Pin Tool Data Auto Test Design Doc

|  |  |  |
| --- | --- | --- |
| Version | Author | Comments |
| 0.1 | Bill Yuan | Initial version |
|  |  |  |

Automatic test script as below:

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

subfolder: \pin\_tool\pin\_data\_val

# Awards

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

1. Frank Zhao who is the sponsor.
2. Charles Cai is responsible for reference manual doc analysis, he provides a uniform excel data to help us easy to parse.
3. Haley Guo is responsible for excel parser, he designs a script to parse the excel to a uniform comparable data.
4. Cindy and Ackey are the test owner of Pin Tool Chip Model, thanks for their fully support.
5. In addition, thanks Chip Model test group, they provide many good ideas and advice.

# Pre-request installation

1. Ruby 2.x.x 64 bits

<http://rubyinstaller.org/downloads/>

1. Ruby gem plugins, you can get them from git repo ruby\_env folder, click setup.rb to install all of them.
   1. Nokogiri
   2. mini\_portile
   3. awesome\_print
2. Reference manual pin database excel files. They have been located in rm\_pin\_db folder.

Please refer to the chapter “How to create pin reference database from RM doc” for more details.

1. Pin Tool internal version 1.0

[\\Zcz09fs.ea.freescale.net\PExProducts\Engineering\_Builds\Pins\_Tool\1.0\b160602\Installer\Win\_x64](file:///\\Zcz09fs.ea.freescale.net\PExProducts\Engineering_Builds\Pins_Tool\1.0\b160602\Installer\Win_x64)

# Knowledge required

1. Ruby
2. Pin Tool
3. MCU Pin Mux Configuration

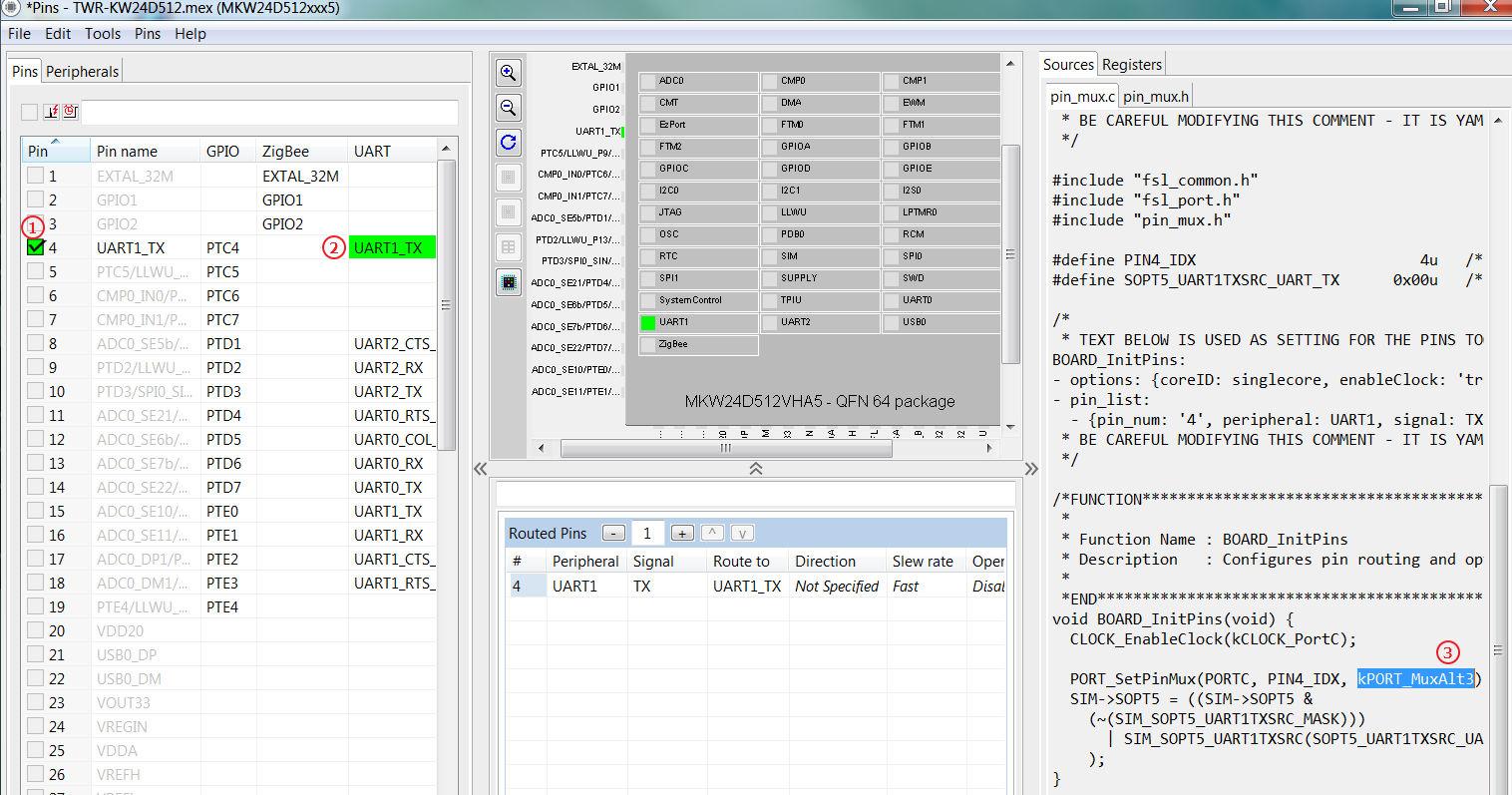
# Feasibility analysis

The data of Pin Tool is actual some xml files, and they include all database which is loaded by Pin Tool GUI. signal\_configuration.xml file is the most important file, and it contains all pin mux information.

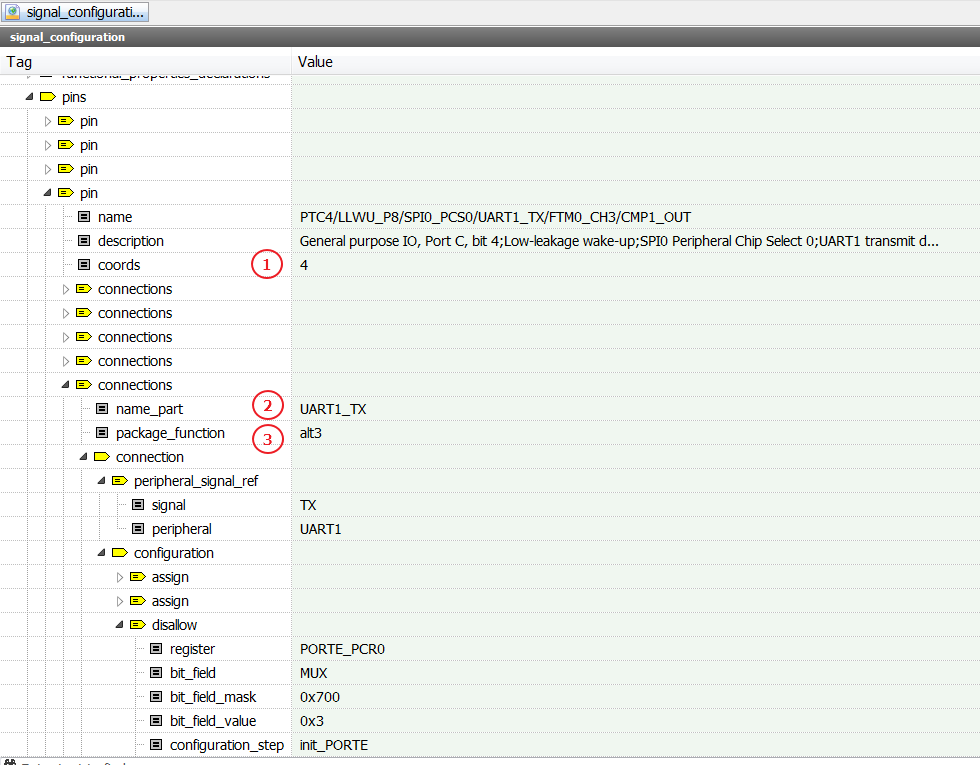
If we can compare the xml information to the reference manual doc, it’s the optimal direct test methodology to validate the pin tool data.

For instance, the following screen shorts show the relationship between the Pin Tool GUI, the XML file and the reference manual document, they all match.

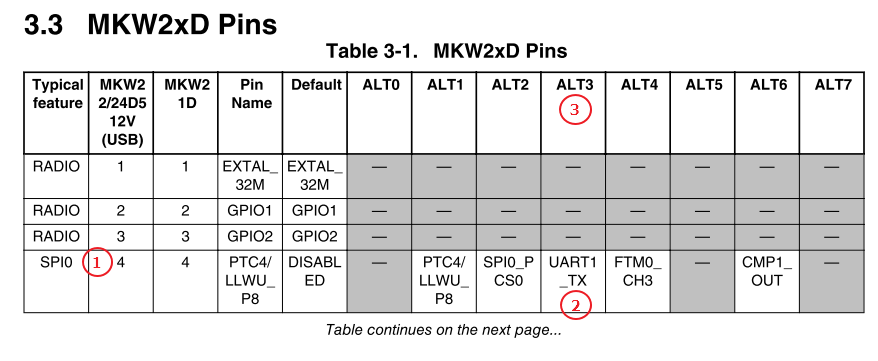
**Pin Tool GUI:**



**XML File:**

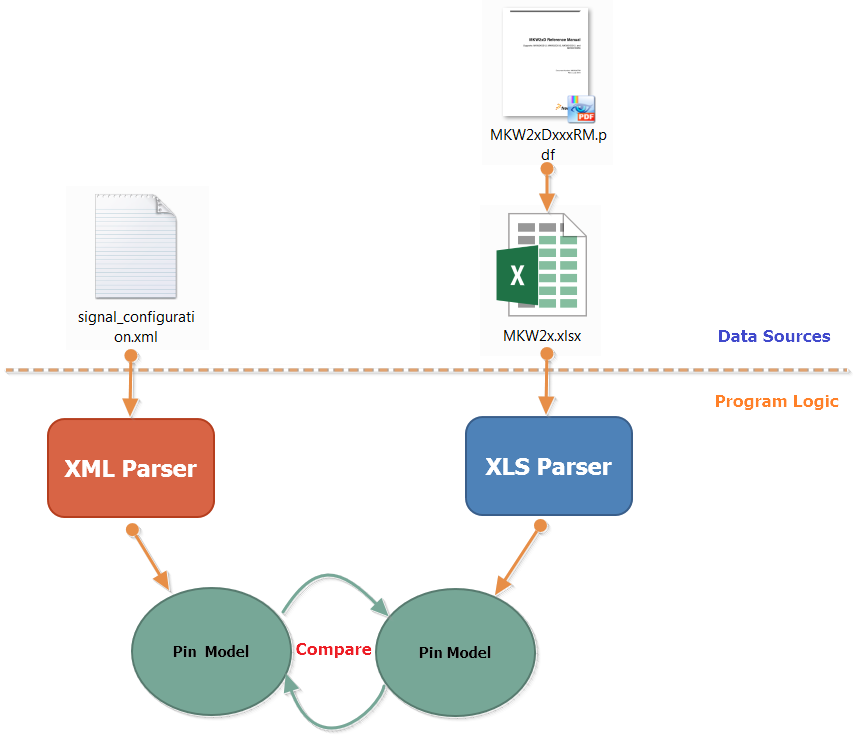


**Reference manual:**



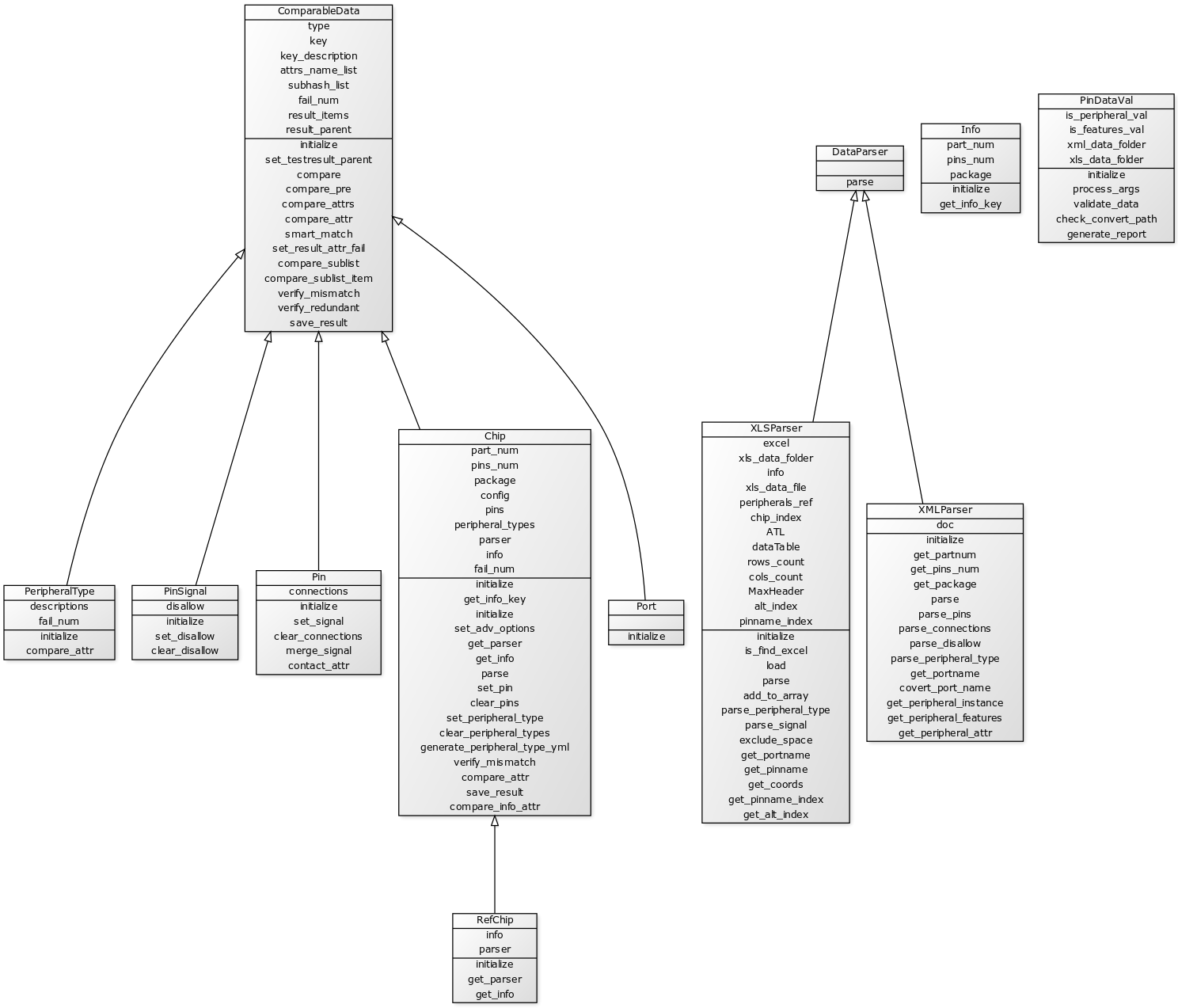
# Auto test design architecture

To compare the data what we want easily, and avoid the side effect for the XML format update in the feature, we design a model to convert all information to a uniform Pin Model data. The simple flow is listed as below.



The flowing is the detailed architecture by UML. Here is the key classes description.

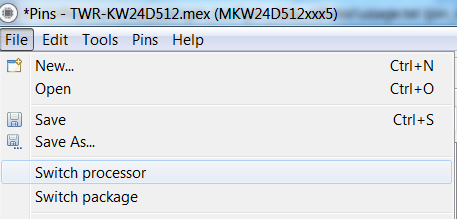
* ComparableData class:
  + The basic class supports to compare the attributes and list member variables, and save the test result.
  + The compare\_att, compare\_sublit, etc can be overwritten if the derived classes have different operation.
* Chip class:
  + It uses to store all pins connections, the peripheral type and chips information. It’s a root to keep all data.
  + It uses Proxy Design Pattern to call actual parser function in XMLParser or XLSParser.
* XMLParser class:
  + It uses xpath to search the xml node and packages them to the uniform pin model data, because it’s high performance and low memory cost. More details about xpath syntax, please refer to <https://msdn.microsoft.com/en-us/library/ms256122(v=vs.110).aspx>
  + If the xml format updates, we just need update this source code is enough.
* XLSParser class:
  + It uses WIN32OLE to open and get the excel data, and packages them to uniform pin model data.
* PinDataVal class:
  + The main entry to start the pin data auto test, it implements the flexible arguments parser.

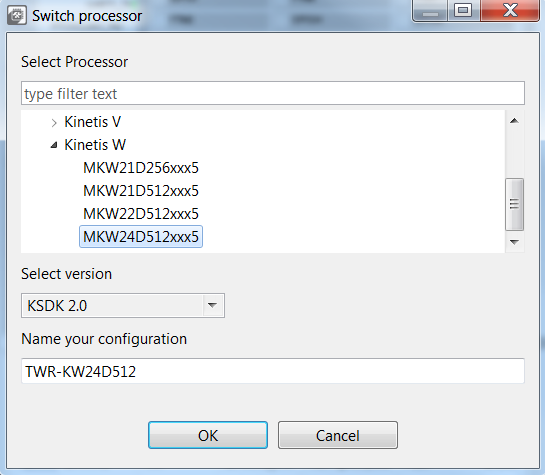


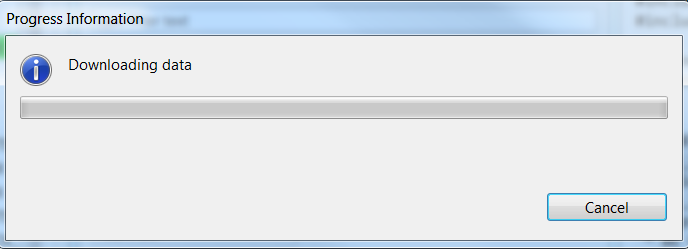
# user guides

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

1. Download the pin data by pin tool. The xml data will be located in C:\ProgramData\NXP\KEx\_Tools\1.0\mcu\_data







1. Get the auto test script, the “dev” branch is the latest code but not stable, the “master” branch is the previous stable version, the “release” branch is the testing version, will merge to “branch” after bug fixes.

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

1. Execute command in CMD window, note that, you’d better make sure the excel reference data is ready before testing:

*ruby verify\_data.rb -d C:\ProgramData\NXP\KEx\_Tools\1.0\mcu\_data\kex\_tools\processors\MK65FN2M0xxx18*

More detailed usage help: verify\_data [options]

-d [the parent folder of pin tool xml data processor],

--dir specify the parent folder of one processor

-r [Optional -- the parent folder of reference excel files from RM docs],

--ref Optional -- specify the parent folder of reference excel files

-p, --peripheral is enable peripheral instance value validation

-f, --features is enable features value validation

-h, --help print this help

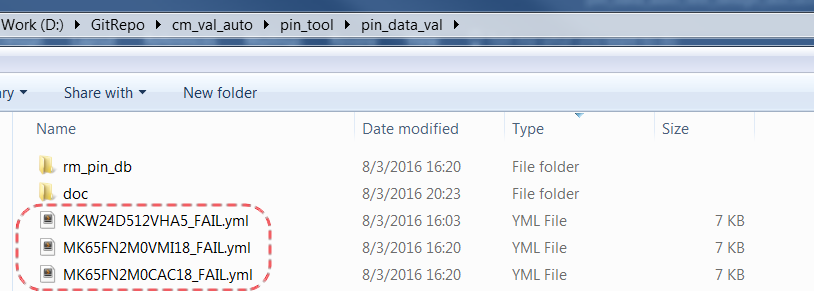
Note that, the “-p” and “-f” has not supported in the v 0.1, we are going to support it in next release.

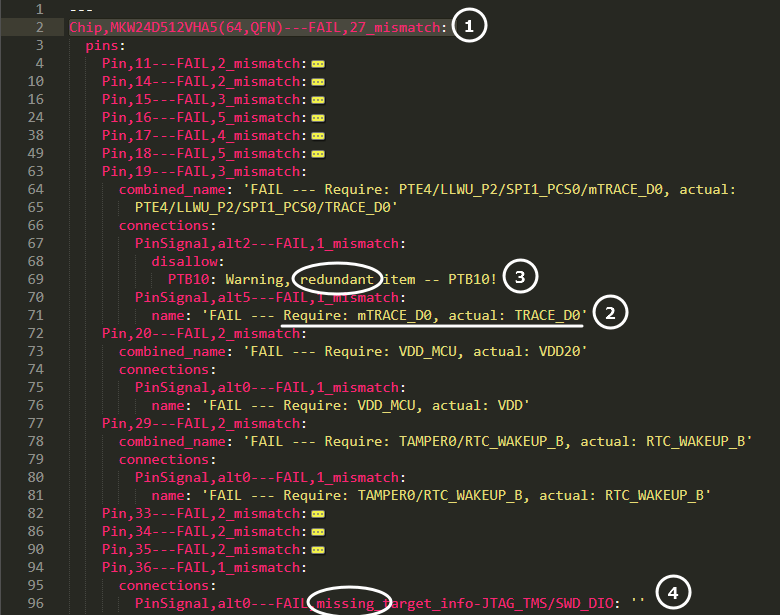
# Test report

The test result will be saved to a yaml file in current folder and divided by parts number name, and it only records the failures to reduce the efforts to check failures. There are three kinds of failures.

1. Mismatch: The xml data does not match the required reference manual data
2. Miss: The xml data misses the data that the reference manual required
3. Redundant: The xml data has redundant info after comparing the reference manual data.

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





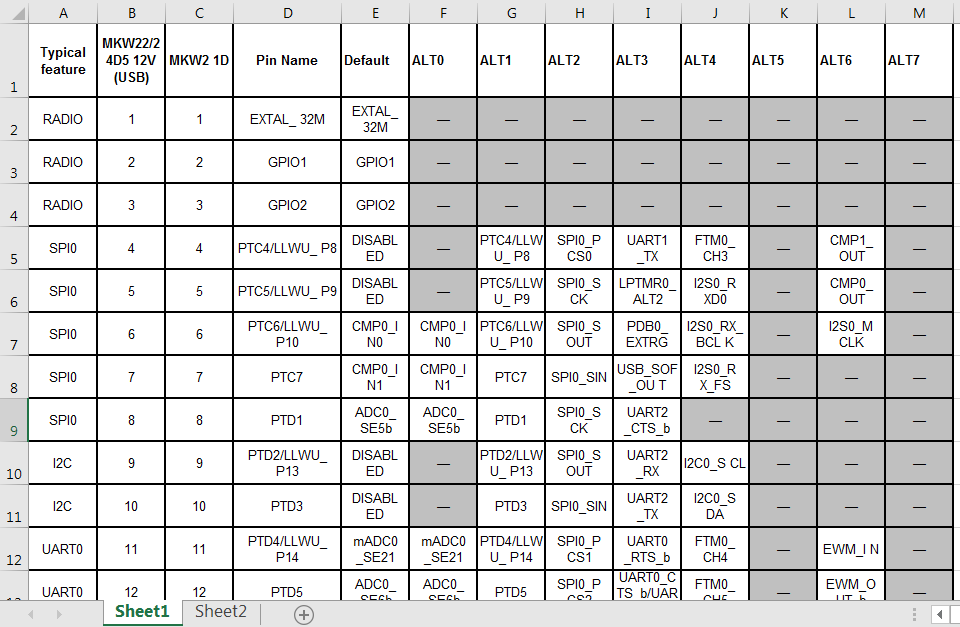
1. The chip info, what’s the part number and package. And list the how many mismatch in total.
2. Mismatch failure, list the required and actual value.
3. Redundant item failure.
4. Missing item failure.

# How to create pin reference database from rm doc

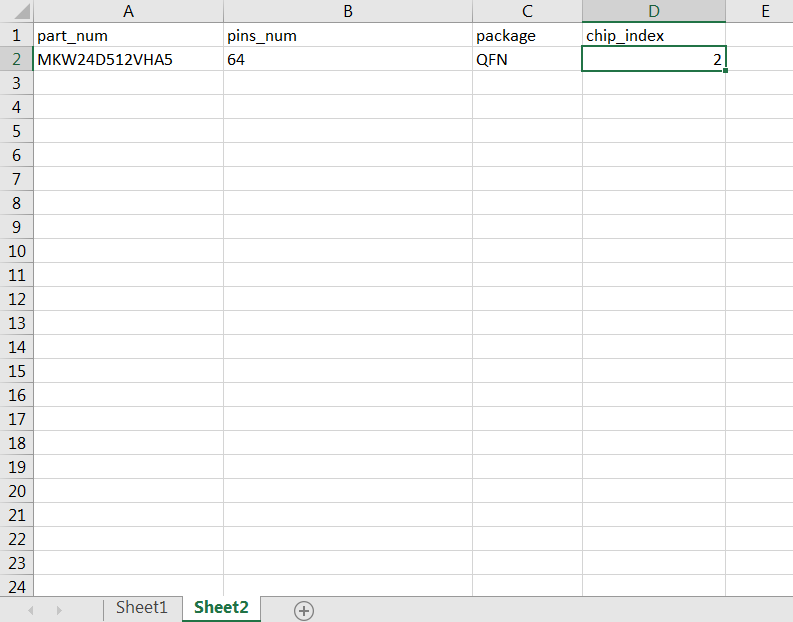
The excel database is almost same as the PDF reference manual doc. You just need open the PDF by word 2013 or higher, then copy the pin mux table to the excel, and modify a little to make the format is as the same as the reference manual.

The excel database includes two tables:

1. The first table should be the pin mux table.



1. The second table should be the part number and chip index table. The chip index is the column index the “Sheet1”, for example, MKW24D512VHA5 belongs to MKW22/24D5 12V (USB), it’s the second column in “Sheet1”, so set it as “2”.



# FAQ

1. Can the pin data auto test script support batch test for several chips?

**[Answer:]** Yes, you just need put all xml data into one parent folder, and set to the “-d” parameter. It will scan all signal\_configuration.xml files, and start the comparison one by one, all test report will be generated individual by part number name.

1. How long the auto test script need take?

**[Answer:]** It just takes about 2~3 seconds for one chip part number comparison and generating the test report.