Doxygen Guideline

Introduction

The purpose of this page is to provide a uniform style of Doxygen commenting for the RDK system. It will serve as a reference for current and future developers, while documenting the RDK system as it evolves. Ultimately, this will establish a consistent manner of documentation to strengthen the simplicity, readability, scalability, writability, reliability, and maintainability of the system.

Documentation Style

Doxygen documentation can be generated in many formats(HTML, LaTeX, RTF, PDF, DOC) . HTML generation has support for more plugins and is easier to refactor as the system changes. Doxygen style should follow a consistent format to aid development across different IDEs. Additionally, it reduces issues when generating documentation.

Standard Doxygen Tag Format

```
/**
 * @tagname
 */
```

This is an example of a Java doc style Doxygen tag, since it uses the "@" symbol. Tags using the "\tagname" style are considered Qt style Doxygen tags.

There should be a header file containing only Doxygen tags or a separate Doxygen file that acts as a guide for the components, classes, methods, and variables (e.g. DoxygenMainpage.h). This can be done using the @mainpage tag at the top of the file.

System

There should be a header file containing only Doxygen tags or a separate Doxygen file that acts as a guide for the components, classes, methods, and variables (e.g. DoxygenMainpage.h). This can be done using the @mainpage tag at the top of the file.

Main Page Tag Example

```
/**
 * @mainpage                    Title of Document
 *
 */
```

Example of HAL system Doxygen Guideline (Note: source code was also modified to support correct generation of documentation)

File

A file should contain the @file tag at the top of the file. This supports generation of a file list tab on the main page. It also helps when files contain multiple classes.

File Tagging Example

```
/**
 * @file                FileName.h
 *
 * @brief               Brief file description.
 *
 *                      Verbose file description.
 */
```

Classes

Classes can be tagged in a number of different ways, but in general they are tagged using the @brief and @class tags before the class declaration. Having the @author, @date, and @version supports tractability as the system is versioned throughout the software lifecycle. When updating classes, update comments like this:

Class Tagging Example

```
#include <iostream>
using namespace std;


/**
 * @brief               Brief class description
 *
 *                      Verbose description of class.
 *
 * @class               Class Name
 */

class ClassName {
               public:
                      ClassName();
                          ~ClassName();

                      int var1;                    /**< Comment about public member variable*/

                      /**
                                  *@brief           Brief method description
                                           *
                                           *                 Verbose description of method
                                           *
                                           *@param           Parameter in the method’s definition
                                           *
                                           *@return          Return value of method
                                       */
                      int Function1(int x);

               protected:
                                          int var2;              /**< Comment about protected member variable*/

                      /**
                                  *@brief            Brief method description
                              *
                                           *                  Verbose description of method
                                       *
                                           *@param            Parameter in the method’s definition
                                           *
                                           *@return           Return value of method
                                           */
                                          int Function2(int x);

               private:
                     int var3;                        /**< Comment about private member variable*/

                                          /**
                                           *@brief             Brief method description
                                            *
                                           *                  Verbose description of method
                                           *
                                           *@param            Parameter in the method’s definition
                                       *
                                           *@return           Return value of method
                                           */
                      int Function3(int x);

};
```

Structs

A struct can be tagged in the same way a class, but it is best to use the @struct tag. When updating structs, update comments like this:

Struct Tagging Example

```
/**
 *@brief             Brief struct description
 *
 *@struct            Struct Name
 */
```

Methods

Methods can be tagged in a number of ways, but in general the @brief, @details, @param, and @return tags are used before a method's declaration or implementation. When updating methods, update comments like this:

Method Tagging Example

```
/**
 *@brief              Brief method description
 *
 *                    Verbose description of method
 *
 *@param                     Parameter in the method's definition
 *
 *@return             Return value of method
 *@retval             Verbose explanation of return values
 */
int addNumbers(int x)
{
        int sum = 0;
    sum += x;
    return sum;
}
```

Variables

When updating variables, update comments like this:

Variable Short Hand Tag Example

```
int number;                /**< Comment about number*/
```

Enumerated Types

Enumerated types are tagged using the @enum. When updating enum types, update comments like this:

Method Tagging Example

```
/**
 *@brief          Brief enum description
 *
 *@enum           enum Name
 */
```

Miscellaneous

There are many tags you can use with HTML markup to create unique Doxygen documentation for a given file, class, method, or variable. The following are common tags that should be used when appropriate.

Informative Tags

```
/**
 *@note           A brief remark about the implementation to help clarify.
 *
 *@attention      An important remark that may cause code to break.
 *
 *@warning        An import remark that may depend on random conditions etc.
 *
 *@see            A reference to a class or a link to documentation (e.g. http://document.1a.com)
 */
```

Maintenance Tags

```
/**
 *@bug            A remark about a known bug in the code.
 *
 *@todo           A remark of what needs to be done to fix issues or remaining work.
 *
 */
```

Format Font Tags

```
/**
 *@a              Formats following word in special font (used for hyperlinks)
 *
 *@b              Formats following word in bold
 *
 *@em             Formats following word in italic
 *
 *@c              Formats following word in monospaced typewriter font
 *
 */
```

Structed List Tags

```
/**
 * - bulleted list item1
 *  - sub bulleted item1
 *
 * - bulleted list item2
 *
 */
```

Numbered List

```
/**
 * -# numbered list item1
 * -# numbered list item2
 *
 */
```

Displaying Code

```
/**
 *@code
    i++;
 *@endcode
 */
```

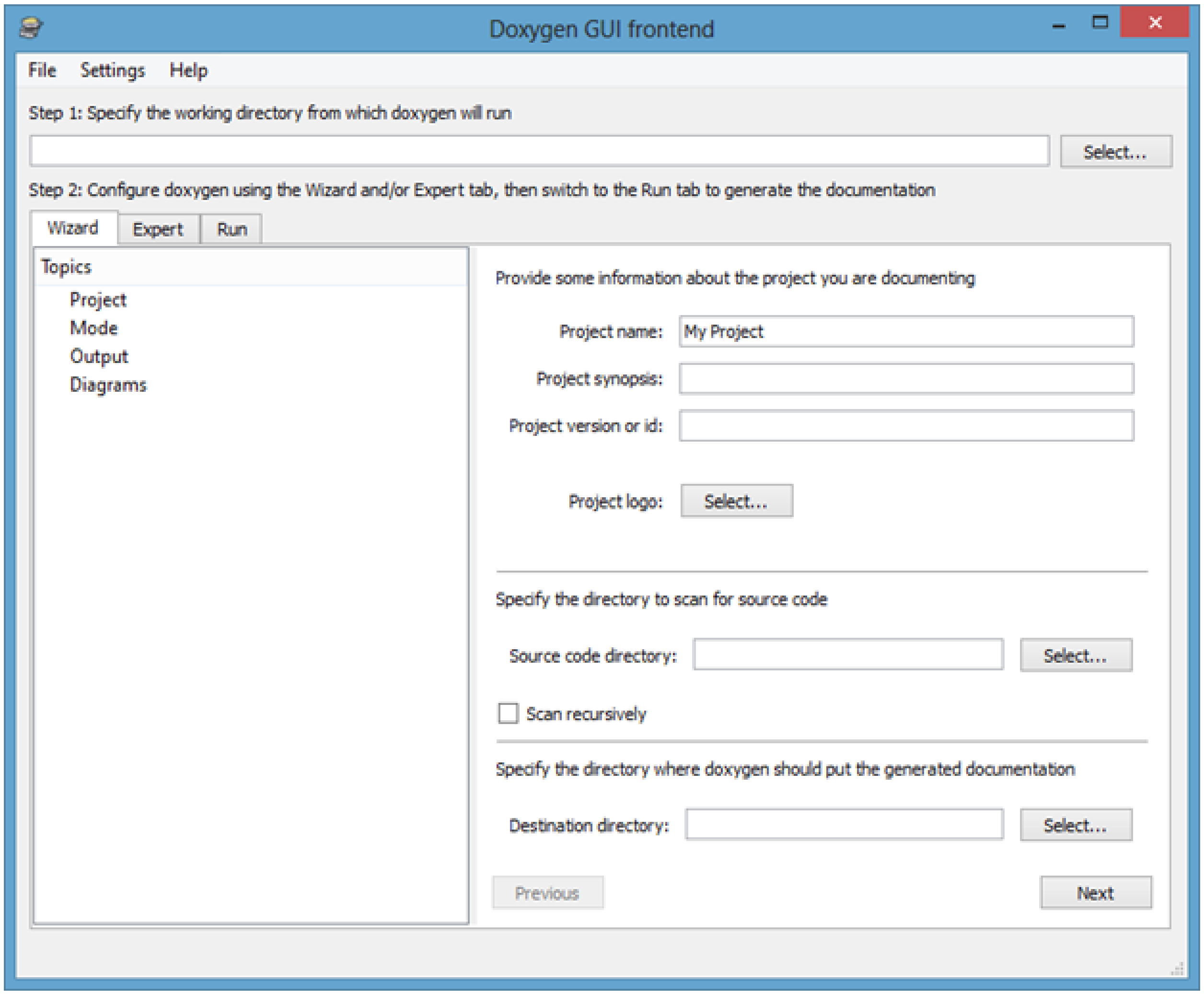
Setting up Doxygen Environment on Windows

Before generating Doxygen documentation, make sure to have the following:

Doxygen: http://www.stack.nl/~dimitri/doxygen/download.html (Contains Doxywizard)

Graphviz: http://www.graphviz.org/ (Click the Download link on the left side of the page)

- Navigate to the DoxyWizard (comes with Doxygen setup) application and configure it:



Doxygen GUI frontend

File Settings Help

Step 1: Specify the working directory from which doxygen will run

Step 2: Configure doxygen using the Wizard and/or Expert tab, then switch to the Run tab to generate the documentation

Provide some information about the project you are documenting

Project name: My Project

Project synopsis:

Project version or id:

Project logo: Select...

Specify the directory to scan for source code

Source code directory:

☐ Scan recursively

Specify the directory where doxygen should put the generated documentation

Destination directory: