Full guide: <https://www.doxygen.nl/manual/index.html>

# Doxygen Format

## File Header

Put at the top of the file.

/\*\*

 \* @file  doxygen.h

 \* @brief  File containing example of doxygen usage for quick reference.

 \* @author  My Self

 \* @date  9 Sep 2012

\* @copyright GNU Public License

 \*/

## Function

/\*\*

 \* @brief  Brief description of what the function does

 \*

 \* Details into how it does

 \* @param  param1  [in] Description of the first parameter of the function.

 \* @param  param2  [in] The second one, which follows @p param1.

 \* @return  Describe what the function returns.

 \* @see  http://website/

 \* @code

 \* Sometimes it is also convenient to include an example of usage:

 \* int out = func(param1, param2);

 \* printf("something...\n");

 \* @note  Something to note.

 \* @warning  Something to warn.

 \*/

int BoxStruct func(int param1, int param2 /\*, ...\*/);

## Struct

/\*\*

 \* @brief Use brief, otherwise the index won't have a brief explanation.

 \*

 \* Detailed explanation.

 \*/

typedef struct \_BoxStruct {

  int a;    /\*\*< Some documentation for the member BoxStruct#a. \*/

  int b;    /\*\*< Some documentation for the member BoxStruct#b. \*/

  double c; /\*\*< Etc. \*/

} BoxStruct;

## Enum

/\*\*

 \* @brief Use brief, otherwise the index won't have a brief explanation.

 \*

 \* Detailed explanation.

 \*/

typedef enum \_BoxEnum {

  BOXENUM\_FIRST,  /\*\*< Some documentation for first. \*/

  BOXENUM\_SECOND, /\*\*< Some documentation for second. \*/

  BOXENUM\_ETC     /\*\*< Etc. \*/

} BoxEnum;

## Variable

Various ways:

int var; /\*\*< Detailed description after the member \*/

int var; /\*!< Detailed description after the member \*/

int var; ///< Detailed description after the member

         //!<

int var; //!< Detailed description after the member

         //!<

int var; //!< Brief description after the member

int var; ///< Brief description after the member

/\*\* Brief description after the member (must start with 2 stars) \*/

int var;

## Preprocessor

Same as variables

# Grouping

Each source code file has a group to contain all its implementations (global variables, functions, enumerations, structures, etc.).

/\*\*

\* @addtogroup <name> [(title)]

\* @{

\*/

// Code of global variable, functions, enum, structure, etc.

/\*\*

\* @}

\*/ /\* End of group \*/

**Example:**



# Tips

## Working with Preprocessors

**Way #1: Apply for all preprocessors:**

# In the Doxygen configuration file

ENABLE\_PREPROCESSING = YES

# Or

# ENABLE\_PREPROCESSING = NO

* YES 🡪 If a preprocessor is not defined (#ifdef … is false), the document for members of that preprocessor will not be generated.
* NO 🡪 Don't care about the preprocessor. So all members will be generated.

However, graphs (generated by Graphiz) will not be outputted.

**Way #2: Apply for some specific preprocessors:**

# In the Doxygen configuration file

ENABLE\_PREPROCESSING = YES

PREDEFINED = <name-of-preprocessor>

Now if your code is like:

#if defined ISA

    int a;  ///< a is ...

#endif

And you define:

# In the Doxygen configuration file

ENABLE\_PREPROCESSING = YES

PREDEFINED = ISA

Then document for variable a will be generated.