Header Auto Test Design Doc

|  |  |  |
| --- | --- | --- |
| Version | Author | Comments |
| 0.1 | Andy Xu, Cammie Zhao | Realize function mainly |
| 0.2 | Andy Xu, Cammie Zhao | Still working on improvement |

Automatic test script as below:

git clone [\\10.193.108.11\shareserver\ChipModelVal\header\_auto](file:///\\10.193.108.11\shareserver\ChipModelVal\header_auto)

subfolder: \header\ header\_data

# Awards

This project needs thanks to many, especially: (forgive me if I miss any one)

1. Victor Cui who is the sponsor.
2. Andy Xu is responsible for reference manual doc analysis, he designs a script to collect necessary information of registers and create a data frame.
3. Cammie Zhao is responsible for header file data analysis, she designs a script to make registers information into a data frame.
4. In addition, thanks NPI design team and doc team, they provide many good ideas and advice.

# Pre-request installation

1. Python 3.7.x. 64 bits

<https://www.python.org/downloads/>

1. Python modules, you need to install some module to ensure the script can run, which just use pip command to install module.
   1. pandas
   2. numpy
   3. lxm
2. Java Version 8 Update 191

<https://www.java.com/zh_CN/>

1. Reference manual and header files, they have should be put in header\_data folder.

# 3 Knowledge required

1. Python
2. java
3. data frame

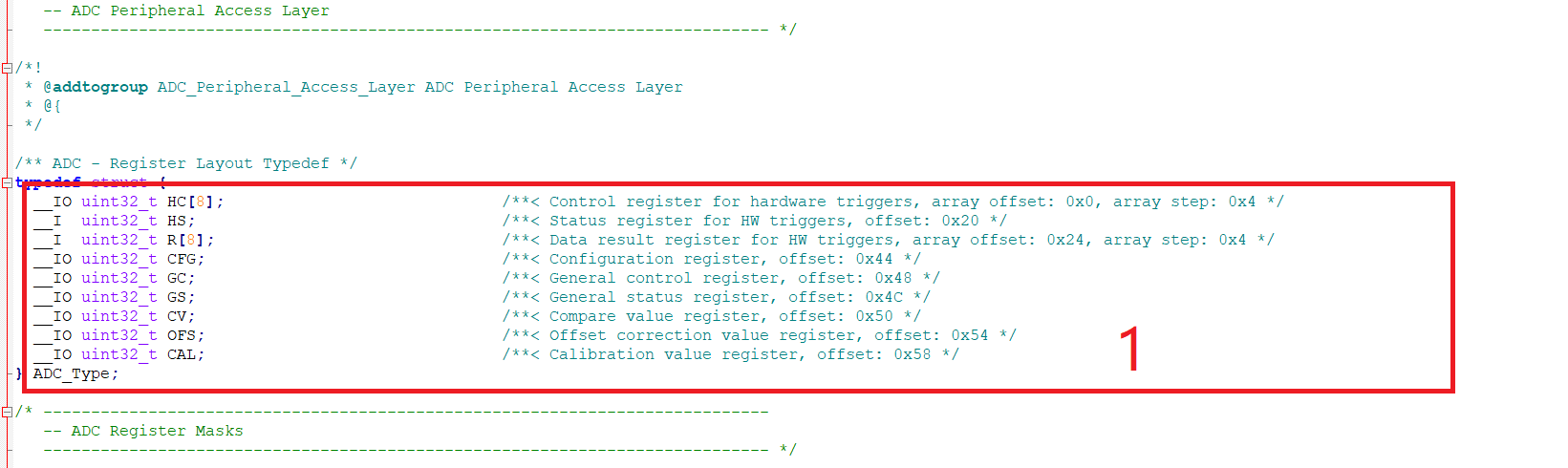
# 4 Feasibility analysis

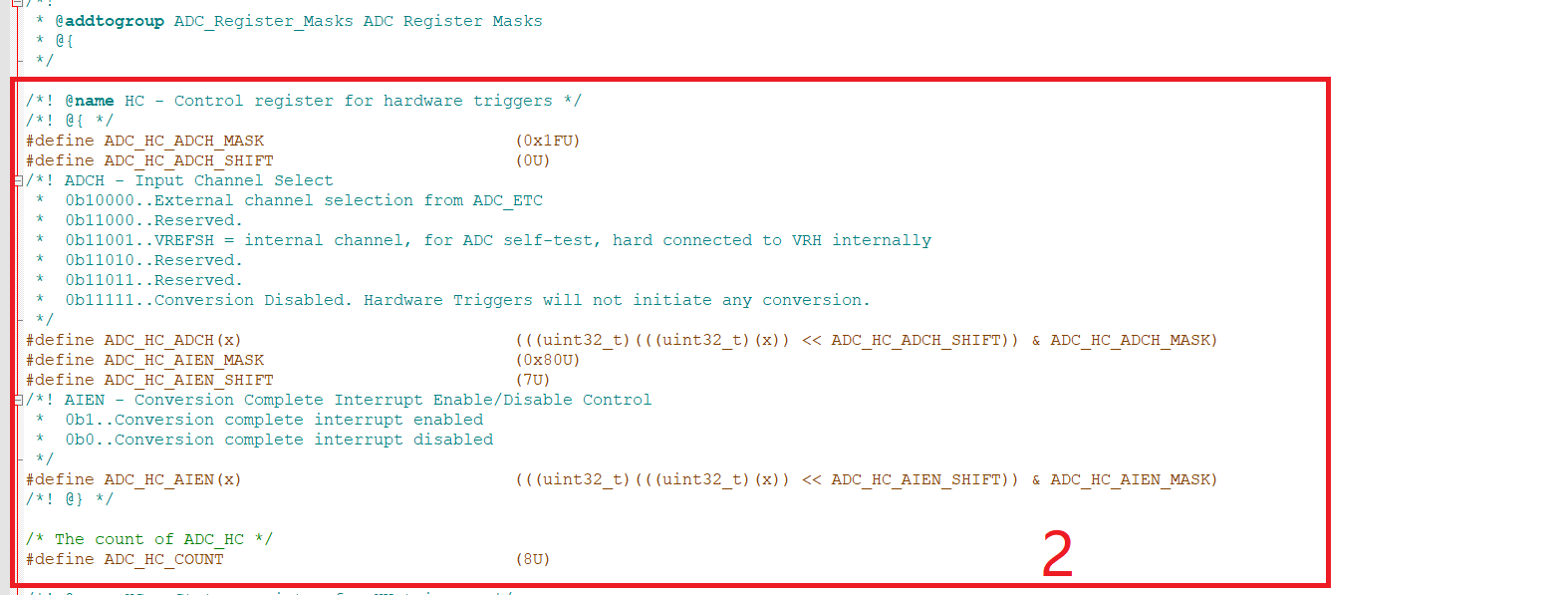
The data of header file is regular lines, and they include all information of registers, which is easily to create a data frame to cover necessary data.

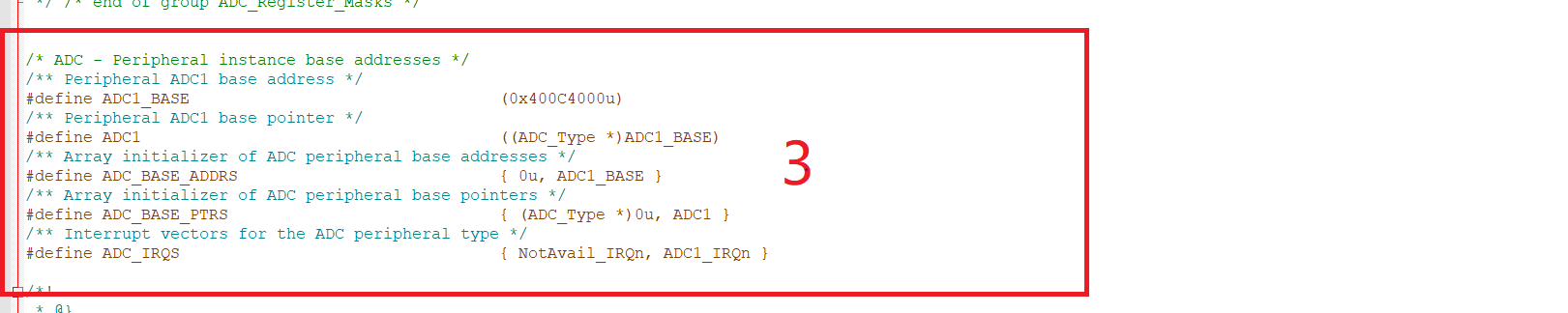
If we can extract the same data from reference manual and also create a data frame follow rules, then compare the two data frames, it’s the optimal direct test methodology to validate the header file data.

For instance, the following screen shorts show the relationship between the header file and the reference manual document, they all match.

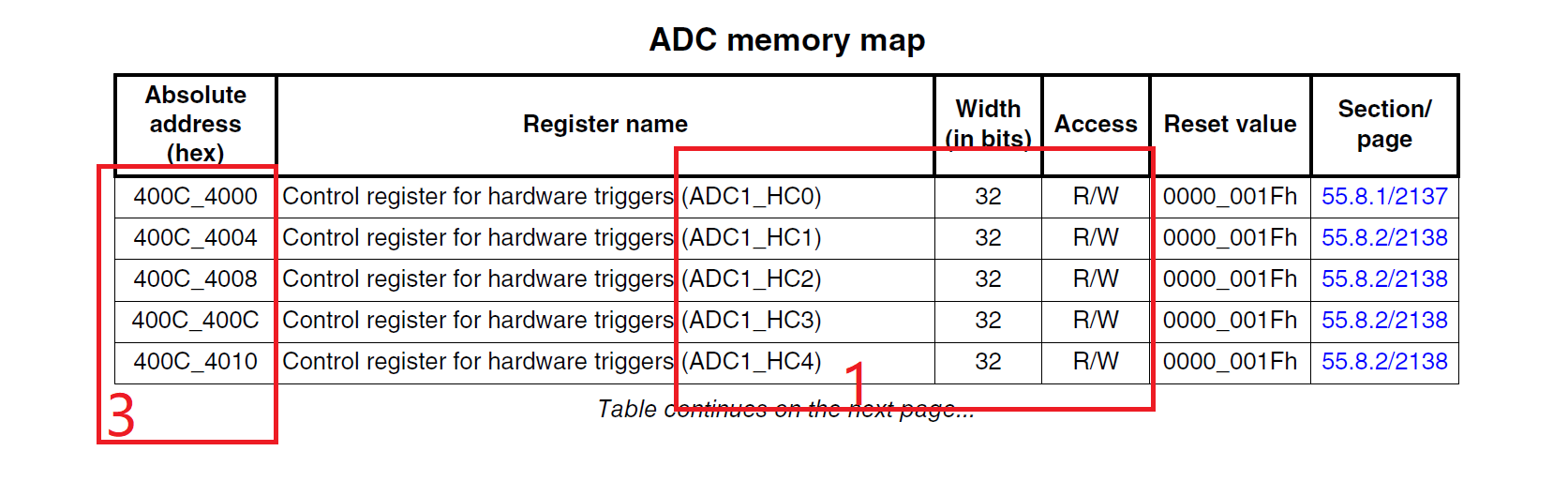
**Header file:**

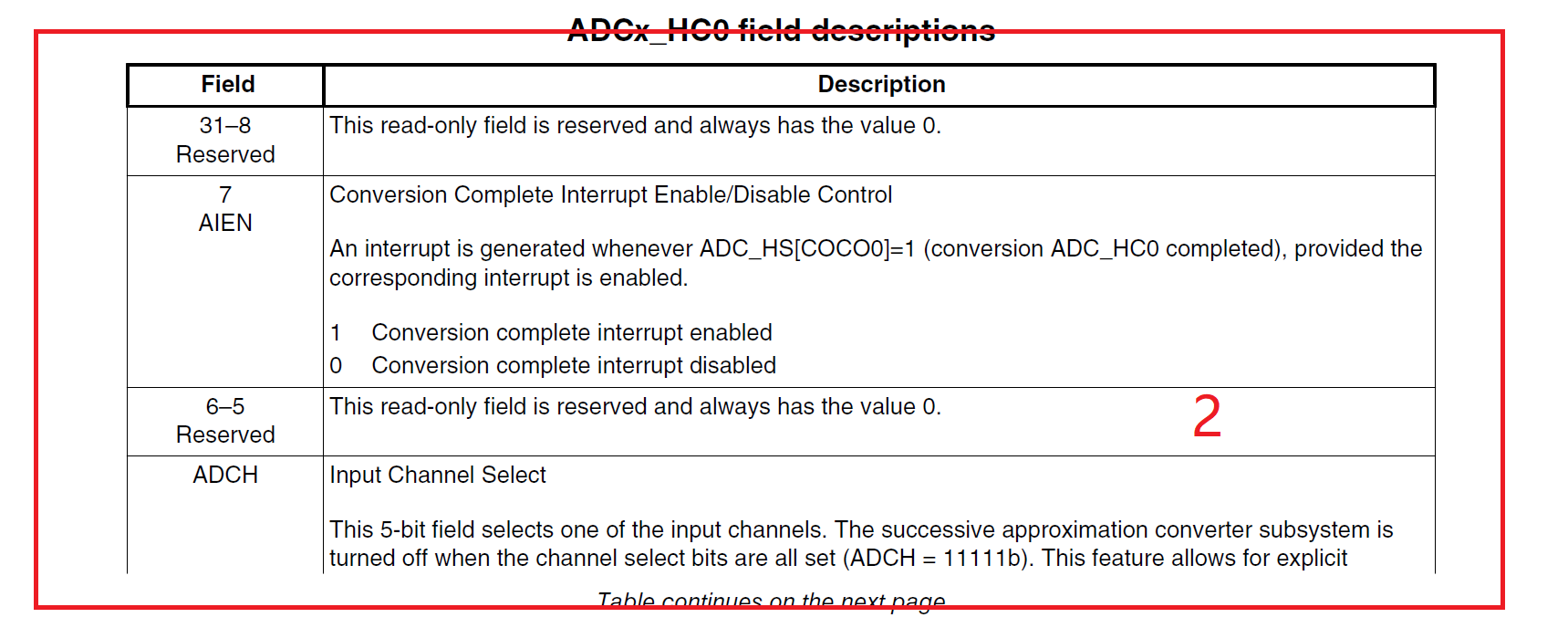






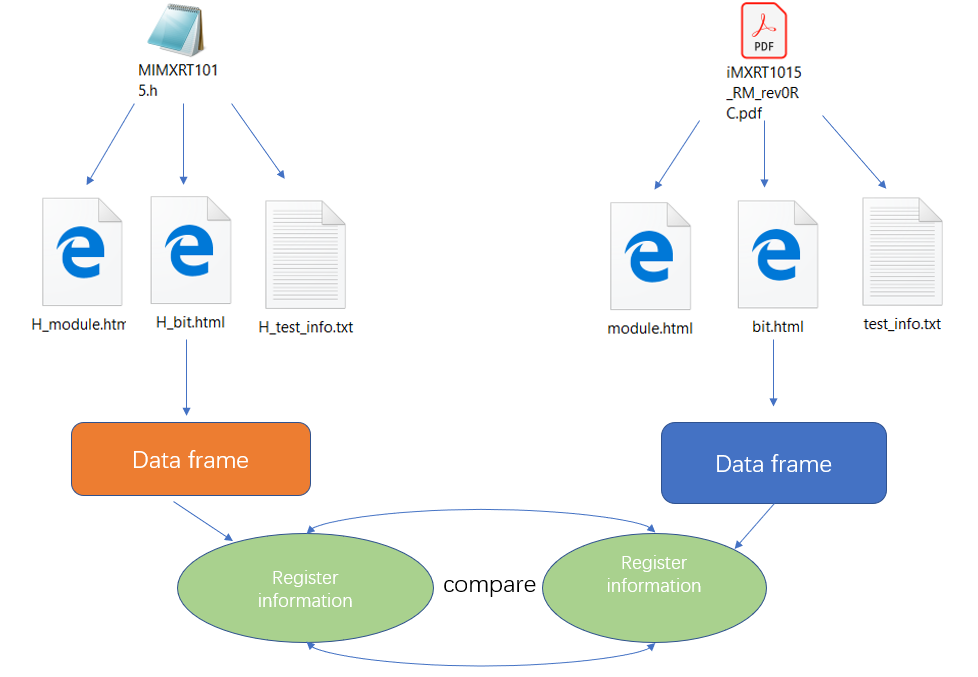
**Reference manual:**





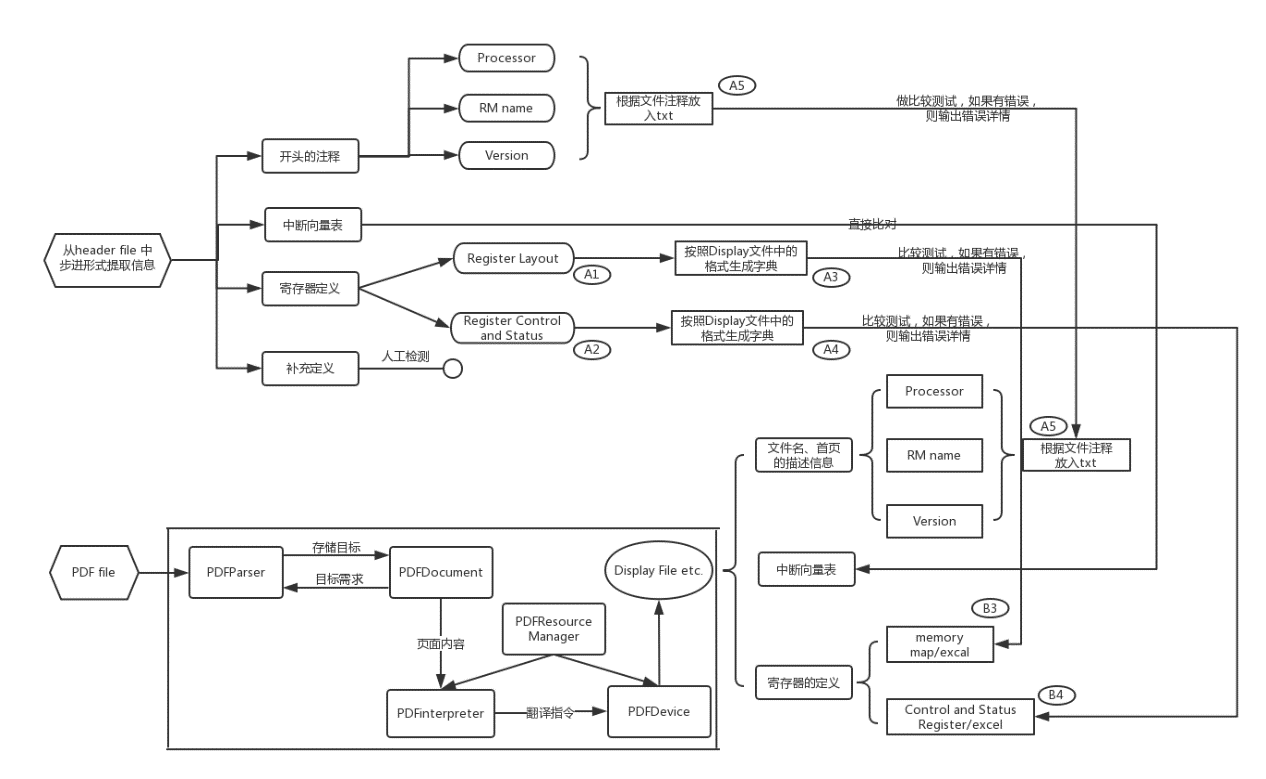
# 5 Auto test design architecture

To compare the data what we want easily, we design a model to use two sources create data frame, which can be compare more accurate. The simple flow is listed as below.



Here is the key steps description.

* parser the reference manual data
* The part will analyze PDF, then translate and save the content into a list
* extract necessary information
* It will find out the necessary information according to catalog and some rules we create
* create data frame of RM
* This part will create data frame about register information, bit field and page number.
* create data frame of header file
* It will create a same format data frame of header file.
* compare two data frames
* It will compare two sources of data, then classify errors by types, and write all error into error.yml.
* visual interface of automation script
* Have a visual interface of script, then when you use the script, you can just input header file name and PDF name, then use the button to get result.



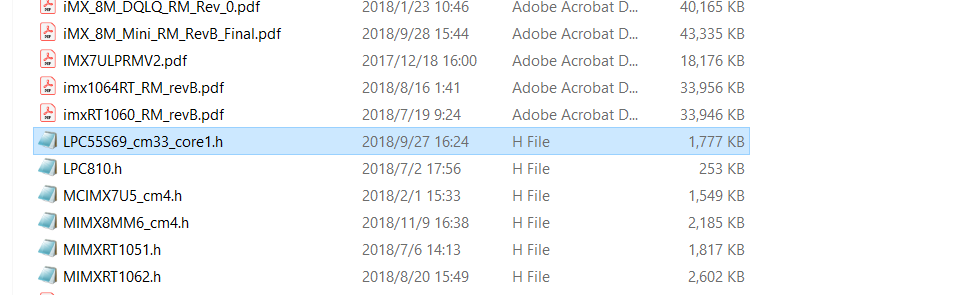
# 6 user guides

It’s very easy to use, just one argument is OK. More details, please refer to the following steps.

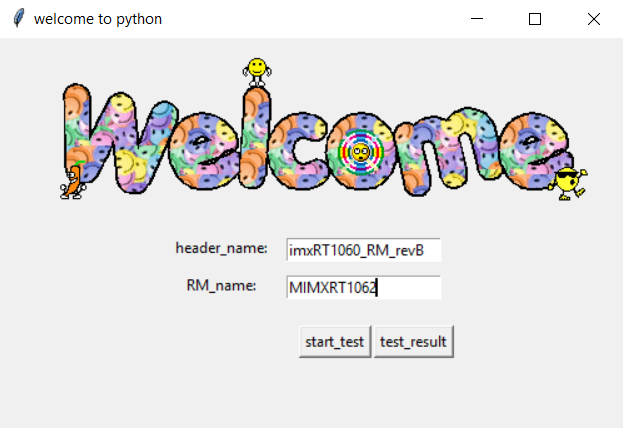
1. Get the auto test script

git clone <http://10.192.244.6:8080/cm_val_auto>

1. Copy header file and PDF into assign folder(header\_data)



1. Input header and PDF name into interface



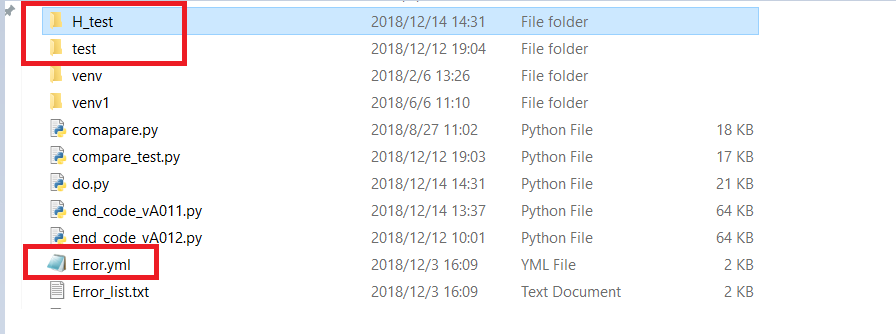
4.check the error.yml

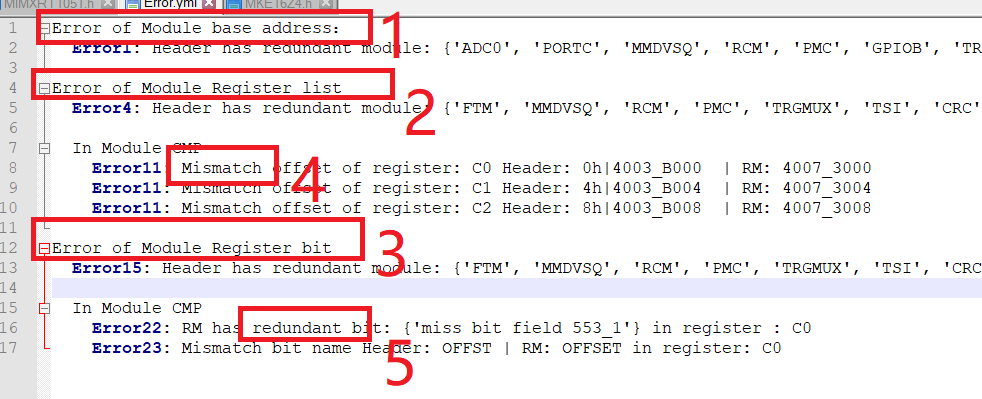
# 7 Test Report

The data of header file and RM will be saved in H\_test and test folder in case you want see the read data, the test result will be saved to a yml file in current folder, and it records the error and the place in RM and header file. There are three kinds of failures.

1. Mismatch: The header data does not match the reference manual data
2. Redundant: The header data has redundant info after comparing the reference manual data or the opposite.

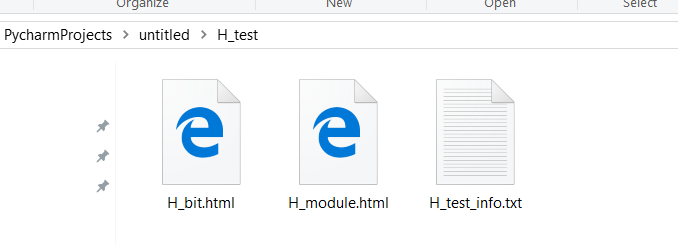
The Yaml supports to collapse the items to focus on the failures. Here is an example of test report.

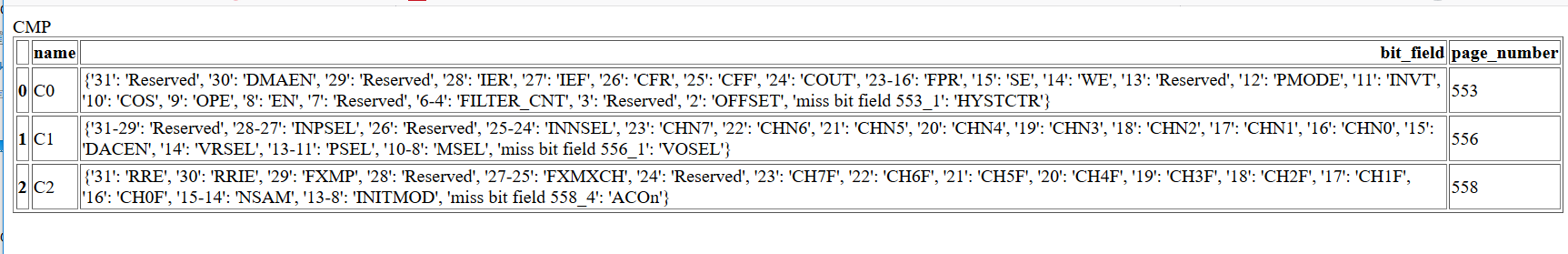


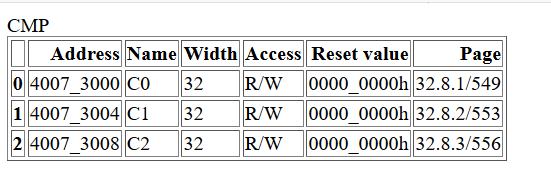


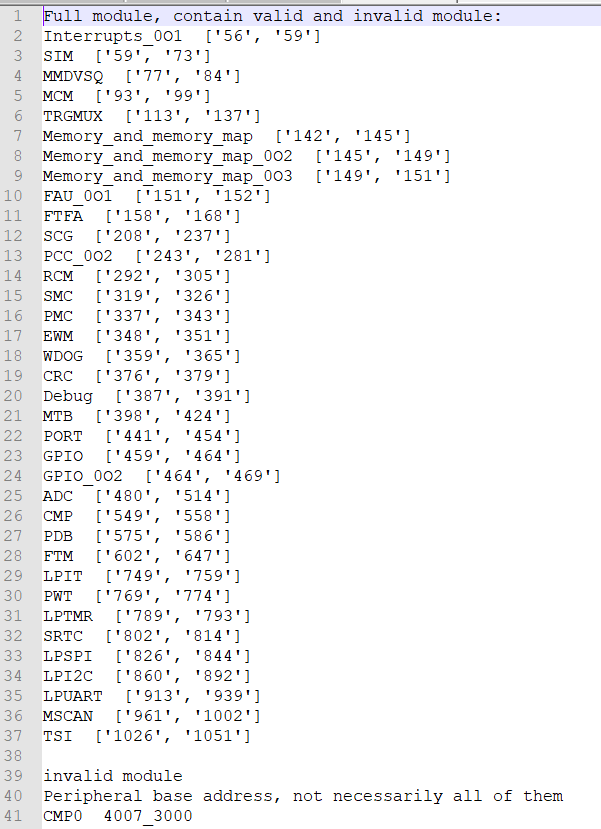
1. The issues of module base address
2. The issues of module register list
3. The issues of bit field.
4. Mismatch item failure.
5. Redundant item failure.

And in folder H\_test and test, there are more specific information of register:









1. Three files have different kinds of information of registers.
2. The bit.html file has all bit information about registers.
3. The module.html file has registers’ offset and access information.
4. The test\_info.txt file has the module page number and other information.

# 8 How to parser the content of PDF

The crucial part of this script is parser the content of PDF accurately, only the correct information can ensure the following comparative test.

# 9 FAQ

1. Can the auto test script cover all NPI?

**[Answer:]** Not now, even though we are trying to support all kinds of NPI, the PDF of LPC family is kind of different and data format is hard to process. So we only support IMX and Kinetis header file auto test. Still working on LPC family.

1. How long the auto test script need take?

**[Answer:]** It depends on the size of PDF,1000 pages take about 34 minutes, but as the number of pages increases, the time will take longer. So don’t need to wait for the result, you can do other jobs and wait for the end.

1. How time is saved by using this script?

**[Answer:]** It may cut half of time, though the script can do compare job for us, we also have to check the error.yml, the error list may have some unnecessary cause by PDF data or the logic of script, so it may take some extra time.