionFlagAppending

MoveOptionFlagMilestone

MoveOptionFlagNoSmoot

MoveOptionFlagKeyPoints

MoveOptionFlagPrecise

10cm
MoveOptionFlagWithYaw

rpos::features::motion_planner::MoveOptions

MoveOptionFlag flag

boost::optional<double> speed_ratio;

()

rpos::features::ArtifactProvider

`rpos/features/artifact_provider.h`

rpos::core::Feature

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

SDK

ArtifactProvider(const ArtifactProvider&)

ArtifactProvider& operator=(const ArtifactProvider&)

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()

rpos::features::LocationProvider

SLAM

`rpos/features/location_provider.h`

rpos::core::Feature

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

SDK

LocationProvider(const LocationProvider&)

LocationProvider& operator=(const LocationProvider&)

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(rpos::features::location_provider::MapKind kind = rpos::features::location_provider::EXPLORERMAP);

enum MapKind
EXPLORERMAP = 0,

SWEEPERMAP = 10,

UWBMAP = 20

bool setMapUpdate(bool update, rpos::features::location_provider::MapKind kind = rpos::features::location_provider::EXPLORERMAP);

rpos::features::MotionPlanner

`rpos/features/motion_planner.h`

rpos::core::Feature

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

SDK

MotionPlanner(const MotionPlanner&)

MotionPlanner& operator=(const MotionPlanner&)

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

locations	const std::vector<rpos::core::Location>&	
appending	bool	
isMilestone	bool	true false

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

location	const rpos::core::Location&	
appending	bool	
isMilestone	bool	true false

rpos::actions::MoveAction getCurrentAction()

Action::isEmpty()Action::isEmpty()true

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

rpos::features::SweepMotionPlanner

Slamware Core

`rpos/features/sweep_motion_planner.h`

rpos::core::Feature

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

SDK

SweepMotionPlanner(const SweepMotionPlanner&)

SweepMotionPlanner& operator=(const SweepMotionPlanner&)

rpos::actions::SweepMoveAction startSweep()

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

rpos::actions::MoveAction goHome()

rpos::features::system_resource::DeviceInfo

ID

`rpos/features/device_info.h`

DeviceInfo()

DeviceInfo(const DeviceInfo&)

DeviceInfo& operator=(const DeviceInfo&)

std::string deviceID() conststd::string& deviceID();

deviceID

int manufacturerID() constint& manufacturerID();

manufacturerID

std::string manufacturerName() conststd::string& manufacturerName();

manufacturerName

int modelID() constint& modelID();

modelID

std::string modelName() conststd::string& modelName();

modelName

std::string hardwareVersion() conststd::string& hardwareVersion();

std::string softwareVersion() conststd::string& softwareVersion();

rpos::features::SystemResource

API

`rpos/features/system_resource.h`

rpos::core::Feature

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

SDK

SystemResource(const SystemResource&)

SystemResource& operator=(const SystemResource&)

int getBatteryPercentage()

56%56

bool getBatteryIsCharging()

bool getDCIsConnected()

int getBoardTemperature()

std::string getSDPVersion()

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

features::system_resource::HeartBeatToken startHeartBeat(int heartBeatTimeoutInSeconds);

Slamware CoretokenSlamware core

heartBeatTimeoutInSeconds	int	

void refreshHeartBeat(features::system_resource::HeartBeatToken token);

tokenstartHeartBeattokentimeouttokentimeout

token	features::system_resource::HeartBeatToken	refreshHeartBeatstopHeartBeat

void stopHeartBeat(features::system_resource::HeartBeatToken token);

rpos::features::location_provider::Map

`rpos/features/location_provider.h`

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

SDK

Map(const Map&)

Map& operator=(const Map&)

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()

rpos::features::location_provider::MapType

MapType

`rpos/features/location_provider.h`

MapTypeBitmap8Bit

8

rpos::features::location_provider::BitmapMap

`rpos/features/location_provider.h`

rpos::features::location_provider::Map

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

SDK

BitmapMap(const BitmapMap&)

BitmapMap& operator=(const BitmapMap&)

rpos::features::location_provider::Map

BitmapMapPixelFormat getMapFormat()

rpos::features::location_provider::BitmapMapPixelFormat

BitmapMapPixelFormat

`rpos/features/location_provider.h`

BitmapMapPixelFormat8Bit

1

rpos::features::motion_planner::Path

PathLocation

`rpos/features/motion_planner.h`

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

Path(const Path&)

Path& operator=(const Path&)

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

rpos::features::system_resource::LaserScan

LaserScanLaserPoint

`rpos/features/system_resource.h`

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

LaserScan(const LaserScan&)

LaserScan& operator=(const LaserScan&)

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

rpos::robot_platforms::objects::CompositeMapReader

compositeMapReader.h

rpos::robot_platforms::SlamwareCorePlatform

SlamwareCorePlatformSlamware

rpos/robot_platforms/slamware_core_platform.h

boost::shared_ptr<CompositeMap> loadFile(const std::string& rcFilePath);

CompositeMap CompositeMapException

boost::shared_ptr<CompositeMap> loadFile(const std::wstring& rcFilePath);

CompositeMap CompositeMapException

boost::shared_ptr<CompositeMap> loadFile(std::string& rErrMsg, const std::string& rcFilePath);

CompositeMaprErrMsgrErrMsg

boost::shared_ptr<CompositeMap> loadFile(std::string& rErrMsg, const std::wstring& rcFilePath);

CompositeMaprErrMsgrErrMsg

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

void saveFile(const std::string& rcFilePath, const CompositeMap& rcCmpstMap);

CompositeMap CompositeMapException

void saveFile(const std::wstring& rcFilePath, const CompositeMap& rcCmpstMap);

CompositeMap CompositeMapException

bool saveFile(std::string& rErrMsg, const std::string& rcFilePath, const CompositeMap& rcCmpstMap);

CompositeMaptruefalserErrMsg

bool saveFile(std::string& rErrMsg, const std::wstring& rcFilePath, const CompositeMap& rcCmpstMap);

CompositeMaptruefalserErrMsg

rpos::core::RobotPlatform

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

SDK

SlamwareCorePlatform(const SlamwareCorePlatform&)

SlamwareCorePlatform& operator=(const SlamwareCorePlatform&)

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

Slamware

host	const std::string&	Slamware CoreIP
port	int	Slamware Core1445
timeout_in_ms	int	

void disconnect()

CORE

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()

Slamware CORE

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

Slamware CORE

type	rpos::features::location_provider::MapType	
area	core::RectangleF	
kind	rpos::features::location_provider::MapKind	

```
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);
```

mapkindSWEEPERMAP

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

Slamware CORE

map	rpos::features::location_provider::Map	
type	rpos::features::location_provider::MapType	
kind	rpos::features::location_provider::MapKind	
partially	bool	

```
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)

Slamware CORE

pose	core::Pose	pose
map	rpos::features::location_provider::Map	
type	rpos::features::location_provider::MapType	
kind	rpos::features::location_provider::MapKind	
partially	bool	

```
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)

```
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()

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

rpos::core::Location getLocation()

rpos::core::Pose getPose()

bool setPose(const core::Pose&)

bool getMapLocalization()

bool setMapLocalization(bool)

bool getMapUpdate()

bool setMapUpdate(bool)

int getLocalizationQuality()

(0100, 50)

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

rpos::action::MoveAction MoveTo(const std::vector<rpos::core::Location>&, bool, bool)

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

rpos::action::MoveAction MoveTo(const rpos::core::Location&, bool, bool)

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

MoveAction

locations	const std::vector<rpos::core::Location>&	
moveOption	rpos::robot::option::MoveOption	
yaw	float	

moveOptionMoveOptions

rpos::actions::MoveAction moveTo(const rpos::core::Location&, MoveOptions&, float yaw)

MoveAction

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

direction	const rpos::core::Direction&	

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

directionrpos::core::ACTION_DIRECTION

FORWARD	
BACKWARD	
TURNRIGHT	
TURNLEFT	

actions::MoveAction moveBy(const core::Direction& direction, const features::motion_planner::MoveOptions&);

MoveOptions

moveOptionMoveOptions

actions::MoveAction moveBy(const float theta, const features::motion_planner::MoveOptions&);

thetaMoveOptionsspeed_ratio200ms

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

actions::MoveAction rotateTo(const core::Rotation& orientation, const features::motion_planner::MoveOptions& options);

MoveOptions

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

actions::MoveAction rotate(const core::Rotation& rotation, const features::motion_planner::MoveOptions& options);

MoveOptions

rpos::actions::MoveAction getCurrentAction()

rpos::core::Action::isEmpty()rpos::core::Action::isEmpty()true

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()

0100

bool getBatteryIsCharging()

bool getDCIsConnected()

int getBoardTemperature()

0.145245.2

std::string getSDPVersion()

std::string getSDKVersion()

SDK

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

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

RestartModeSoft()SDK
RestartModeHard()

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

param	const std::string&	
value	Const std::string &	

paramSYSPARAM_ROBOT_SPEED

value

1.SYSVAL_ROBOT_SPEED_HIGH

2.SYSVAL_ROBOT_SPEED_MEDIUM

3.SYSVAL_ROBOT_SPEED_LOW

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

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

param	const std::string&	

paramSYSPARAM_ROBOT_SPEED

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

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

IDIDID

rpos::features::system_resource::DeviceInfo

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

void clearRobotHealth(int errorCode)

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

	ssid	password	channel	ip	dns	gateway	mask
NetworkModeAp							
NetworkModeStation			--	--	--	--	--
NetworkModeWifiDisabled	--	--	--	--	--	--	--

1. modessidpasswordchannel--

2. NetworkModeWifiDisabled

```
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()

modessidip

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

ImpactSensorInfo

ImpactSensorInfo

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

Id		IdAPI
Pose		Pose
Type		ImpactSensorTypeDigital ImpactSensorTypeAnalog
refreshFreq	Hz	Hz

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

mapkeyAPIidvalueImpactSensorValue

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

Time	Long		
Value	Float		0~FLT_EPSILONFLT_MAX“<FLT_EPSILON” valueFLT_MAX “<1000”1000

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

ImpactSensorValue

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

ImpactSensorValue

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

Composite mapMapLayerGridMapLayerLineMapLayer

MapLayer	
GridMapLayer	explorer mapsweep map
LineMapLayer	

```
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);
```

metadata.setUsage()

explore	
sweep	
virtual_walls	

metadata.set<rpos::core::RectangleF>("area", area)

features::system_resource::HeartBeatToken startHeartBeat(int heartBeatTimeoutInSeconds);

Slamware CoretokenSlamware core

heartBeatTimeoutInSeconds	int	

void refreshHeartBeat(features::system_resource::HeartBeatToken token);

tokenstartHeartBeattokentimeouttokentimeout

token	features::system_resource::HeartBeatToken	refreshHeartBeatstopHeartBeat

void stopHeartBeat(features::system_resource::HeartBeatToken token);

features::system_resource::PowerStatus getPowerStatus();

bool isDCConnected;

DockingStatus dockingStatus;

Docking

bool isCharging;

int batteryPercentage;

0~100

SleepMode sleepMode;

enum SleepMode

SleepModeUnknown,

Slamware

SleepModeAwake

The device is waking up, please wait for some time
SleepModeWakingUp

SleepModeAsleep

enum DockingStatus

Docking
DockingStatusUnknown

DockingSlamware

DockingStatusOnDock,

DockingStatusNotOnDock

void wakeUp();

getSweepTimeMs

int getSweepTimeMs();

float getSweepArea();

core::Pose getHomePose();

(0,0)

features::location_provider::PointPDF getAuxLocation();

getAuxLocation()recoverLocalization

```
auto locationPdf = platform. getAuxLocation();
 Location location = locationPdf.location;
 float distant = locationPdf.circular_error_probability;
 RectangleF area((location.x() - distant), (location.y() - distant), 2*distant, 2*distant);
  auto act = platform. recoverLocalization(area);
  act.waitUntilDone();
```

actions::MoveAction recoverLocalization(const core::RectangleF& area);

area20*20

features::motion_planner::Path getRobotTrack(int count);

0

std::vector<core::Line> getLines(features::artifact_provider::ArtifactUsage usage);

ArtifactUsageVirtualWall	
ArtifactUsageVirtualTrack	

bool addLine(features::artifact_provider::ArtifactUsage usage, const core::Line& line);

bool addLines(features::artifact_provider::ArtifactUsage usage, const std::vector<core::Line>& lines);

bool removeLineById(features::artifact_provider::ArtifactUsage usage, rpos::core::SegmentID id);

bool clearLines(features::artifact_provider::ArtifactUsage usage);

detail::objects::UpdateInfo getUpdateInfo();

currentVersion	string	
newVersion	string	
newVersionReleaseDate	string	
newVersionChangeLog	string	

bool startFirmwareUpdate();

detail::objects::UpdateProgress getFirmwareUpdateProgress();

UpdateFirmwareStep currentStep; // Range from 0 to {totalSteps - 1}.
unsigned int totalSteps; // The number of total steps.
std::string currentStepName; // The name of the current step.
unsigned int currentStepProgress; // Expressed as a percentage.
UpdateProgressStatus status; // Current status

enum UpdateFirmwareStep

UpdateFirmwarePreparing

UpdateFirmwarePrepareFinished

UpdateFirmwareDownloading

UpdateFirmwareDownloadFinished

UpdateFirmwareUpdating

UpdateFirmwareUpdateFinished

enum UpdateProgressStatus

UpdateProgressSuccess

UpdateProgressError

UpdateProgressInit,

UpdateProgressUpgrade