KBSW180120 Win32-CompositeMap

composite_map_demo, .stcmcomposite map

- ■
  - ■
    - ■
- ■
- ■
- ■

- ●
  - Visual Studio 2010 SP1
  - Slamware Windows SDK:Slamware Windows SDK
  - RoboStudio():Robostudio installer
  - Sample Code:
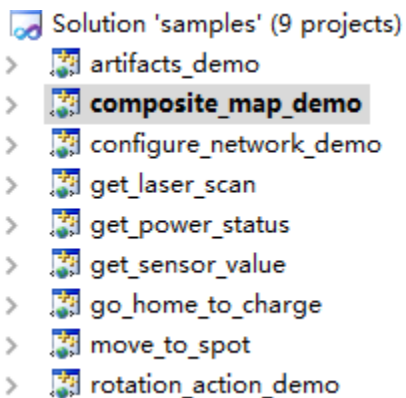
    > Visual Studio
    >
    > Visual Studio 2010SP1.Net FrameworkSP1

- ●

  - Slamware SDP mini
  - Slamware SDP
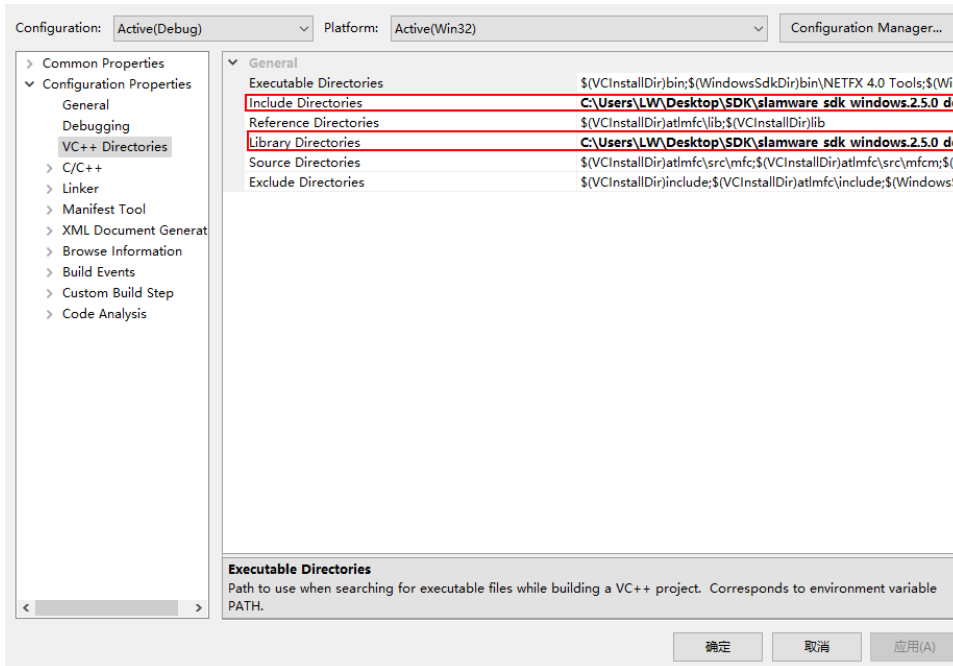 - Slamware Slamware
  - Zeus/Apollo

Win32-

---

1. **samplescomposite_map_demo, StartUp project**

   Solution 'samples' (9 projects)
   - > artifacts_demo
   - > **composite_map_demo**
   - > configure_network_demo
   - > get_laser_scan
   - > get_power_status
   - > get_sensor_value
   - > go_home_to_charge
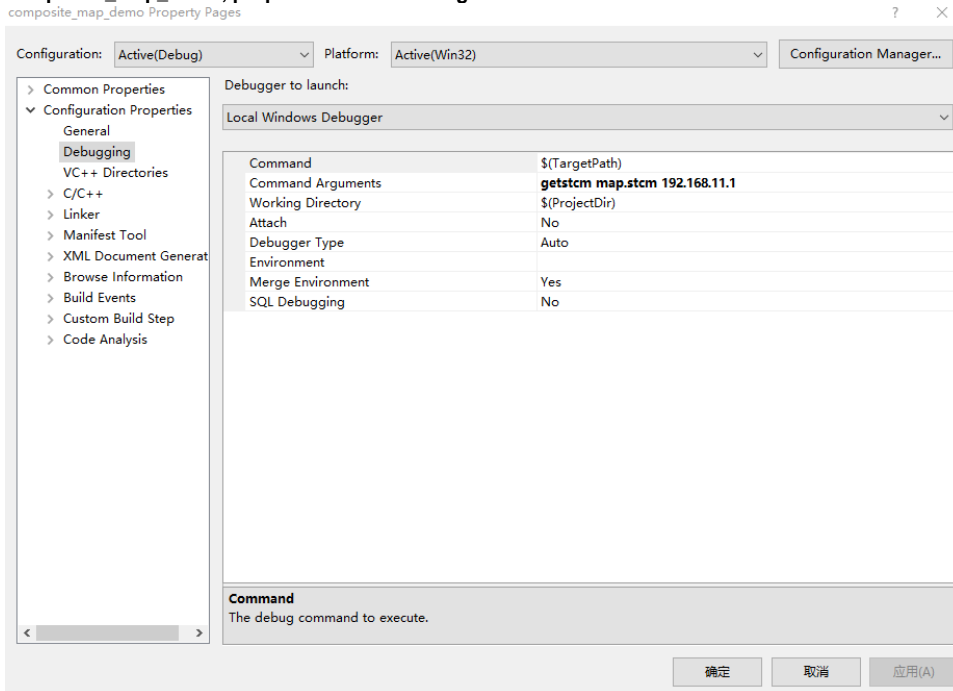   - > move_to_spot
   - > rotation_action_demo

2. **composite_map_demo, Slamware SDK includelib**

Slamware SDKincludelibVisual Studio

3. **composite_map_demo, propertiesCommand Arguments**

composite_map_demo   [OPTS] [filename] <SDP IP Address>
SDP IP Address   The ip address string of the SLAMWARE SDP
getstcm filename   download compositeMap
setstcm filename   upload compositeMap
-h   Show this message

4. **F5**

- **composite_map_demo getstcm map.stcm 192.168.11.1** (slamwarecomposite map, map.stcm)

Robostudio

- **composite_map_demo setstcm map.stcm 192.168.11.1**(map.stcmslamware)

Robostudio

- slamwarecomposite map

composite map

```
bool StcmMapWriter(const std::string file_name, SlamwareCorePlatform platform) {
        CompositeMap composite_map = platform.getCompositeMap();
    CompositeMapWriter composite_map_writer;
    std::string error_message;
    bool result = composite_map_writer.saveFile(error_message, file_name, composite_map);
    return result;
}
```

- composite mapslamware

composite map

```
bool StcmMapReader(const std::string file_path, rpos::core::Pose pose, SlamwareCorePlatform platform) {
    CompositeMapReader composite_map_reader;
    std::string error_message;
    boost::shared_ptr<CompositeMap> composite_map(composite_map_reader.loadFile(error_message, file_path));
    if (composite_map) {
        platform.setCompositeMap((*composite_map), pose);
        return true;
    }
    return false;
}
```

- composite map

map layer

```
    CompositeMapReader composite_map_reader;
    std::string error_message;
    boost::shared_ptr<CompositeMap> composite_map(composite_map_reader.loadFile(error_message, file_path));
    if (composite_map) {
        for (auto it = composite_map->maps().begin(); it != composite_map->maps().end(); ++it) {
            auto layer = *it;
            std::string usage = layer->getUsage();
                        std::string type = layer->getType();
                        std::cout << "Layer Usage : " << usage << std::endl;
                        //get grid map layer
            if (type == GridMapLayer::Type) {
                                auto grid_map = boost::dynamic_pointer_cast<GridMapLayer>(layer);
                                std::cout << "Map Position : (" << grid_map->getOrigin().x() << " , " <<
                                    grid_map->getOrigin().y() << ")" <<std::endl;
                                std::cout << "Map Resolution : (" << grid_map->getResolution().x() <<
                                    " , " << grid_map->getResolution().y() << ")" <<std::endl;
                                std::cout << "Map Dimension: (" << grid_map->getDimension().x() <<
                                    " , " << grid_map->getDimension().y() << ")" <<std::endl;
                                std::cout << "Map Data:" << std::endl;
                                for (auto it = grid_map->mapData().begin(); it != grid_map->mapData().end();
++it) {
                                    std::cout << (int)*it << " " ;
                                }
                                std::cout << std::endl << std::endl;
            }
                        //get line map layer
            else if (type == LineMapLayer::Type) {
                auto line_map = boost::dynamic_pointer_cast<LineMapLayer>(layer);
                                for (auto it = line_map->lines().begin(); it != line_map->lines().end();
++it) {
                                    auto line = it->second;
                                    std::cout << "start: (" << line.start.x() << " , " << line.start.y()
<< ")" << std::endl;
                                    std::cout << "end: (" << line.end.x() << " , " << line.end.y() <<
")" << std::endl;
```

```cpp
                                }
                                std::cout << std::endl;
                }
                        //get pose map layer
                        else if (type == PoseMapLayer::Type) {
                                auto pose_map = boost::dynamic_pointer_cast<PoseMapLayer>(layer);
                                for (auto it = pose_map->poses().begin(); it != pose_map->poses().end();
++it) {

                                        auto pos = it->second;
                                        std::cout << "Position : (" << pos.pose.x() << " , " << pos.pose.y()
<< ")" << std::endl;
                                }
                                std::cout << std::endl;
                        }
                        else if (type == PointsMapLayer::Type) {
                                //TODO: get Points map layer
                                std::cout << std::endl;
                }
                        else {
                                //TODO: get unknown map layer
                                std::cout << std::endl;
                }
        }
```