ISIS-OBC First Project Setup

This document will explain creating new project directories and setting the proper locations of the library directories to work with the ISIS-OBC First-Project Eclipse project.

The first-project is a template eclipse project that can be used to quick start a new ISIS-OBC project. The First-Project directory can be copied in place and renamed for each new project. Next open each of the Eclipse files located in the new project directory (.cproject, .project and *.launch) using a text editor (e.g. notepad) and perform a rename of all occurrences of *isis-obc-first-project* to the new project name.

The linkage to the libraries make use of relative paths, which are pointing to the directory containing the first-project. The ISIS-OBC libraries are distributed as zip files and should be extracted next to the project that will be using them. An example of a directory structure is shown in Figure 1. The example shows the isis-obc-hal-0.1.8 zip which contents, the 'hal' directory, is extracted to the same directory. The first-project can now find the library and will successfully build.

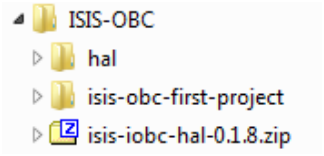

Figure 1: Relative file locations. The library directory should be in the same directory as the project.

Library updates are simple: just replace the existing library directory with the new version of the library. The projects will automatically use the new libraries during the next build.

Linking an Additional Library

Several libraries are available through ISIS. To use an additional library it must be linked into the project. This involves setting include paths and adding the library name in Eclipse project properties. In this example the satellite-subsystems library will be linked.

1. First the new library is extracted to the directory that also includes the project. The directory is shown in Figure 2.

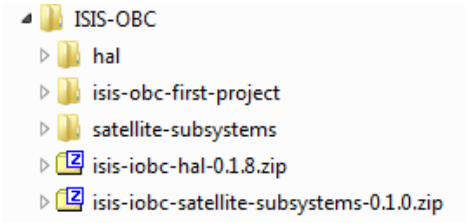

Figure 2: Satellite-subsystems library added to the directory.

2. Add the include path to the library: Open Eclipse, right click the first project and open project properties. Navigate to *C/C++ Build* > *Settings*. Under Tool Settings choose *Cross ARM C Compiler* > *Includes*. Add the path to the *include* folder of the library. For satellite subsystems this is "${external-dependencies}/satellite-subsystems/satellite-subsystems/include". See Figure 3.

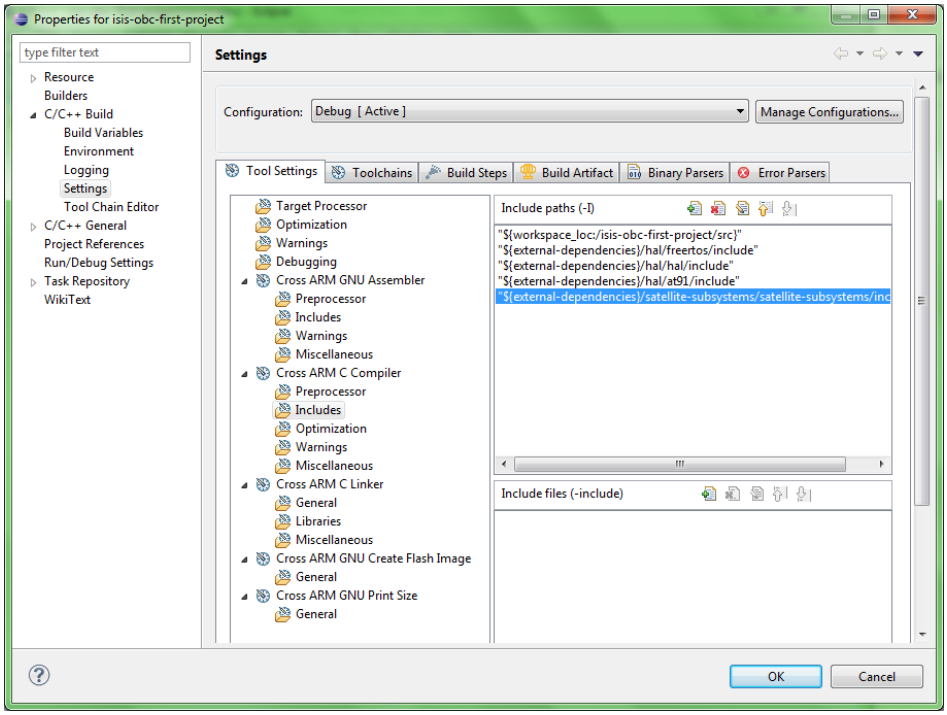

Figure 3: Adding the library include path

3. Add the library to the linker settings: Select *Cross ARM C Linker > Libraries*. Add the library name with a capital D at the end to the *Libraries* list. Add the path to the *lib* directory to the *Library search path* list. See Figure 4.

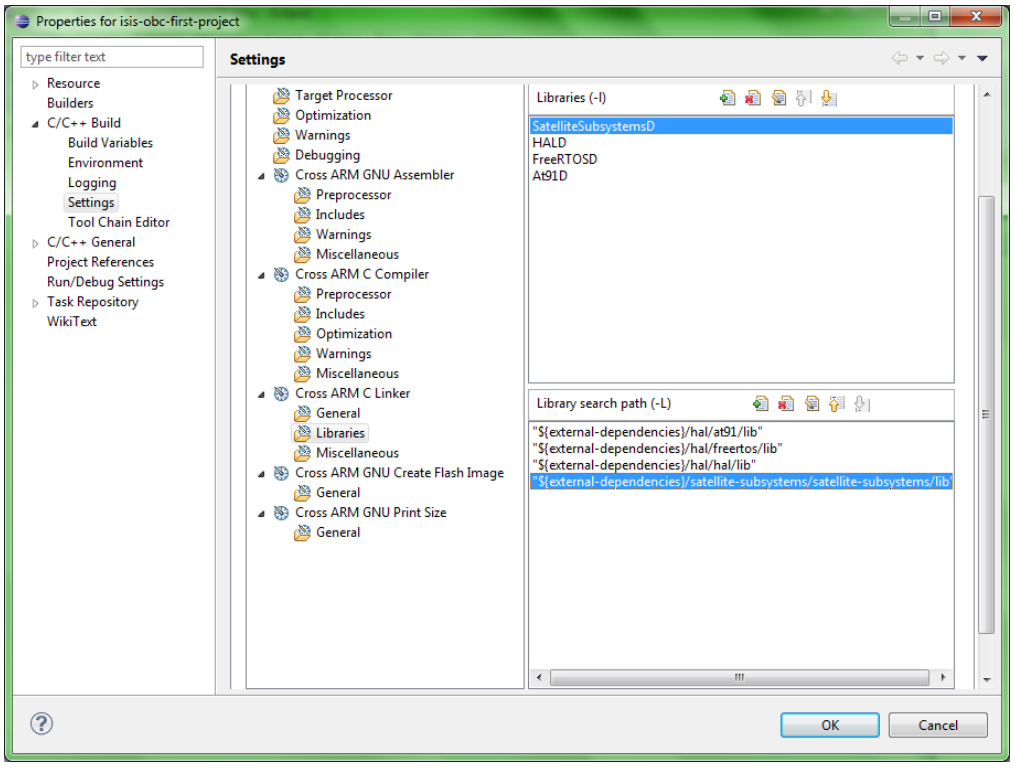

Figure 4: Add the library to the linker settings

Click ok. The library can now be used from within the project.