KBSW183302 Win32-

robot_health,

-
  -
  -
-
-
-

-
  - Visual Studio 2010 SP1
  - Slamware Windows SDK:Slamware Windows SDK
  - RoboStudio():Robostudio installer
  - Sample Code:

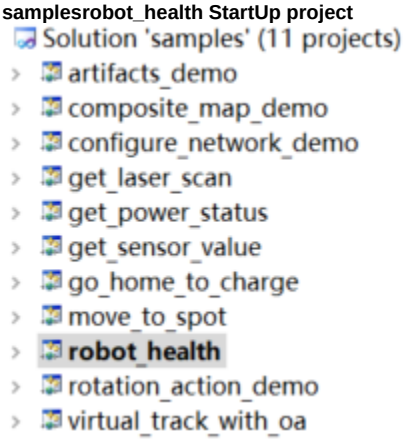
    > Visual Studio
    >
    > Visual Studio 2010SP1.Net FrameworkSP1

-
  - Slamware SDP mini
  - Slamware Slamware
  - Apollo/Ares/Athena

Win32-

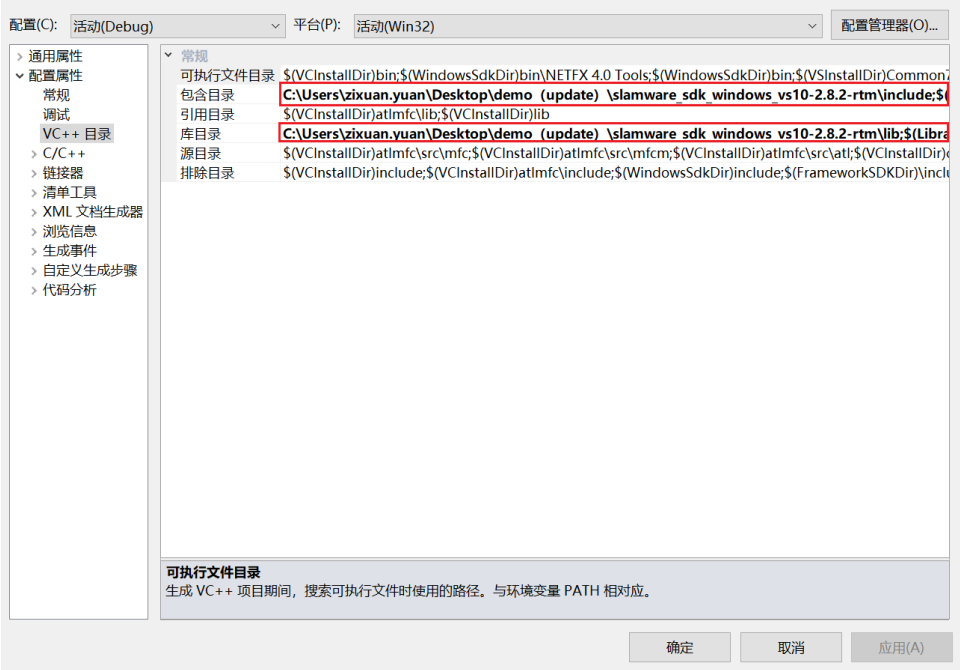
---

1. **samplesrobot_health StartUp project**

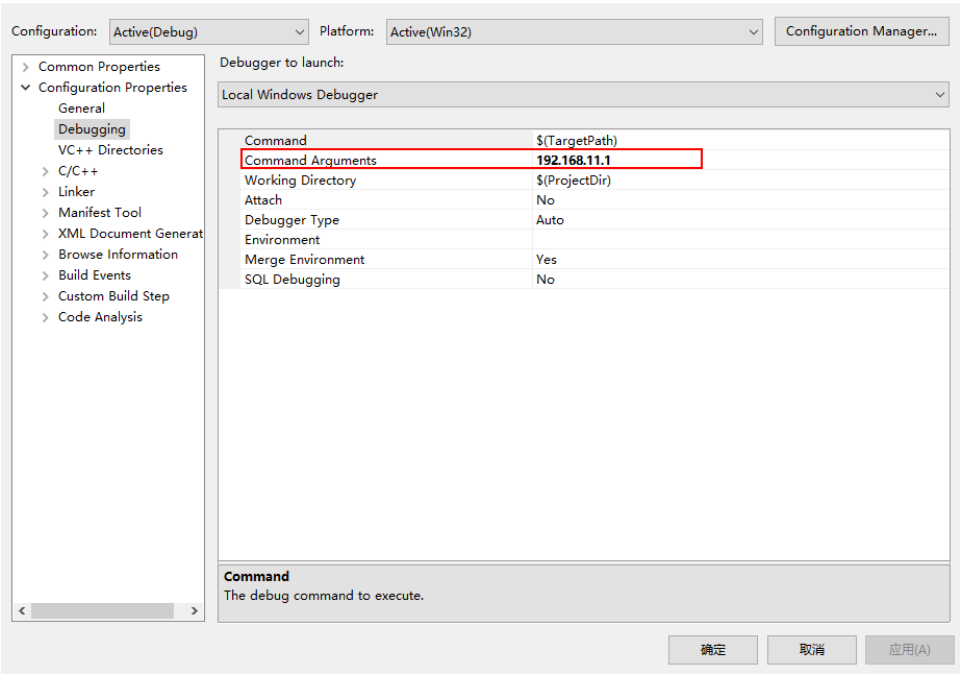


2. **robot_health, Slamware SDK includelib**

   > Slamware SDKincludelibVisual Studio

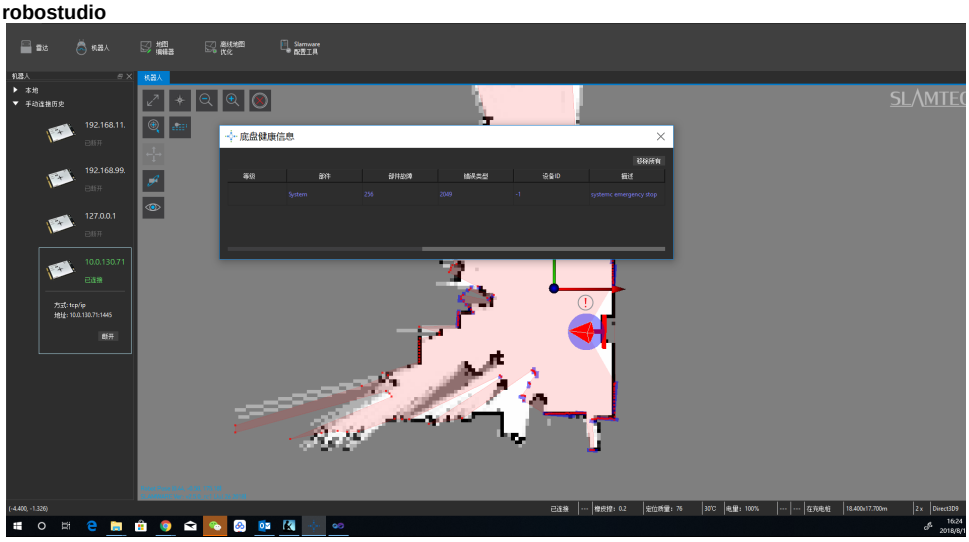


3. **robot_health, Debuggingcommand Arguments 192.168.11.1**
   robot_health <IP address>

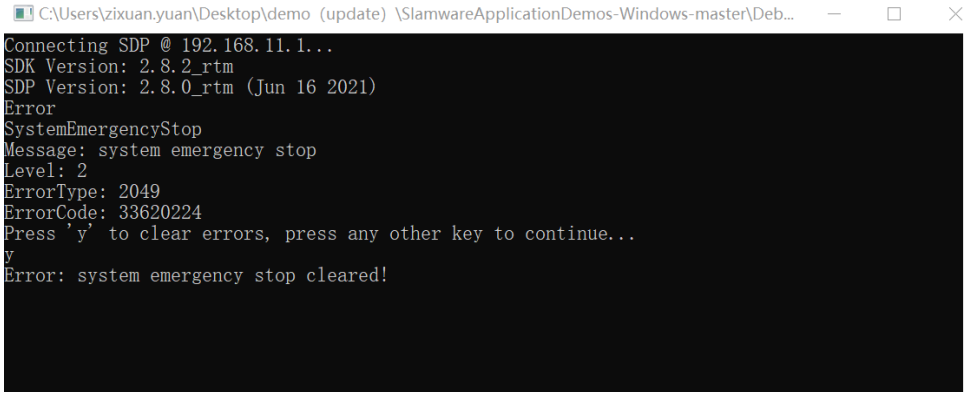


4. **F5**

5. **robostudio**



6. **console**



-

```
        SlamwareCorePlatform sdp = SlamwareCorePlatform::connect(ip_address, 1445);
                std::cout <<"SDK Version: " << sdp.getSDKVersion() << std::endl;
                std::cout <<"SDP Version: " << sdp.getSDPVersion() << std::endl;

                while(true){
                        BaseHealthInfo robot_health = sdp.getRobotHealth();
                        if(robot_health.hasError)
                                std::cout << "Error" << std::endl;
                        if(robot_health.hasFatal)
                                std::cout << "Fatal" << std::endl;
                        if(robot_health.hasWarning)
                                std::cout << "Warning" << std::endl;
                        if(*robot_health.hasLidarDisconnected)
                                std::cout << "LidarDisconnected" << std::endl;
                        if(*robot_health.hasSdpDisconnected)
                                std::cout << "SdpDisconnected" << std::endl;
                        if(*robot_health.hasSystemEmergencyStop)
                                std::cout << "SystemEmergencyStop" << std::endl;
                        for (auto it = robot_health.errors.begin();it != robot_health.errors.end();
++ it) {
                                std::cout << "Message: " << it->message << std::endl;
                                std::cout << "Level: " << it->level << std::endl;
                                std::cout << "ErrorType: "<< it->componentErrorType <<std::endl;
                                std::cout << "ErrorCode: " << it->errorCode << std::endl;
```

```
                        }

                        int errors_size = robot_health.errors.size();
                        if(errors_size > 0){
                                std::cout << "Press 'y' to clear errors, press any other key to
continue..." << std::endl;
                                char is_error_clear;
                                std::cin >> is_error_clear;
                                if(is_error_clear == 'y' || is_error_clear == 'Y') {
                                        for (auto it = robot_health.errors.begin();it != robot_health.
errors.end(); ++ it) {
                                                sdp.clearRobotHealth(it->errorCode);
                                                std::cout << "Error: " << it->message << " cleared!"
<< std::endl;
                                        }
                                }
                        }
                }
```