**LabVIEW VI standard**

(Version 2)

1. **Overview**

This standard framework file is utilized to help the group developer in their LabVIEW programming. In our group, each group member takes the responsibility for one or more function blocks in MGCC and MGLC. This file is to design a standard for each function VI.

Basically, we will have three independent kinds of projects: 1) MGCC project, 2) MGLC project, and 3) Visualization. Thus, each compactRIO should have only one real-time project running on it. My idea is to design only three independent projects which will be shared among developers in our group.

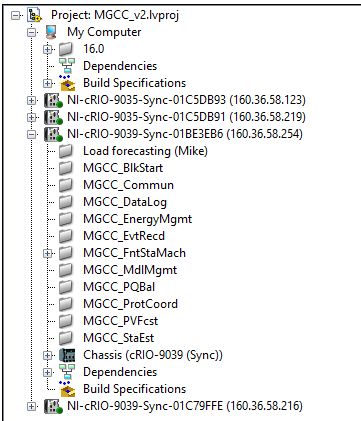
1. **MGCC project framework**

**Functions**

In this project, several functions will be implemented. The list of functions and the related developer is shown as follows:

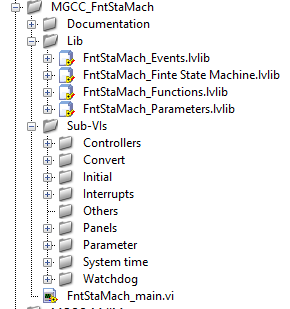
1. MGCC\_PQBal
2. MGCC\_BlkStart
3. MGCC\_Commun
4. MGCC\_DataLog
5. MGCC\_EnergyMgmt
6. MGCC\_EvtRecd
7. MGCC\_FntStaMach
8. MGCC\_LoadFcst
9. MGCC\_MdlMgmt
10. MGCC\_ProtCoord
11. MGCC\_PVFcst
12. MGCC\_StaEst

Here, each function will be implemented in an independent document, e.g., MGCC\_FntStaMach, as follows.

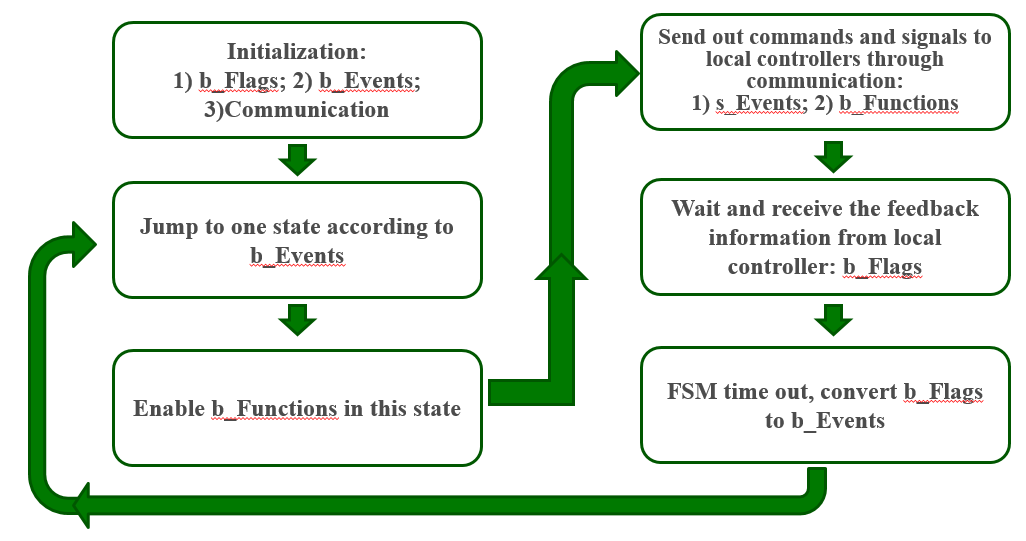


**Frameworks**

The framework for each function document contains libs, sub-Vis, simulation models and local main VI. They should also be classified into predefined documents such as following. The local main function should be listed independently.



Please carefully design your local main VI. This one will be implemented in the higher level main VI. For this main VI, please 1) listed all the outputs into an independent lib and make sure all the inputs can be found from the libs of other VIs. 2) it would be good to show the function block diagram which will make the functions, input/output much more clearly.



3) please do take care of the response time of your VIs because you may need watchdog timer to restart the compactRIO. 4) Create an ENABLE BUTTON, this is important for higher level VI to give commands to enable these functions in main VI.

The simulation models, e.g., Simulink model, should also be stored in a sub-folder in the project for reference purposes. (According to Dr. Wang’s suggestion on 8/28/2017. Updated on 8/31/2017)

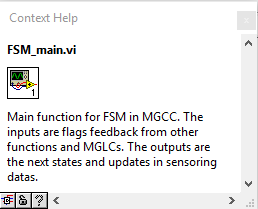
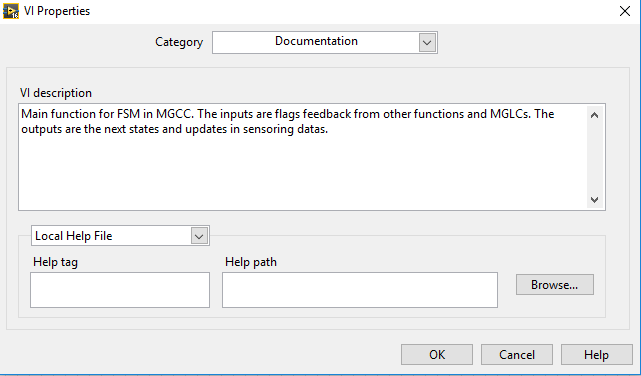
**Comments**

The comments and VI descriptions are necessary in writing LabVIEW codes. In this project, please write the comments in your local main VI and give VI description in the VI document.

Suggested contents:

1. Functionality
2. Data type
3. Valid range (for inputs)
4. Default value (for inputs) You can also list the default value in parentheses as part of the control or indicator name.
5. Behavior for special values (0, empty array, empty string, and so on)
6. Additional information, such as if the user must set this value always, often, or rarely

The example is shown as follows:



1. **Name Convention**

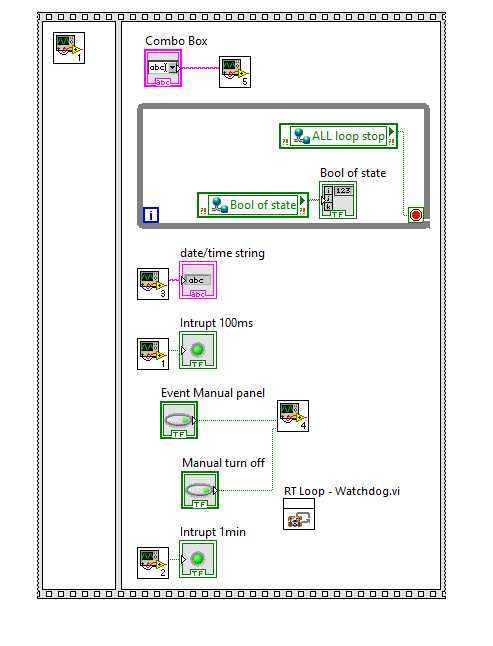
The file/variable name regulation will be introduced by Lin in detailed.

Please refer to name\_convention.xlx provided by Lin.

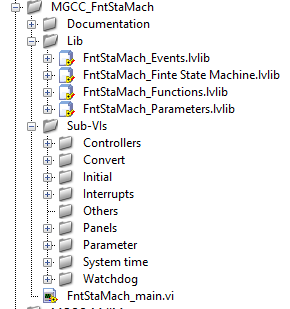
1. **Programming tips**

There are several recommendations for function Vis:

1. Store all files within a single root directory (Different from virtual document, do store them in a single root document.)
2. Divide your application into logical pieces of manageable size, make it clear and easy to read. This will be benefit to other developer. Try to normalize your VI within one screen E.g., :



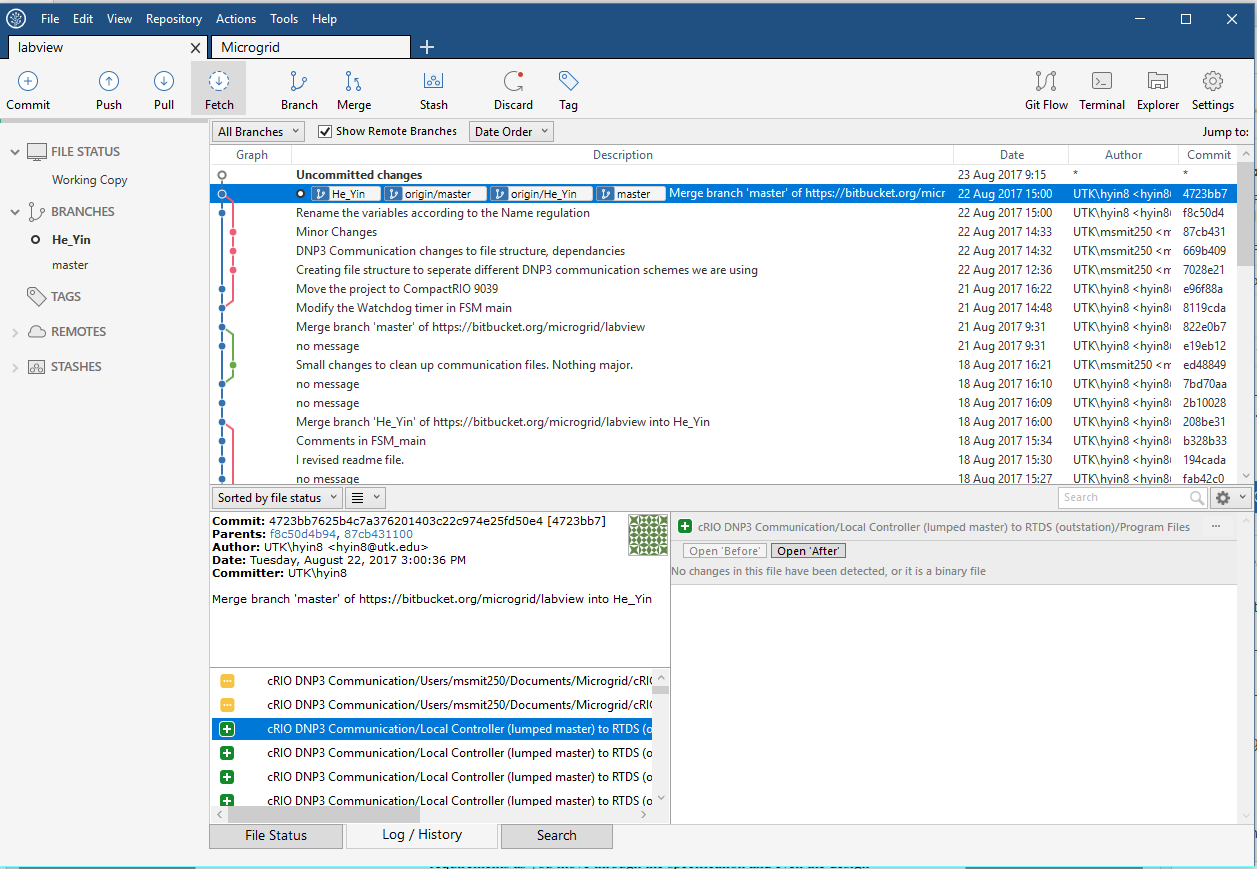
1. Use logical and descriptive naming conventions (According to Lin’s regulation)
2. Separate the top-level VIs from other source code



1. Begin with a high-level block diagram that includes the main components of your application (for example, the block diagram could include separate frameworks for configuration, acquisition, analysis, data display, data logging, and error handling)
2. Group or “bucket” files according to predetermined criteria
3. Write more comments to help other developers for variables, lines and sub-Vis.
4. Use more math-code blocks for complicated calculations.
5. Other tips please refer to LabVIEW\_rules\_to\_writes.docx
6. **Sourcetree and Bitbucket**

In this project, we are using the GIT(sourcetree) and remote server(Bitbucket) as our source control tools. In this section, we will introduce how to use sourcetree and Bitbucket.

1. Get your account from (https://bitbucket.org/account/signin/) and send a message to HE YIN with your registration address.
2. Download the sourcetree from (<https://www.sourcetreeapp.com/>)
3. If HE YIN has confirmed your e-mail address, congratulation!, you are in and can see our team project “Microgrid”. The Interface of sourcetree is shown as follows:



1. Now you may pull from the master and get the current project files. Remember to install LabVIEW 2016 ,CompactRIO, real-time module, and math module.
2. When you have modified your codes, please create your own branch. Remember pull from master FIRST and then push.
3. If you want to find your old version or Beta version, please use check out from your previous check points.
4. **Check list**

**Please refer to (http://www.ni.com/pdf/manuals/321393d.pdf)**