

# Documentation of Analysis Code Analysis Procedures and more.

Benedikt Zihlmann

December 5, 2021

# Documentation options

- Documentation is essential
- There are several "layers" of documentation
  - Basic code documentation: comments, doxygen
  - Basic analysis steps documentation: pdf, ...
  - Specific "package" documentations: pdf, ... (ReactionFilter, DSelector, AmpTools, ...)
  - How to pages, FAQ pages: wiki

# Current Status

## Wiki:

- At top level wiki page no link labeled "Documentation"
- All Documentation is burried at various levels of different links
- Hard to find if you do not already know where to look

→ Task1: Coherency, bring order into chaos.

## Topic Documentation:

DSelector <https://www.overleaf.com/project/5ca37385d3a0d471b3c1a430>

AmpTools <https://github.com/mashephe/AmpTools/wiki>  
[https://github.com/mashephe/AmpTools/blob/master/AmpTools\\_User\\_Guide.pdf](https://github.com/mashephe/AmpTools/blob/master/AmpTools_User_Guide.pdf)

Overview <https://www.overleaf.com/9649723516hgfwhsnwdjvw>

→ Task2: Updates, by authors AND users.

Code Documentation: doxygen (halld\_recon, halld\_sim)

→ Task3: Missing in most DANA plugins.

# What is doxygen

Doxygen is a "frame work" that geneates documentation in web/html form based on "marked commets" found in source code like C++ or python or ... .

Doxygen uses a configuration file to define its behavior to produce:

- A list of all code files, a page for each header and code file, navigable
- A list of all classes found, naviable. Each page represents one class and include class parameters, public (and private) member functions and variables.
- Behavior of doxygen is configurable with required config file

Example: doxygen MyConfigFile

## Doxygen Config file

Example of some configuration parameters:

- `OUTPUT_DIRECTORY = locationWhereToWriteHTML`
- `BRIEF_MEMBER_DESC = yes # precede detailed doc by brief command`
- `REPEAT_BRIEF = yes # prepend brief description`
- `AUTOLINK_SUPPORT = yes # identify known strings and generate links`
- `EXTRACT_PRIVATE = yes # include private members in documentation`
- `EXTRACT_LOCAL_CLASSES = yes # include locally defined classes`
- `RECURSIVE = yes # search subdirectories for code`
- `FILE_PATTERNS = # which files to scan for content`

and many, many, many, many more.....

# Example doxygen result

## Example of test: Overview page

The screenshot shows a Firefox web browser window. The address bar displays the URL `https://userweb.jlab.org/~zhimann/html/`. The browser's bookmark bar contains several entries, including 'Project free tv - 3k soor...', 'Virtual Regatta Instone...', 'GlasX', 'Chesapeake Bay Opera...', 'Project Free TV Stream...', 'SolarMovie - Watch M...', and '123Movies - Watch HD...'. The page title is 'My DOXY Test Libs'. Below the title, there is a navigation menu with 'Main Page', 'Modules', 'Classes', and 'Files'. The main content area is titled 'My DOXY Test Libs Documentation'. In the bottom right corner of the page, it says 'Generated by [doxygen](#) 1.8.11'. The browser's taskbar at the bottom shows two open windows: 'harpis (zhimann) - TigerVNC' and 'My DOXY Test Libs: Main Page --'.

# Example doxygen result

## Example of test: Class List (dir if inheritance)

The screenshot shows a web browser window displaying the Doxygen-generated Class List for a project named "My DOXY Test Libs". The browser's address bar shows the URL "https://userovob.jlab.org/~zhimann/html/annotated.html". The page title is "My DOXY Test Libs". The navigation tabs include "Main Page", "Modules", "Classes", and "Files". The "Classes" tab is active, and the "Class List" sub-tab is selected. The main content area is titled "Class List" and contains the text: "Here are the classes, structs, unions and interfaces with brief descriptions: (detail level 1.2.3)". Below this text is a list of classes, each with a small icon to its left. The classes listed are: DAnalysis, alist, async\_filebuf, caen1190confg, ccalcluster\_t, centroid\_t, cilist, cilist, cluster\_t, complex, DAnalysisResults, DAnalysisResults\_factory, DAnalysisUtilities, DAnalysisUtilities\_factory, DApplication, DBCALClump, DBCALClump\_factory, DBCALCluster, DBCALCluster\_factory, DBCALCluster\_factory\_SINGLE, DBCALDigiHit, DBCALGeometry, and DBCALGeometry\_factory. The browser's taskbar at the bottom shows the application "kargo5 (zhimann) - TigerVNC" and the current page "My DOXY Test Libs: Class List — M...".

# Example doxygen result

## Example of test: File List, in subdirectories

The screenshot shows a Firefox browser window displaying the 'File List' page for 'My DOXY Test Libs'. The page title is 'My DOXY Test Libs' and the URL is 'https://userweb.jlab.org/~zhimann/html/files.html'. The page content includes a search bar and a list of subdirectories. The subdirectories listed are: ANALYSIS, BCAL, CCAL, CDC, CERE, DANA, DAQ, DIRC, EVENTSTORE, FCAL, FDC, FHWPC, HDDH, HDGEOMETRY, Include, KINFITTER, PAIR\_SPECTROMETER, PID, RP, START\_COUNTER, TAC, TAGGER, and TOP. The page also indicates that there are 121 details for this list.

## Example of DCDCHit container class:

The screenshot shows a Firefox browser window displaying the 'DCDCHit' class page for 'My DOXY Test Libs'. The page title is 'My DOXY Test Libs' and the URL is 'https://userweb.jlab.org/~zhimann/html/classDCDCHit.html'. The page content includes a search bar and the start of the class documentation.



# Current doxygen GlueX page

## config file: "hald\_recon/src/doc/Doxyfile"

Applications Places Firefox Web Browser Tue Oct 26 08:56

File Edit View History Bookmarks Tools Help

My Drive - Google Drive CCVL\_EndOfYear\_Aw... Doxygen Manual - Grouping My DOXY Test Libs: C++ Hall-D Software Hall-D

Project free tv - 3k soor... Virtual Regatta Inshore... GlaxX Chesapeake Bay Opera... Project Free TV Stream... SolarMovie - Watch M... 123Movies - Watch HD... Other Bookmarks

## Hall-D Software alpha

Main Page Related Pages Modules Namespaces Classes Files Examples

### Hall-D Analysis Software

#### About

This documentation was generated automatically from the source code using the doxygen program. The content of this page is taken from the file `sim-recon/src/doc/mainpage.c++`. To view repository statistics, look here.

#### Introduction

The Hall-D reconstruction software is built upon the C++ JANA framework. The JANA framework was designed for Hall-D but is maintained separately as an independent project and contains no Hall-D specific source code. The Hall-D specific reconstruction software based on JANA is often referred to as DANA.

In broad terms, the JANA framework distinguishes between data classes and algorithm classes. Data classes generally have data members with no more than trivial methods defined. The algorithm classes are where the real work is done. These classes are called factories in JANA. The framework itself is responsible for passing pointers to the data objects between the factories that make them.

For example, a factory that makes calorimeter cluster objects does so by using calorimeter hit objects as inputs. The cluster making factory requests the hit objects from the JANA framework whose job is to locate them and return their pointers.

This design provides a loose coupling between the factory classes. The factory needing a type of data object as input doesn't need direct knowledge of the factory that actually generates those objects. Furthermore, multiple factory classes can exist that implement different algorithms, but deliver the same type of data objects. Which exact algorithm that is used can be specified by the user at run time rather than at compile time.

The naming convention used for classes is to have the factory class name be the name of the data class of the objects it provides, but with "\_factory" appended. For example, the algorithm that produces `DMCALCluster` objects would be named `DMCALCluster_factory`. If an alternate algorithm exists that produces the same type of data objects, it will be "tagged" and the tag appended to the end of the name (e.g. `DMCALCluster_factory_HOUGH`). All Hall-D specific classes start with the letter "D". (JANA classes start with "J").

Below is a thumbnail of the call graph between factories implemented in DANA. For practical reasons, only the names of the data classes are shown. Click on it to get a larger picture where individual classes can be clicked to jump to the corresponding factory page. In the image, the "first" request comes at the top from the `JvntProcessor` object asking for `DPhysicsEvent` objects. The `DPhysicsEvent` algorithm requires `DMTRicketSet` as (its only) input as shown by the arrows. Eventually, requests lead back to data from the input file which are the classes shown in green trapezoids at the bottom of the graph.

Generated on Fri Jun 26 2015 14:12:11 for Hall-D Software by doxygen 1.8.5

harpis@pikmanu - TigerVNC Hall-D Software: Hall-D Analysis S...

## Usage/consuming output of Doxygen

Doxygen is useful to explore the capabilities and dependences of code like classes, variables and their methods.

- What is this library about (this = TOF, BCAL, ...)?
- What does this class do?
- What data does this class use/handle/provide?
- What external parameters/data does this class need?
- Could also create man pages if desired.

If you are new, need to learn basic concepts first, you need basic how to documentation. (Doxygen is not useful)

- What is a plugin and how does it work? usage of MakePlugin, ect.
- How to analyze rest files? usage of MakeDSelector, etc.
- What are the capabilities of DSelector?
- How to pages are a good start.(find/search?)

## Conclusion (mine)

- Documentation is a two line project with 3 parts.
- Part One: Doxygen to document the actual code.
- Part Two: PDF/text to document(s) the "usage" of code, how to(s)
- Part Three: PDF/text to document(s) concepts of analyses.