**User Guide for flowGen for IQfact/IQfact\_plus**

last update: August 4, 2011

**Introduction**

The purpose of the flowGen program is to help the user generate large test files from a small set of user-specified parameters and values. The program iterates over all the parameter values specified by the user, creating tests for all possible combinations. The parameters and the values to be used are specified in an input file in the .csv (comma-separated values) format.

The technology is also specified in the input file. Multiple input files can be specified as well, in which case the program processes them in order and builds one test file for the whole data. Also, multiple independent input descriptions can be specified in one file, separated by blank lines.

**Quick Start**

To use the program do the following:

1. Create an input file in csv (comma-separated values) format as described in the section below.

2. Run the flowGen program using the input file, for example:

> flowGen input\example1\_wifi.csv -o output\example1\_wifi\_out.txt

This will generate the output test flow file 'example1\_wifi\_out.txt'.

3. Edit the output file, adding any values for limits that cannot be determined by the program. A better solution is that, if the limit values are set and will not need to be changed frequently, is to edit the template files and add limit expressions (described later in this document) so that the limit values will always be put into the output test file without any editing.

**Output Formats**

FlowGen can output multiple formats, with the default being the "IQlite" format, which is the output format described in this document.

**Examples**

In the installation directory for the flowGen program, the directory 'input' contains the following example input files:

example1\_wifi.csv

example2\_wifi\_mps.csv

example3\_wifi.csv

example4\_bt.csv

The 'output' directory contains the corresponding output files created by running those examples:

example1\_wifi\_out.txt

example2\_wifi\_mps\_out.txt

example3\_wifi\_out.txt

example4\_bt\_out.txt

In the root directory there are .bat files for running each of these examples:

run\_example1.bat

run\_example2.bat

run\_example3.bat

run\_example4.bat

Examining these example files can give you a better understanding of what the program does.

**Input File Format**

The program uses the “csv” format, so that a spreadsheet program like Microsoft Excel, OpenOffice, and others can be used to read and write the data. The “csv” format is a common format supported by these and other programs.

In the input file, the first line consists of the "TECHNOLOGY" keyword followed by a list of the parameter names that values will be specified for. Each keyword must be separated from the next one by a comma. The subsequent lines in the file consist of the parameter values corresponding to the keywords. The data is in a column format, so the parameter values must be in the same column as its keyword name. Values fields are separated by commas. A field may be empty, which is indicated by two commas in succession. All columns must be accounted for, so enough commas must be added. Any number of values may be specified for each parameter, so some columns will contain more values than others. A parameter name specified in the first line must have at least one parameter value specified.

The TECHNOLOGY column must have only one value specified (multiple technologies specified in the same input file are not supported now).

Only parameters specified in the input file will be processed by the program. All other parameters will be assigned their default values (or whatever value is specified in the template files - more on this later).

Following is an example input file (this example iterates over all the valid data rates for the WIFI 802.11g protocol):

TECHNOLOGY,DATA\_RATE,FREQ\_MHZ,TEST,TX\_POWER\_DBM

WIFI,OFDM-6,2412,TX\_VERIFY\_EVM,15.0

WIFI,OFDM-9,,,

WIFI,OFDM-12,,,

WIFI,OFDM-18,,,

WIFI,OFDM-24,,,

WIFI,OFDM-36,,,

WIFI,OFDM-48,,,

WIFI,OFDM-54,,,

In this example, the technology specified is “WIFI”. Eight values for DATA\_RATE are specified (OFDN-6 through OFDM-54), one channel (2412 Mhz), one test analysis (TX\_VERIFY\_EVM), and one value for the TX\_POWER\_DBM parameter (15.0). The program will generate tests for all combinations of these parameters. In this example, only the DATA\_RATE parameter has multiple values specified. The other parameters will use only the one value for all tests. All other parameters not specified in the file (e.g., BANDWIDTH, CABLE\_LOSS, etc.), will have their default values assigned (or whatever values are specified in the template files). This input file will thus generate eight TX\_VEIRFY\_EVM tests.

Note the extra commas added which are necessary in csv format to specify field separation, even for empty fields. Do not use spaces or other whitespace characters in the .csv input file; they will be interpreted as part of the data.

The input file treats the analysis type or measurement as a parameter, and the program will create the appropriate test types and will use the test type as part of the combinations along with the other parameters.

The program will generate the necessary header sections including tester and DUT setup and disconnect also, so that the output test file is a complete file.

Now consider another input file example:

TECHNOLOGY,FREQ\_MHZ,DATA\_RATE,TEST,TX\_POWER\_DBM,ANTENNA

WIFI,5180,OFDM-9,TX\_VERIFY\_EVM,16.0,"(1,0,0,0)"

WIFI,5300,OFDM-36,TX\_VERIFY\_MASK,,

WIFI,5745,,RX\_VERIFY\_PER,,

This example is again for WIFI, specifying channels and rates for the 802.11a protocol. It contains multiple values for multiple parameters. FREQ\_MHZ, DATA\_RATE, and test contain multiple values that the program will iterate over and create combinations of tests for. This input file will thus generate 3 x 2 x 3 x 1 x 1 = 18 tests.

Following is an example of an input file for the WIFI\_MPS technology.

TECHNOLOGY,FREQ\_MHZ,DATA\_RATE\*,TEST\*,TX\_POWER\_DBM\*,ANTENNA

WIFI\_MPS,2422,DSSS-1,"E,M,P",15.0,"(1,0,0,0)"

,2457,CCK\_5-5,"E,M,P,S",,

,2484,,"L,H,S",,

The WIFI\_MPS technology supports “multipackets” which are sequences in which some parameters (e.g., channel, antenna) can only have one value while other parameters (e.g. DATA\_RATE, TX\_POWER\_DBM, and the test measurement type) can take multiple values. The parameters which can take multiple values within a multipacket sequence are denoted by a ‘\*’ after their names in the input file. These parameters **must** have the ‘\*’ notation, while the others must not. The above example will create three multipacket sequences, one for each of the three channels (2422, 2457, 2484). The ANTENNA and TX\_POWER\_DBM parameters will always use the same one value that is given. Each multipacket sequence will iterate over the values of DATA\_RATE and test type and TX\_POWER\_DBM, so there will be 2 x 2 x 1 = 4 sections in each of the three TX multipacket sequences, and 2 x 2 x 1 = 4 sections in the one RX multipacket sequence.

For WIFI\_MPS the test type is specified by a special string of letters separated by commas. This represents a sequence within a sequence only for the test type. “E,M,P” means a TX\_VERIFY\_EVM - TX\_VERIFY\_MASK - TX\_VERIFY\_POWER sequence. This is specified within one “TX\_TEST\_ADD” section in the output file. The “L,H,S” test sequence is an RX sequence which indicates a RX\_VERIFY\_PER - RX\_SENSITIVITY - RX\_SWEEP\_PER sequence.

**Multi-section csv File Input**

FlowGen can process a file containing multiple “csv sections”. A “section” is a complete csv description that could exist as a single csv file input. This allows more flexibility in creating a more varied flow file, while allowing the user to have the whole description in one csv file.

Each of the “sections” must be separated from other sections by a blank line, or a line containing all whitespace, or a line containing all commas. FlowGen will split this file into temporary separate csv files, and then process each of those temporary files in the normal way. Each of the “sections” of a multi-section csv file must conform to the rules that a single-section csv file does.

Microsoft Excel will add a line of all commas for any blank lines for any empty row you add while editing the csv file. It will also add commas to the ends of any lines representing empty columns. This very commonly happens when creating a csv file of different sections, since each section may have a different number of parameters. FlowGen will detect the excess commas and recognize it for an empty column.

Here is an example of an two-section csv file:

TECHNOLOGY,FREQ\_MHZ,DATA\_RATE,TEST,TX\_POWER\_DBM,ANTENNA

WIFI,2432,OFDM-9,TX\_VERIFY\_EVM,15.5,"(1,0,0,0)"

,2447,DSSS-1,TX\_VERIFY\_MASK,,

,,,,,

TECHNOLOGY,DATA\_RATE,FREQ\_MHZ,TEST,TX\_POWER\_DBM

WIFI,OFDM-6,2412,TX\_VERIFY\_EVM,15

,OFDM-9,2417,TX\_VERIFY\_POWER,

,OFDM-12,,,

,OFDM-18,,,

,,,,

**Running the Program**

The program is run through the python interpreter and is run as follows:

> flowGen input\_file+ [-o output\_file\_name] [-f output\_format]

The program can take one or more input files on the command line. The “-o” option can be used to specify the name of the output file. If not specified, it will write the output to “test\_flow.txt”.

The “-f” format specified the output format, which for the IQlite format should be “iqlite”, or if the option is not specified, then the default value of “iqlite” is assumed. The other supported formats are not described here.

Examples:

> flowGen test1.csv test2.csv test3.csv -o all\_tests.txt

> flowGen test1.csv test2.csv test3.csv -o all\_tests.txt -f iqlite

The program will read the three input files, and generate one output file.

**About Template Files**

The program uses template files to write out the output test file. It has almost no knowledge about the output format. The template files reside in the ‘templates’ directory, and consist of multiple files containing sections of text in the test file, directives to the program, and substitution macros embedded in the text sections. The program reads the appropriate template files, performs the substitutions, and writes it out to the output file.

The ‘templates’ directory has sub-directories for each support technology: ‘WIFI’, ‘WIFI\_MPS’. At the root level and in the sub-directories there are header and tail files, and files with names of the sections they contain, such as ‘TX\_VERIFY\_EVM.txt’. The user may modify any of these files to add limits or connection parameters or any other specific data that does not change during the running of the test.

The files with names ‘template.txt’ and ‘sequence.txt’ contain only directives to the program, and should NOT be edited by the user. These files contain no text that is written to the output file.

The template files contain substitution macros at the places in the section text where the parameter values should be written. Here is an example from the ‘TX\_VERIFY\_EVM.txt’ template file:

TX\_VERIFY\_EVM

...

>FREQ\_MHZ [Integer] = %%sub getvalue("FREQ\_MHZ", "2412")%%

>TX1 [Integer] = %%sub getvalue("ANTENNA", 0, "1")%%

>TX2 [Integer] = %%sub getvalue("ANTENNA", 1, "0")%%

>TX3 [Integer] = %%sub getvalue("ANTENNA", 2, "0")%%

>TX4 [Integer] = %%sub getvalue("ANTENNA", 3, "0")%% ....

>TX\_POWER\_DBM [Double] = %%sub getvalue(“TX\_POWER\_DBM”, “15.0”)%%

>BANDWIDTH [String] = HT20

>DATA\_RATE [String] = %%sub getvalue(“DATA\_RATE”, “OFDM-54”)%%

The ‘%%sub ...%%’ string is a substitution directive that will be replaced by the value of the parameter, e.g.:

>FREQ\_MHZ [Integer] = 2442

‘getvalue’ is an internal function used to obtain the parameter value to substitute. The first parameter to ‘getvalue’ is the name of the parameter, and the second value is the default value (“2412” in the above example) that will be used if the user did not specify any value in the input file, unless there are three parameters, then the third parameter is the default value, and the second parameter is an index used when the parameter is a set or array of data. The ‘ANTENNA’ parameter is a special parameter that is described in the next section, which uses an array of values.

Substitution macros are used for parameter values that can vary from test to test and are not global to the test file.

**Alternate Template Directories**

By default, FlowGen will look for the templates in the “./templates” sub-directory. By using the “-t” command line option, you can tell FlowGen to use another directory to find the templates. If FlowGen cannot find the template file in the specified directory, it will try to find it in the default directory. In this way, you can have special partial template directories that contain only the template files that you have customized.

Example:

flowGen -t templates.limits test\test\_flow1.csv

**Creating Input File**

The user must supply the technology being used (‘WIFI’, ‘WIFI\_MPS’, etc.) by adding the ‘TECHNOLOGY’ keyword on the first line of the file, and supplying a value for it (e.g., ‘WIFI’) on the second line, as in the example above. The subsequent lines in the file do not have to specify the technology again.

Any test parameter can be specified in the input file. The templates currently provide for the substitution of the following parameters: FREQ\_MHZ, DATA\_RATE, TEST, TX/RX\_POWER\_DBM, and ANTENNA. Any other parameters that the user wants to specify must have the template files that they are in edited to add the substitution macro string so the program can make the substitution.

**Special Parameters**

There are currently two ‘special’ parameters: TEST and ANTENNA. The parameter ‘TEST’ in the input file specifies the type of test analysis to perform. This parameter is referenced by directives in the template files which cause the program to iterate and generate multiple sections as needed.

The ‘ANTENNA’ parameter is a special handling of the TX and RX antenna parameters, which is designated using ‘TX1’ to ‘TX4’ for TX tests, ‘RX1’ to ‘RX4’ for RX tests, and ‘ANT1’ to ‘ANT4’ for the WIFI\_MPS tests. The ANTENNA parameter groups the values together and treats it as one unit of data, instead of four. This prevents iteration over all the four values if specified separately in the input file, which would make no sense. The ANTENNA is thus specified as a tuple of four values within a string, e.g.:

TECHNOLOGY,FREQ\_MHZ,DATA\_RATE,TEST,TX\_POWER\_DBM,ANTENNA

WIFI,5180,OFDM-9,TX\_VERIFY\_EVM,16.0,"(1,0,0,0)"

....

The output template specifies the ANTENNA keyword passed to the ‘getvalue’ substitution function, and also an the index as the second argument. This tells the program which value from the tuple to replace the substitution macro with (using zero-based indexing). The third parameter is the value assigned to that tuple.

For example, for the following’ lines in the template file:

>TX1 [Integer] = %%sub getvalue("ANTENNA", 0, "1")%%

>TX2 [Integer] = %%sub getvalue("ANTENNA", 1, "0")%%

>TX3 [Integer] = %%sub getvalue("ANTENNA", 2, "0")%%

>TX4 [Integer] = %%sub getvalue("ANTENNA", 3, "0")%% ....

using the above input file will result in the following substitution:

>TX1 [Integer] = 1

>TX2 [Integer] = 0

>TX3 [Integer] = 0

>TX4 [Integer] = 0

The ANTENNA parameter values specified in the csv input file will be used for both the TX and RX parameters.

**Specifying Limits**

FlowGen supports special directives to specify expressions or maps to be used for computing lower and upper limits for test result parameters. The expressions can be anything that can be evaluated by python to a string. It also accepts any input parameter names specified for that test in the expression.

Here is an example of limit expression directives for the POWER\_AVG\_DBM result:

<POWER\_AVG\_DBM [Double] = <%%lim = TX\_POWER\_DBM - 1.5%%,%%lim TX\_POWER\_DBM + 1.5%%>

The limit directive uses the “%%lim” designator. Everything after the “=” until the “%%” is taken to be an expression that will be evaluated. In this case, “TX\_POWER\_DBM” is an input parameter that FlowGen understands, so its value will used in evaluating the expression. The result of the evaluation will be written to the flow file.

FlowGen also support limit maps, which are more complex, and allow the user to specify a mapping of one or more input parameter values to an expression. Following is a simple example:

<FREQ\_ERROR\_AVG [Double] = <%%lmap (DATA\_RATE,FREQ\_MHZ)

OFDM-6,2412= -17-10;

OFDM\*,24\* = -18 + 7;

\*,\*8 = -33;

\*,\*=-19;%%,>

The limit map directive uses the “%%lmap” designator. Due to the potential size of the maps, it may extend over multiple lines. The format is:

%%lmap (param1, param2, ....)

pattern1, pattern2 = expression;

pattern1, pattern2 = expression;

...

pattern1, pattern2 = expression;%%

The number of parameters must match the number of patterns. FlowGen will process the map by starting at the beginning and finding the first patterns that match the input parameter values. It will then use the corresponding expression to compute the limit value. After the end of map designator “%%”, the rest of the line continues. A result can have limit maps or just expressions for either or both limits.