**LabVIEW VI standard**

(Version 1)

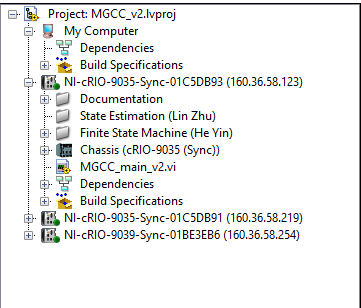
**Overview**

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.

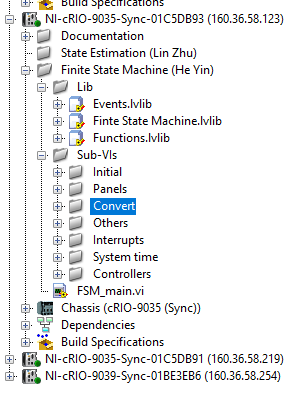
Basically, we will have two independent kinds of projects: 1) MGCC project, 2) MGLC project. Thus each compacrRIO should have only one real-time project running on it. My idea is to design only two independent projects which will be shared among developers in our group.

**MGCC project**

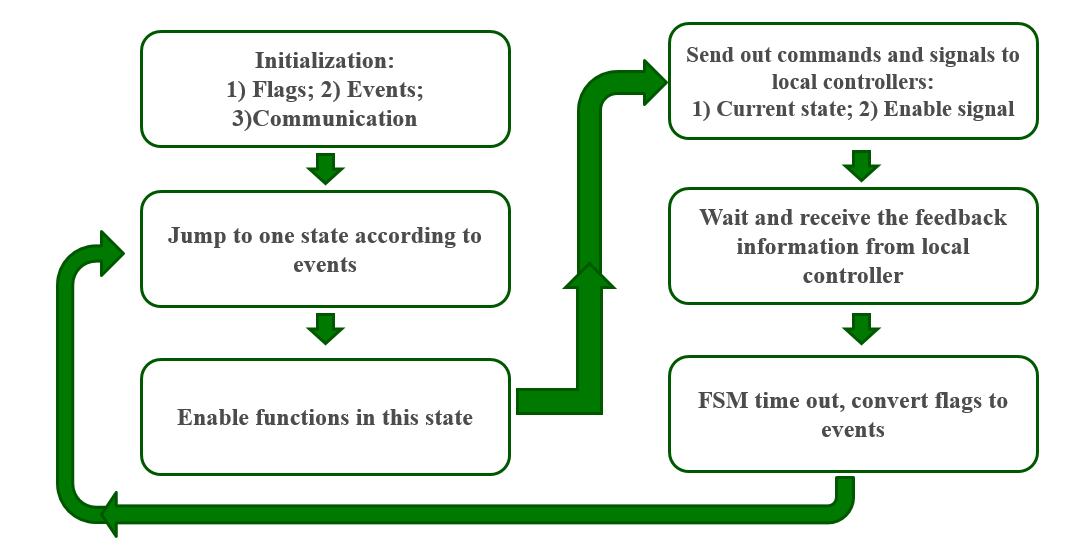
In this project, several functions will be implemented such as finite state machine, state estimation, power balancing, load/PV prediction, protection, and communication. Here, each function will be implemented in an independent document with the name of the developer, e.g., Finite State Machine (He Yin), as follows.



The framework for each function document contains libs, sub-Vis 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. That one will be implemented in the mean 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) leave an ENABLE BUTTON, this is important for higher level VI to give commands to function main VI.

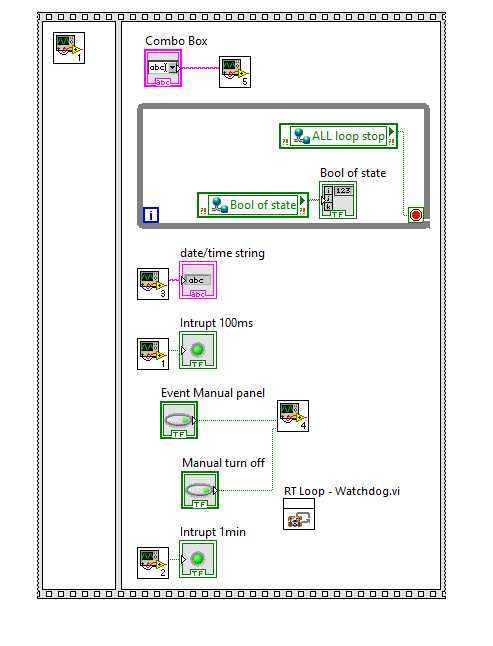
**Name Regulation**

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

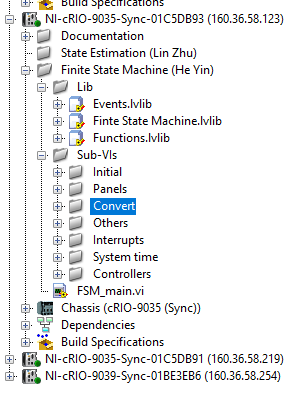
**Comments and recommendations**

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 complicate calculations.