Export Software Design from Doxygen to DOORS using XML

Table of contents

[1 Overview 2](#_Toc413769488)

[2 Create doxygen-compatible source documentation 3](#_Toc413769489)

[2.1 File header documentation 3](#_Toc413769490)

[2.2 Function header documentation in source file 3](#_Toc413769491)

[2.3 Class documentation in header file 4](#_Toc413769492)

[3 Installation 4](#_Toc413769493)

[3.1 Pre-installation steps 4](#_Toc413769494)

[3.2 Installing the tool chain 5](#_Toc413769495)

[4 Run the tool chain 5](#_Toc413769496)

[4.1 Create the doxygen report and convert it to plain text 5](#_Toc413769497)

[5 Import generated plain text documentation into DOORS 9](#_Toc413769498)

[6 Import documentation step by step 11](#_Toc413769499)

[7 FAQ 12](#_Toc413769500)

# 1 Overview

Source code -> Doxygen/XML -> XML/DOM parser(python) -> plain text -> DOORS

Intension: import the documentation of all functions and classes to DOORS.

**The steps:**

1. Create doxygen-compatible source documentation (chapter 2)
2. Install doxygen (chapter 3.1)
3. Install the tool chain, the python script (chapter 3.2)
4. Configure doxygen configuration file to create XML output (chapter 3.2)
5. Run doxygen and the python script  
   You can run the tool chain *alternatively*
   1. from Visual Studio if available
   2. from command line
   3. Scons
6. Import plain text file into DOORS

**How the python script works:**

The python script reads the doxygen-generated doxygen/xml/index.xml file and walks through all referenced \*.c/\*.cpp files, C++ classes and structs (that contains function definitions). EVE code, \*.k files, is supported since convert\_xml.py version 1.5.

A structured plain text will be generated that contains a short description of each implementation file, class and structures.

**What will be parsed:**  
- DESCRIPTION flag in file header  
- @brief description for functions and classes

Source documentation is required in  
- \*.c, \*.cpp files for global or static functions  
- in header files for C++ classes  
- \*.k files since, EVE code (functions) since convert\_xml.py version 1.5.

# 2 Create doxygen-compatible source documentation

## 2.1 File header documentation

/\*! \file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

COMPANY: Continental Temic, DE/DC, A.D.C. GmbH

PROJECT: CSF200

CPU: general

COMPONENT: cam

MODULNAME: cam\_rot.c

**DESCRIPTION: Routines for full camera rotation and translation**

AUTHOR: Stefan Heinrich

CREATION DATE: July 1997

VERSION: $Revision: 1.4 $

CHANGES:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

## 2.2 Function header documentation in source file

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Functionname: IntLUTSqrt() \*/ /\*!

**@brief integer square root substitution**

@description A faster replacement for (int)(java.lang.Math.sqrt(x)). Completely accurate for x < 2147483648 (i.e. 2^31)...

@param[in] i32\_t x: input value

@return squareroot of input value

@pre -

@post -

@author ???

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

i32\_t IntLUTSqrt(i32\_t x)

{}

**OR**

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Functionname : COD\_CalcHeadwayYWorld \*/ /\*!

**Description : calculate Y coordinate of left and right headway virtual marker at given X-position**

@param[in] - i32\_t xw: X- coordinate of markers [cm]

@param[out] - i32\_t \*pYWorldLeft: Y - coordinate of left marker [cm]

@param[out] - i32\_t \*pYWorldRight: Y - coordinate of right marker [cm]

@return -

@pre -

@post -

@author Timo Seifert

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

Is valid since **convert\_xml.py v1.9**. Please enable AUTOBRIEF in the doxygen config file (see chapter 3.2).

## 2.3 Class documentation in header file

/\*\*

\* **@brief Main application class**

\*/

class C\_PFCAlgo  
{

public:

/\*\*

**\* @brief default constructor**

\*/

C\_PFCAlgo();

}

# 3 Installation

## 3.1 Pre-installation steps

* Doxygen   
  Link from MKS:  
  REPOSITORY/Base\_Development/05\_Algorithm/STP\_SharedToolProjects/04\_Engineering/doxygen (checkpoint 1.15) to your project 04\_Engineering/02\_Development\_Tools/doxygen
* Doxygen directory structure, report generation  
  If you don’t have already a doxygen setup in 04\_Engineering\03\_Workspace\algo\xxx\_sim\sim\_swc\_xxx, copy or create a MKS link:  
  04\_Engineering\03\_Workspace\algo\xxx\_sim\sim\_swc\_xxx\doxygen  
  from  
  Base\_Development/05\_Algorithm/ETK\_EngineeringToolkit/04\_Engineering/SET\_SimExtensions/03\_Design/01\_Supporting\_Documents/conti\_doxygen\_style\_sheet/ (checkpoint AL\_ETK\_SET\_01.06.11, version 1.26 or newer)
* Python 2.7   
  Link from MKS  
  REPOSITORY/Base\_Development/05\_Algorithm/STP\_SharedToolProjects/04\_Engineering/movpy (checkpoint 1.4) to your project 04\_Engineering/02\_Development\_Tools/movpy
* Python module [xml.dom.minidom](https://wiki.python.org/moin/MiniDom)
* DOORS client

## 3.2 Installing the tool chain

If not already done, install the following files to:

|  |  |
| --- | --- |
| convert\_xml.py | Insert and MKS link from  REPOSITORY/Base\_Development/05\_Algorithm/VCL\_Vehicle\_Detection/04\_Engineering/03\_Workspace/algo/vcl\_sim/sim\_swc\_vcl/comvert\_xml.py  to  <project>\04\_Engineering\03\_Workspace\algo\xxx\_sim\sim\_swc\_xxx\convert\_xml.py |
| make\_sim\_swc\_doors.bat | Insert and MKS link from  REPOSITORY/Base\_Development/05\_Algorithm/VCL\_Vehicle\_Detection/04\_Engineering/03\_Workspace/algo/vcl\_sim/sim\_swc\_vcl/make\_sim\_swc\_doors.bat  to  <project>\04\_Engineering\03\_Workspace\algo\xxx\_sim\sim\_swc\_xxx\ make\_sim\_swc\_doors.bat |
| sim\_swc\_doxy\_setup.txt | 04\_Engineering\03\_Workspace\algo\xxx\_sim\sim\_swc\_xxx\ sim\_swc\_doxy\_setup.txt If you already have a doxygen file, just insert the settings. GENERATE\_XML = YES XML\_OUTPUT = xml XML\_SCHEMA = XML\_DTD = XML\_PROGRAMLISTING = NO HIDE\_UNDOC\_MEMBERS = NO HIDE\_UNDOC\_CLASSES = NO JAVADOC\_AUTOBRIEF = YES QT\_AUTOBRIEF = YES  INPUT = <your component’s source directories> |

Where xxx is the abbreviation of your component.

# 4 Run the tool chain

You can run the tool chain either manual, from your component’s Visual Studio simulation project 04\_Engineering\03\_Workspace\algo\xxx\_sim\sim\_swc\_xxx\sim\_swc\_xxx.vcproj, or SCons build (TODO)

## 4.1 Create the doxygen report and convert it to plain text

###### 4.1.1 Run manual

>cd 04\_Engineering\03\_Workspace\algo\xxx\_sim\sim\_swc\_xxx  
>.\doxygen\make\_sim\_swc\_doxy.bat  
>.\make\_sim\_swc\_doors.bat

###### 4.1.2 Run from Visual Studio project

Create an additional custom-build release and insert the doxygen calls into it.  
Prebuild-Event->Command line: .\doxygen\make\_sim\_swc\_doxy.bat  
Postbuild-Event->Command line: .\make\_sim\_swc\_doors.bat

<Configuration  
 Name="Doxygen|Win32"  
 OutputDirectory="$(SolutionDir)$(ConfigurationName)"  
 IntermediateDirectory="$(ConfigurationName)"  
 ConfigurationType="10"  
 >  
 <Tool  
 Name="VCPreBuildEventTool"  
 Description="Call Doxygen"  
 CommandLine=".\doxygen\make\_sim\_swc\_doxy.bat"  
 />  
 <Tool  
 Name="VCCustomBuildTool"  
 />  
 <Tool  
 Name="VCMIDLTool"  
 />  
 <Tool  
 Name="VCPostBuildEventTool"  
 Description="Convert XML to plain text for DOORS import"  
 CommandLine=".\make\_sim\_swc\_doors.bat"  
 />  
 </Configuration>

###### 4.1.3 SCons build

Setup the doxygen build

**Enable doxygen build**

Usually sim\_swc\_xxx project generates also doxygen html documentation.

For using doxygen, the extension must be added to the environment and the location of doxygen tools (in 04\_Engineering\03\_Workspace\algo\xxx\_sim\SConscript.py)

pc\_env = Environment(# list of scons extensions used

tools = ["msvc", "msvs-patched", "mslib", "mslink",

"msvc-addon", "**doxygen**", "fingerprint"],

# path to scons extensions

toolpath = [scons\_adas\_extensions\_path],

MSVS\_USE\_MFC\_DIRS = 1,

TARGET\_ARCH = "X86",

# path to doxygen tool

**DOXYGEN\_TOOLS = doxygen\_dir,**

# manifest files need to be included in the dlls/apps

WINDOWS\_EMBED\_MANIFEST = True,

# use Visual Studio 2005

MSVC\_VERSION = "8.0",

# specify processor for scons build

BUILD\_TARGET = "SIM")

Then, we inform scons where to look for the doxygen config file (in 03\_Workspace\algo\xxx\_sim\sim\_swc\_xxx\**simenv\_config.scfg**)

# determine if doxygen is built

generate\_doxygen = True

# doxygen config file name

doxy\_file\_name = "sim\_swc\_doxy\_setup.txt"

# target name used by scons doxygen. The complete name will be "{component\_name}\_{scons\_doxygen\_target\_name}", for example "sib\_doxy"

scons\_doxygen\_target\_name = "doxy"

The documentation is created in 04\_Engineering\03\_Workspace\algo\xxx\_sim\sim\_swc\_xxx\doxygen\ folder by calling Doxygen builder (04\_Engineering\03\_Workspace\algo\xxx\_sim\sim\_swc\_xxx\**SConscript\_simenv.py**)

This also copies conti.png file needed by html files to the html folder because when clean is executed, html folder is deleted.

if generate\_doxygen:

vars()[component\_name + "\_" + scons\_doxygen\_target\_name] = env.Doxygen(doxy\_file\_name)

# restore image

env.AddPostAction(vars()[component\_name + "\_" + scons\_doxygen\_target\_name],   
 Copy(Dir(os.path.join("doxygen", "html")).abspath, File(os.path.join("doxygen", "style",

"conti.png")).abspath))  
 **convert\_xml\_to\_doors = "cd ..\\" + component\_name + "\_sim\sim\_swc\_" + component\_name + "\ && ..\..\..\..\\02\_Development\_Tools\movpy\movpy.exe ./convert\_xml.py ./doxygen/xml ./doxygen/doors.txt && cd ../../" + component\_name**

**env.AddPostAction(vars()[component\_name + "\_" + scons\_doxygen\_target\_name], Action(convert\_xml\_to\_doors))**

Alias(component\_name + "\_" + scons\_doxygen\_target\_name, vars()[component\_name + "\_" +   
 scons\_doxygen\_target\_name])

**Install the python script**

Install the python script: put **convert\_xml.py** under your 02\_Development\_Tools /scons\_tools/scons\_common\_scripts/**doxygen/** *(to create)*

Add in your local 03\_Workspace\algo\XXX\**sconscript\_setup\_config.scfg** the green line to copy your python script to the correct place :

{

"name" : "sim\_swc",

"copy" : True,

"dest\_folder" : "04\_Engineering/03\_Workspace/algo/" + component\_name + "\_sim/sim\_swc\_" +

component\_name,

"source" : ["sim/sim\_env/SConscript\_simenv.py",

"sim/sim\_swc/SConscript.py",

**"doxygen/convert\_xml.py"]**

},

**Enable Doxygen XML generation**

Python script 04\_Engineering\02\_Development\_Tools\scons\_tools\scons\_adas\_extensions\**doxygen.py** is used to generate doxygen html documentation. Enable XML generation

*def DoxyEmitter(source, target, env):*

"""Doxygen Doxyfile emitter"""

# possible output formats and their default values and output locations

output\_formats = {

"HTML": ("YES", "html"),

"LATEX": ("NO", "latex"),

"RTF": ("NO", "rtf"),

"MAN": ("NO", "man"),

"XML": (**"YES**", "xml"),

}

Run doygen build

O4\_Engineering\03\_Workspace\algo\xxx\scons xxx\_doxy

# 5 Import generated plain text documentation into DOORS

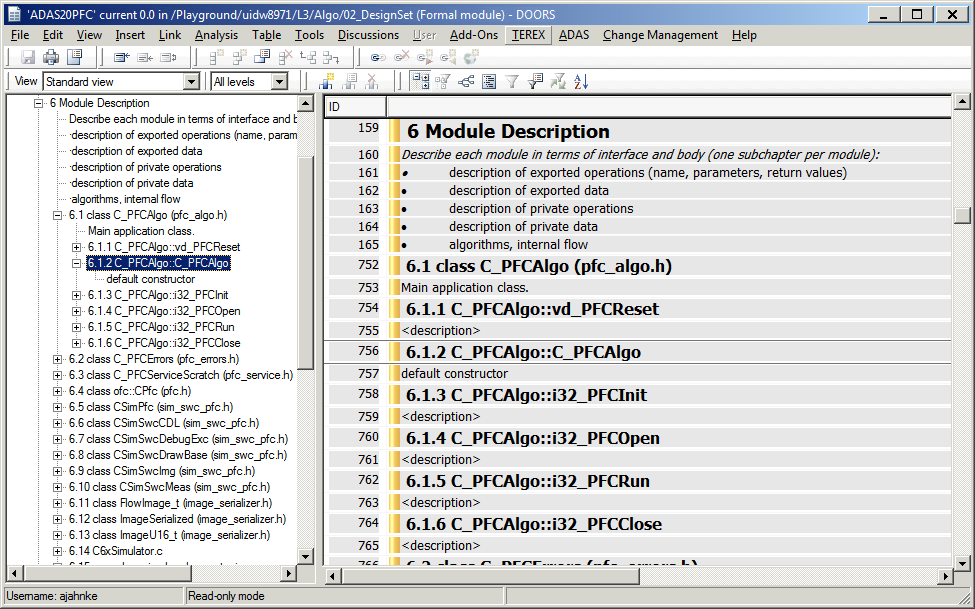
Open a DOORS client or the Citrix Web interface <https://citrix-web.conti.de/>

1. Create a new formal module  
   if non-existent. Use the ADAS 2.0 Template (ADAS/99\_templates/Templates/ADAS V2.0/L3/Algo/02\_DesignSet/L3\_ALGO\_Design\_Spec\_template, [doors://rbgs854a:40000/?version=2&prodID=0&urn=urn:telelogic::1-503e822e5ec3651e-M-0003c283](https://cws1.conti.de/content/00012669/Team%20Documents/03_General/06_Process/01_ModuleTests/Wiki/Module%20Tests%20-%20L3-Component-Design-Specification.pptx)). Right-Click onto L3\_ALGO\_Design\_Spec\_template and select Copy from context menu

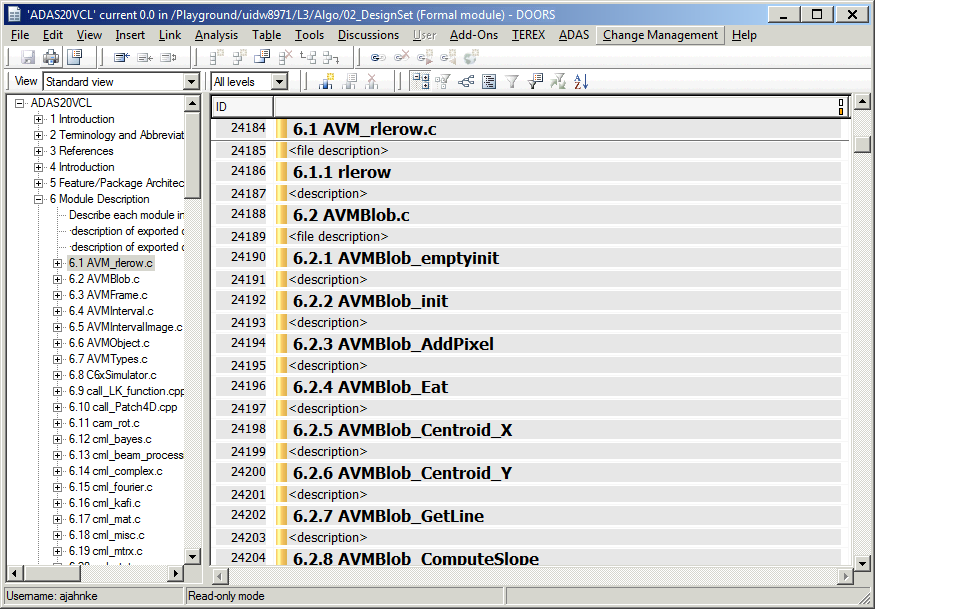
Navigate to the location/folder where you want to create the formal module. From context menu (right-click), choose “Paste Special” and rename the document.

1. Prepare the new formal module  
   Open the new document for “Exclusive Edit”. Delete section 6.1.
2. Import plain text file  
   Select row “6 Module Description”.  
   From menu, choose “File → Import → Plain Text”. Select the file from local file system and click button “Import…”.  
   In dialog “Configure import – DOORS”, set “Decomposition level” to “Paragraph”. Keep the other settings. Click button “Import”.

A C++ will look like this:



C project:



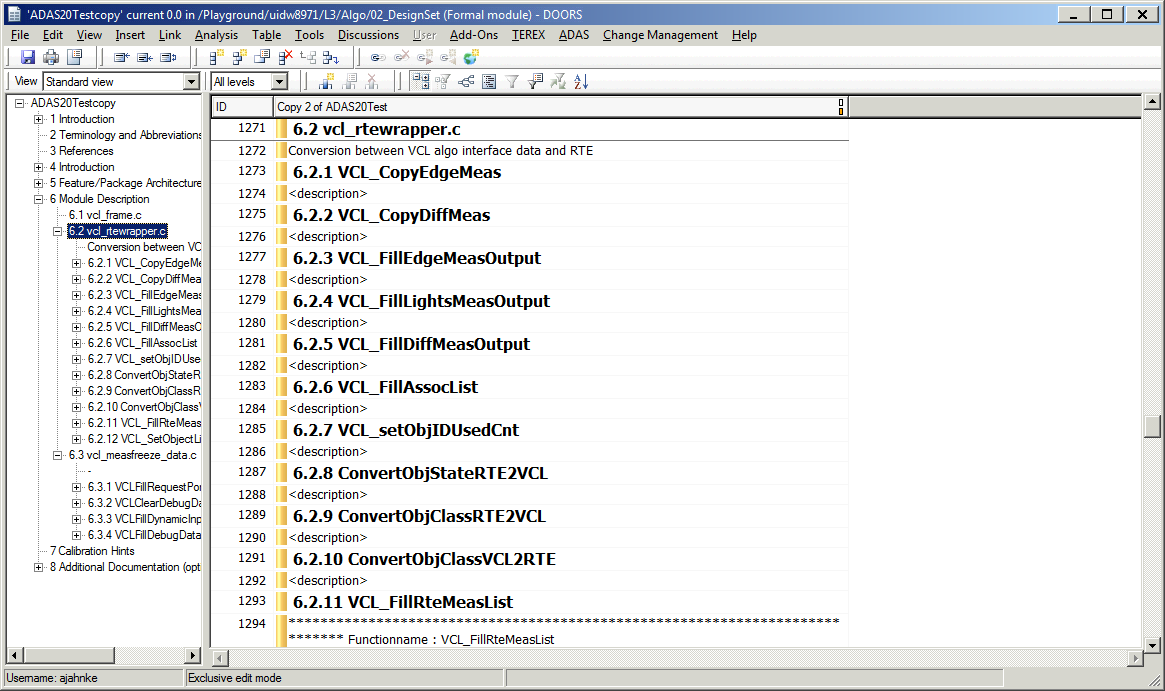
# 6 Import documentation step by step

It is possible to import the source documentation file-wise into DOORS.

The steps:

* Configure sim\_swc\_doxy\_setup.txt to only parse the desired files:  
  INPUT = .\..\..\..\..\01\_Source\_Code\algo\vcl\frame\vcl\_frame.c \  
   .\..\..\..\..\01\_Source\_Code\algo\vcl\frame\vcl\_rtewrapper.c
* Run the tool chain (see ch. 4) and import the result to DOORS (see ch. 5)
* Configure sim\_swc\_doxy\_setup.txt to only parse the second chunk of files:  
  INPUT = .\..\..\..\..\01\_Source\_Code\algo\vcl\frame\vcl\_measfreeze\_data.c
* Run the tool chain (see ch. 4)
* Open the DOORS formal module for editing and place the cursor into the last row of the previous import.
* Import the formal module there. DOORS will insert the new content at the right position and will update the numbering correctly.

The screenshot:



# 7 FAQ

**F: convert\_xml.py prints error message concerning XML file:**

./doxygen/xml/class\_s\_a\_c\_1\_1\_depth.xml

exception reading file ./doxygen/xml/class\_s\_a\_c\_1\_1\_depth.xml

(<class 'xml.parsers.expat.ExpatError'>, ExpatError('not well-formed (invalid token): line 224, column 38',), <traceback object at 0x03AEA1E8>)

<traceback object at 0x03AEA1E8>

*Answer:* there are unprotected characters or syntax errors in the doxygen-generated XML. Remove the syntax errors by hand and run the script again.

**F: How to enable conditional compiled code/ macros?**

In your doxygen configuration file, sim\_swc\_doxy\_setup.txt:

ENABLE\_PREPROCESSING = YES  
EXPAND\_ONLY\_PREDEF = YES  
PREDEFINED = DOXYGEN \  
 *MY\_MACRO*

**F: How to process EVE code?**

EVE code (\*.k files) are supported since convert\_xml.py v1.5.

To add them to the doxygen report, change the FILE\_PATTERNS flag in you doxygen configuration file.

FILE\_PATTERNS = \*.c \  
 \*.cc \  
 \*.cxx \  
 \*.cpp \  
 \*.c++ \  
 \*.java \  
 \*.ii \  
 \*.ixx \  
 \*.ipp \  
 \*.i++ \  
 \*.inl \  
 \*.h \  
 \*.he \  
 \*.hh \  
 \*.hxx \  
 \*.hpp \  
 \*.h++ \  
 \*.idl \  
 \*.odl \  
 \*.php \  
 \*.php3 \  
 \*.inc \  
 \*.m \  
 \*.mm \  
 \*.dox \  
 **\*.k**

**F: How to avoid duplicates when re-importing?**

A re-import creates duplicates. No automatically way to avoid this. If you want to update a few entries, do it manually.  
Note, you can create the DOORS document file-wise, see chapter 6.

**F: How to skip function forward declarations (extern functions)?**

Sometimes, at the beginning of a source file there might be some *extern* function forward declarations of functions defined in other files. By default, they are inserted into the doxygen reports and will be processed by the tool chain.  
But there’s a possibility to put some doxygen tags around them to mark them as skipped:

/\*!

\* \cond DOXY\_FORWARD\_DEFINITION

\*/

extern void PredictTemplatePos(CameraObject\_t \*pCamObj, i16\_t \*TrackRect);

extern i16\_t CheckVehicleHyp(SubImage\_t \*pObjRect, VCLObjectClass\_e HypType, i32\_t BottomMeasured);

/\*!

\* \endcond

\*/

It is a kind of conditional compilation similar to *#ifdef DOXY\_FORWARD\_DEFINITION* and the condition is not fulfilled. You can choose the name of the condition on your own, see doxygen documentation <http://www.stack.nl/~dimitri/doxygen/manual/commands.html#cmdcond>.