GUI Quick Start Guide:
foc_encoder_speed
Encoder based FOC Speed & Torque control with Sensorless (SMO) redundancy

Version 1.0.3 *Motor Solutions*

GUI Composer

GUI Composer is an HTML5 based graphical user interface design tool that is integrated with the Code Composer Studio (CCS) development environment which allows creation of custom GUIs that directly instrument the software variables running on the TI processor. GUI Composer simplifies project development, increases productivity and decreases debugging by combining all the software tools necessary to develop both target code and an instrumentation GUI into one single convenient development environment.

Although GUI Composer is integrated into CCS, it is not necessary for the computer connected to the target to have CCS installed. A free, smaller GUI Composer Runtime installation is available for any computer that needs to run a created GUI.

The GUI Composer Wiki page contains all download, usage, architecture, and support information.

GUI Composer Details

All project folders and files **must reside** in the webapps directory of a valid GUI Composer installation to work correctly:
c:\ti\guicomposer\webapps <project_folder>

Note that for each GUI a project folder is included in the MotorWare directory for completeness only
\sw\solutions\<solution_name>\gui\guicomposer\webapps
This folder is simply copied to c:\ti\guicomposer\webapps\ upon install

A GUI Composer Project is made up of six different types of files

1. **.html**
   - The heart of the aesthetics of a GUI Composer GUI is defined in the html files.
   - Widget type, size and position as well as character type, font, size, and background image/color are defined.
   - A GUI can be made up of one or more html files and each file can have any filename.
2. **.json**
   - json files are what link the target variables in the CCS project to the GUI instruments
   - For each and every html file there is exactly one corresponding json file with the same filename.
   - For each and every widget there is one corresponding widget-to-target_variable definition in the .json file.
3. **appConfig.ccxml**
   - This is the target configuration file which defines how GUI Composer interacts with the target. Typically a target configuration file is imported from CCS.
   - The target configuration file must be named appConfig.ccxml

4. **appInitScript.js**
   - This initializes and launches the Debug Server, runs the appConfig.ccxml to create a JTAG socket connection with the target, and loads appProgram.out
   - The JavaScript file has to be named appInitScript.js.
5. **appProgram.out**
   - This is the project specific binary file built from CCS that gets loaded into the target each time the GUI is launched.
   - GUI Composer only reads/loads a target file if it is named appProgram.out
6. **launcher.exe**
   - The launcher is the executable that starts a GUI Composer session, using the other files to build and connect the GUI
   - The launcher should always be present in a GUI Composer project directory and should always be named launcher.exe.
   - A shortcut to **c:\ti\guicomposer\webapps** <project_folder>\launcher.exe is used to place the GUI executable in a logical location within an independent MotorWare directory structure:
     \sw\solutions\ <solution_name >\boards\<board>\<family>\<target>\gui\launcher.exe

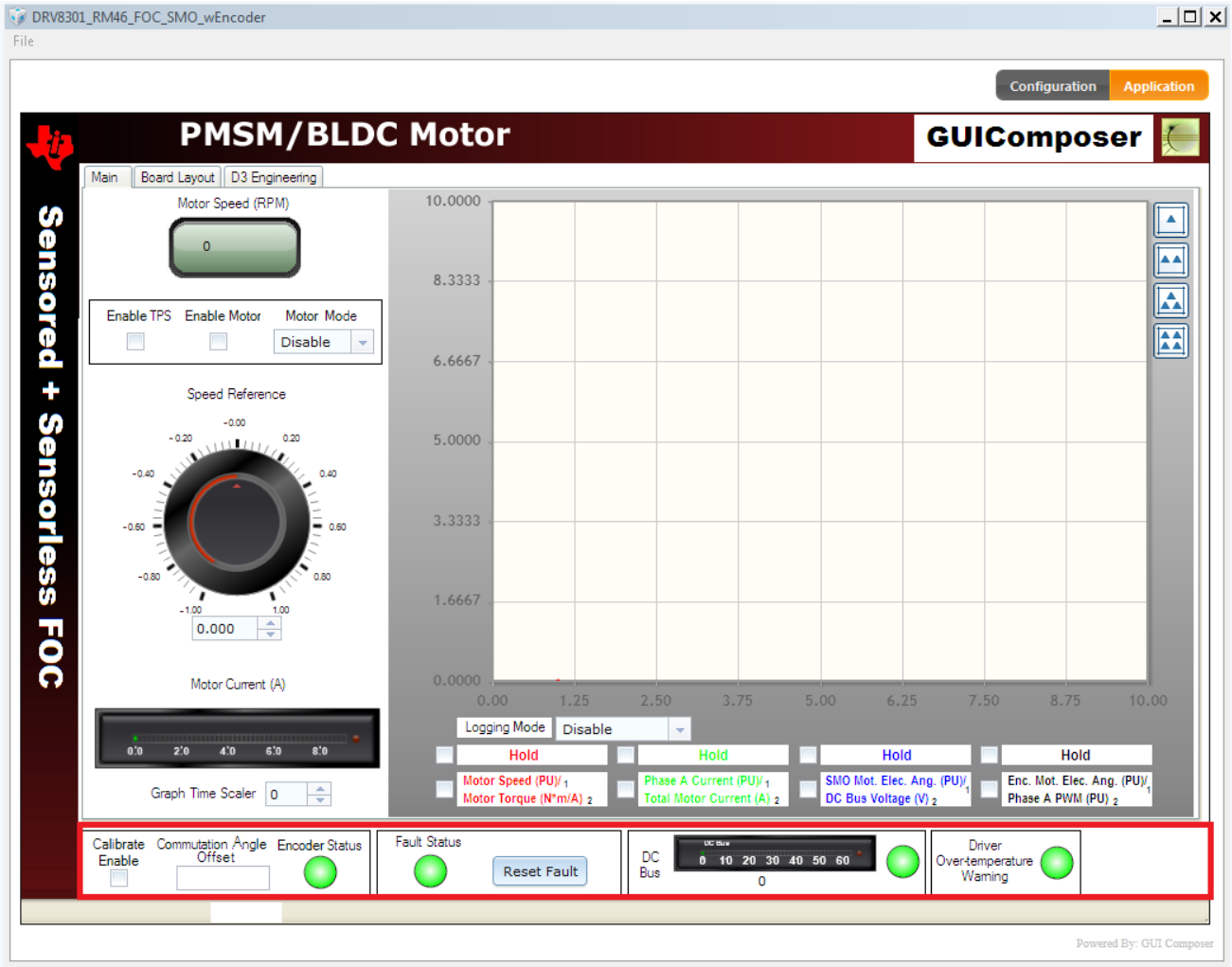
Running a GUI Composer GUI

1. Make sure all hardware instructions have been followed from \sw\boards\kits\<kit_part_number>\docs\<kit_part_number>_READMEFIRST.pdf
2. Run \sw\solutions\<solution_name>\boards\<board>\<family>\<target>\gui\\launcher.exe
3. If everything was configured correctly the launcher should automatically
   a. Start GUI Composer
   b. Detect & connect to the controlCARD target
   c. Load **c:\ti\guicomposer\webapps** <project_folder>\ appProgram.out to the target
   d. Run the appProgram
   e. Configure and display the instrumentation GUI
4. Errors
   a. If the GUI does not launch
      i. Make sure GUI Composer Runtime is installed
      ii. Verify you can run a launcher.exe from C:\ti\guicomposer\webapps\<project_name>
   b. If the GUI gets stuck on the configurations tab, rectify the Console Output error.
      - Make sure all jumpers and switches are set to the appropriate position
      - **c:\ti\guicomposer\webapps** <project_folder>\ project files are present, named correctly, and are in proper location.
      - If the problem persists triple check ALL HARDWARE CONNECTIONS on the controlCARD, drive board, motor, and any sensor connections

foc_encoder_speed

Encoder based FOC Speed & Torque control with Sensorless (SMO) redundancy

Solutions Supported:
\sw\solutions\foc_encoder_speed\boards\drv8301kit_revD\hercules\tms570ls1227\gui\launcher.exe
\sw\solutions\foc_encoder_speed\boards\drv8301kit_revD\hercules\rm46l852\gui\launcher.exe

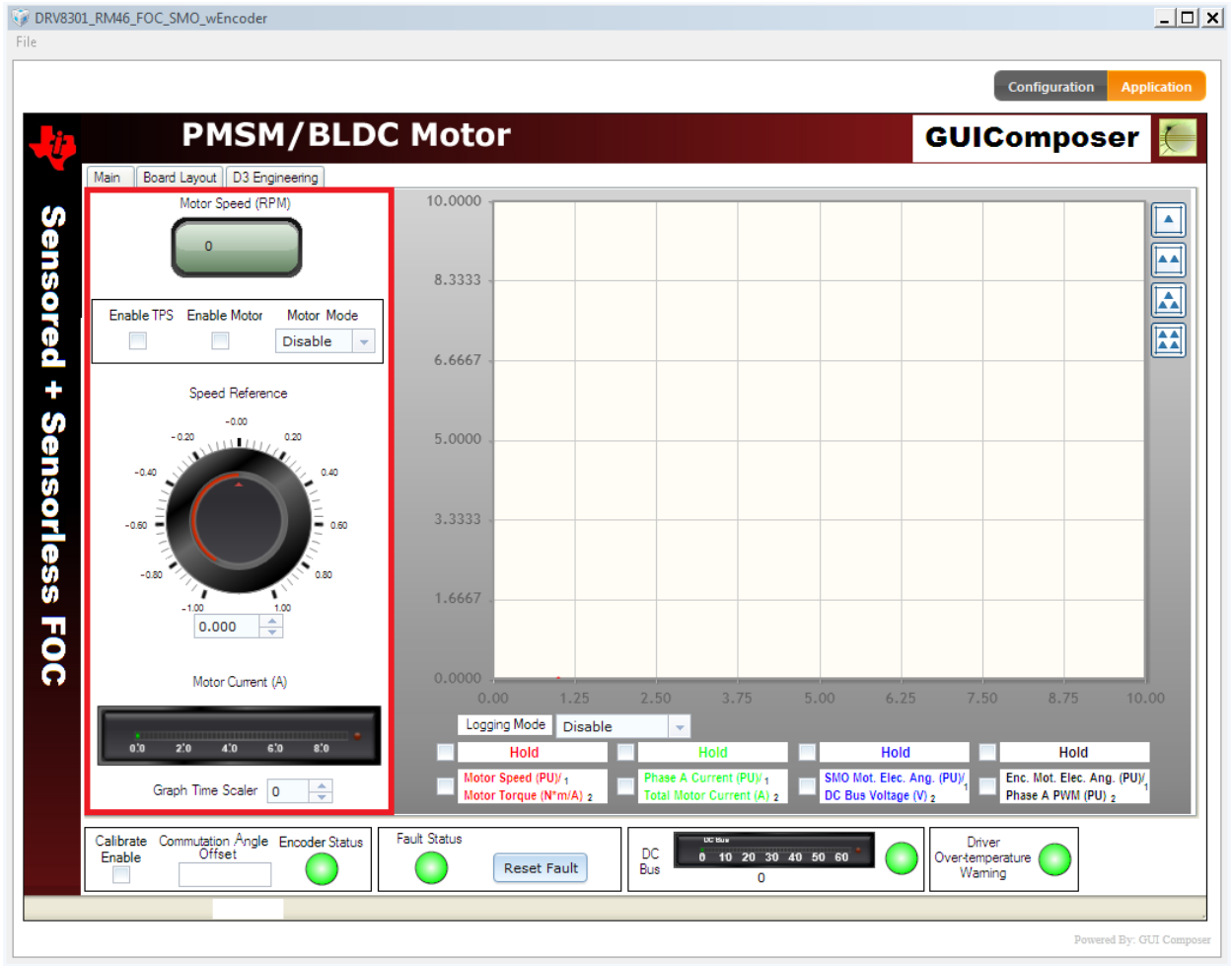


Common Controls

The bottom portion of the **Main** GUI tab contains set-up and status information

- **Calibrate Enable Check Box**: The Calibrate Enable check box in conjunction with the Enable Motor check box is used to calculate the Commutation Angle Offset (see below). To calculate the Commutation Angle Offset, first check the Calibrate Enable check box then check the Enable Motor check box. You should see the motor give approximately one full revolution and the Commutation Angle Offset displayed. At this point the motor has been calibrated you should uncheck the Calibrate Enable check box.

- **Commutation Angle Offset**: This watch window displays what the Commutation Angle Offset for the connected motor is. Every motor will have a different Commutation Angle Offset. The Commutation Angle Offset should remain constant through the GUI session. Calculation of the commutation angle offset only done once per GUI session.
- **Encoder Status**: The on-screen LED will turn red whenever there is a fault signaled by the Encoder program code. If the encoder on-screen LED red, ensure that the motor encoder pins are properly connected to the DRV83xx board. To reset this fault disable and re-enable the Enable Motor and Calibrate Enable check box.
    - if you switch the motor phasing (UVW to UWV, etc.) so that motor rotates the opposite direction of the positive counting direction of the encoder you will produce an Encoder Fault. You can fix this by connecting the phases properly or by changing the gui variable (and using the new .out; or through CCS Expressions View) of gGUIObj.Motor_Enc_Dir from 1 to -1.
- **Fault Status**: The on-screen LED will turn red whenever there is a fault signaled by the DRV83xx. To reset this fault ensure Enable Motor check box is unchecked and push the Reset Fault button.
- **DC Bus Voltage**: The measured DC bus voltage is displayed both digitally and graphically. The on-screen LED can take three states depending on whether the DC bus is in or out of range.
    - **Yellow**: DC bus is below or above the minimum value
    - **Green:** DC bus is within limits
- **Driver Over-Temperature Warning:** The state of the DRV83xx OCTWn pin is displayed using an on-screen LED. The LED can take two states:
    - **Yellow:** The DRV83xx device is running at abnormal operating temperature.
    - **Green:** The DRV83xx device is running at normal operating temperature.



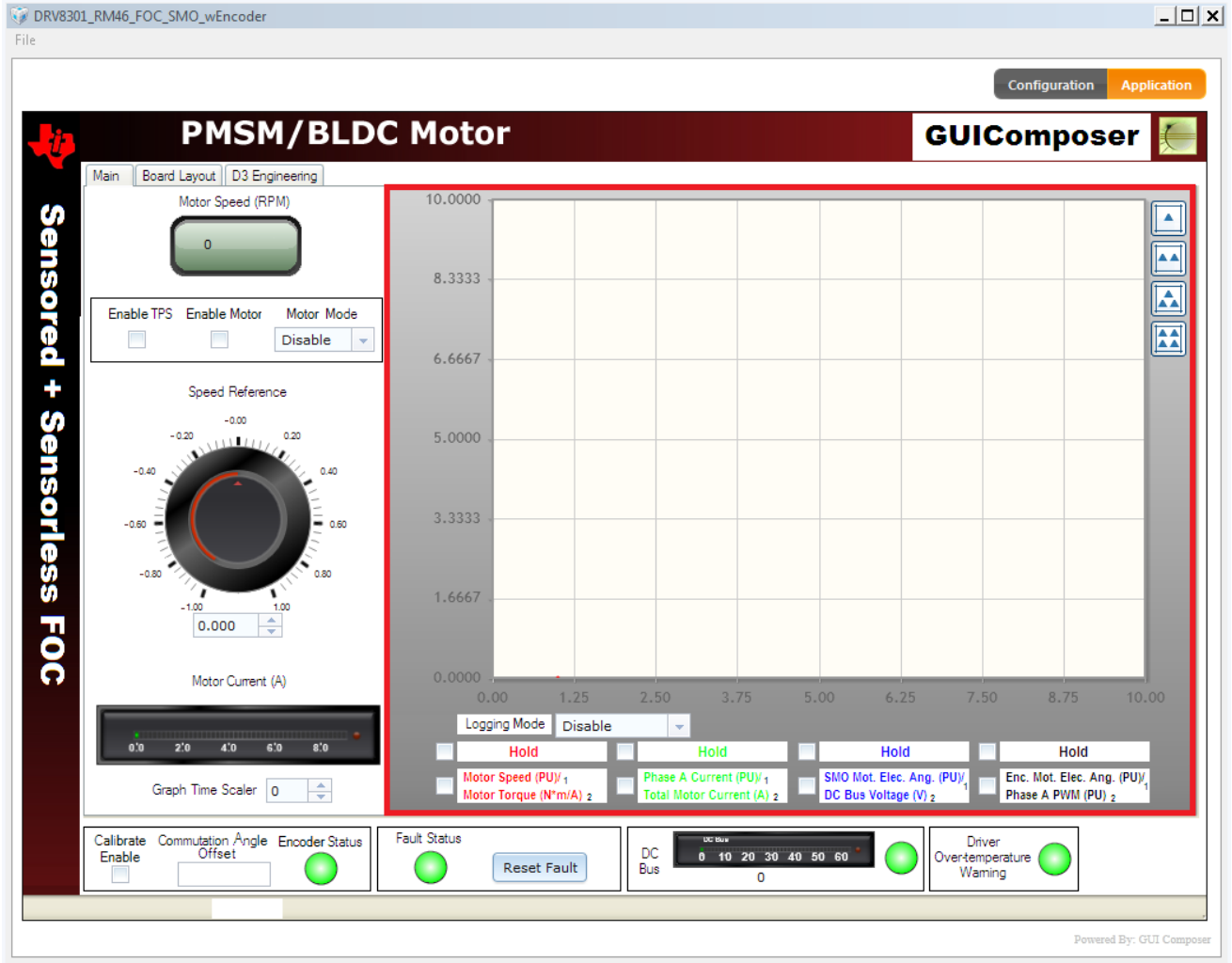
Main Interface Panel

The left portion of the **Main** GUI tab contains controls to vary the motor set-points.

- **Enable TPS Check Box**: The Enable TPS check box is for use with controlCARDs that use the TPS6538x Integrated Safety and Power Companion device, such as the TMDXRM46CNCD. Prior to enabling the motor or doing calibration, the TPS device must be enabled by checking this box in order to safely begin its system watchdog functions and, therefore, proper system operation.
- **Enable Motor Check Box**: The Enable Motor check box is used to start or stop the motor from running. Before this check box can be used as a regular motor-on and motor-off check box, the Commutation Angle Offset must be calculated through the Calibrate Enable check box (see above).
- **Motor Mode Drop Down Box:** Allows the selection of two different control modes
  - **Disable:** This mode is the default Motor Mode and the motor should be disabled. To start the motor Motor Mode drop down box should be set to either Torque or Speed Mode.

- **Torque:** The motor is commutated using the sensored algorithm while the current (torque) is regulated using a PI controller. (Note: An unloaded motor will rapidly accelerate to a very high speed in this mode.)
- **Speed:** The motor is commutated using the sensored algorithm while the motor speed is regulated using a PI controller. The output of the speed loop is a command to the FOC torque controller.

- The **Speed Reference Knob** takes on a separate function for each Motor Mode.
  - **Torque**: The knob adjusts the per-unit (PU) commanded current through the motor. Note: The DRV830x-EVMs have a measureable current range of +/-82.5A which is normalized to a +/-1.0 per-unit (PU) scale.
  - **Speed**: The knob adjusts the per-unit (PU) commanded speed reference
- The actual **Motor Speed (RPM)** is displayed through a digital display.
- The actual **Motor Current (A)** is displayed using a linear gauge. The current should increase with motor load.
- The time scale of the top data present in the graphs can be adjusted by incrementing/decrementing the **Graph Time Scaler**



Graph Section

The right portion of the **Main** GUI tab is the graph section and it contains graphing related controls.

- **Logging Mode Drop Down Box:** Controls what data can be plotted in the graph(s). It changes the variables being written into the data logger buffers. The default value is Disabled mode.
  - **Disable:** This mode is the default Logging Mode and no data should be plotted on the graphs.
  - **Graph One (1):** When Logging Mode is set to this Graph One (1) the Motor Speed, Phase A Current, SMO Motor Electric Angle and Encoder Motor Electric Angle can plotted on the graph(s).
  - **Graph Two (2):** When Logging Mode is set to this Graph One (1) the Motor Torque, Total Motor Current, DC Bus Voltage and Phase A PWM can plotted on the graph(s).
- **Graph data** and **Hold data check box:**
  - **Hold data:** If this check box is set, the data which is being plotted by the respective Graph data check box is paused. If this check box is set then the Graph Data check box

is disabled. This feature is useful when wishing to both isolate and analyze a certain data set in the data logger buffer.

- **Graph data:** If this check box is set, then the data set in the respective data logger buffer is graphed.

- **Number/size graphs:** The number of graphs which are plotting the data in the data logger buffers can be changed dynamically with the foc-encoder-speed GUI. All of the data present in the data logger buffers can be displayed in single graph, plotted in two same/different size graphs, or each can be plotted in its own graph. The graphs can be changed dynamically by clicking on one of the buttons on the right hand side of the graph section.

Shutting Down

1) Once finished evaluating, uncheck the Enable Motor check box to stop the motor.
2) Once the motor comes to a full stop the GUI can be closed.

WARNING

Do not close the GUI until the Enable Motor check box has been unchecked. Closing the GUI without unchecking the Enable Motor check box could leave the processor in an unknown state, causing the system to continue to draw current, possibly damaging the controlCARD, board, host computer, and posing a fire hazard. To avoid potential problems, disconnect the power supply when finished evaluating the board. As the capacitors are charged the PVDD LED (LED1) may remain ON for several seconds. Do not touch the board unless this LED goes OFF**.**