TI Spins Motors

GUI Quick Start Guide:
instaspin_bldc
Sensorless InstaSPIN™-BLDC with commutation, speed, torque, or speed+torque

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.ccxm

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 be always 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 GUI Composer 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

instaspin_bldc

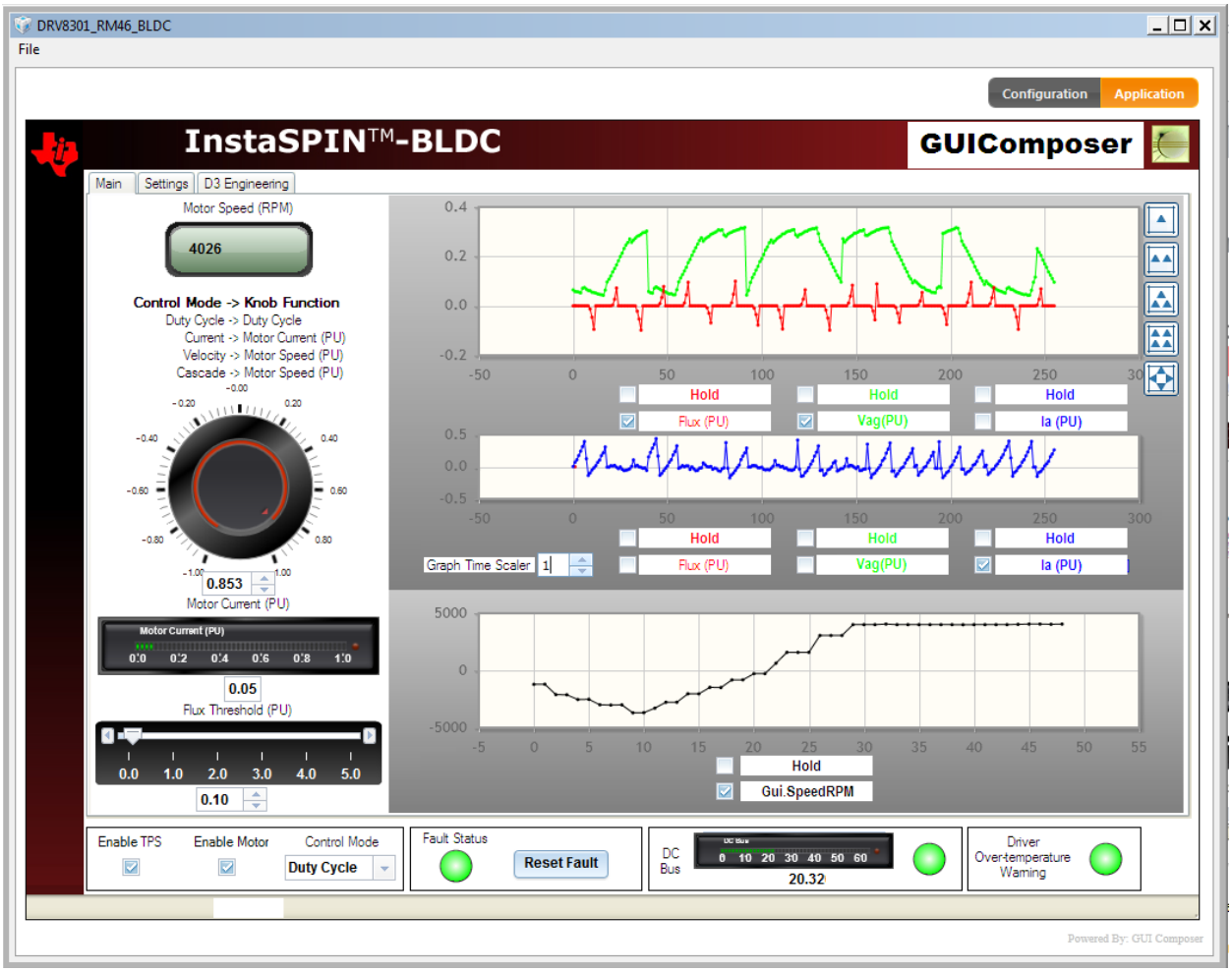
Sensorless InstaSPIN™-BLDC with commutation, speed, torque, or speed+torque

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

Set-up & Status

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

- **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**: Used to start or stop the motor from running.
- **Control Mode Drop Down Box:** Allows the selection of four different control modes
  - **Duty Cycle:** The motor is commutated using the sensorless algorithm but is driven in an open-loop duty cycle mode.
  - **Current:** The motor is commutated using the sensorless 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.)
  - **Velocity:** The motor is commutated using the sensorless algorithm while the motor speed is regulated using a PI controller. The output of the speed controller is a PWM duty cycle.
  - **Cascade:** The motor is commutated using the sensorless algorithm while the motor speed is regulated using a PI controller. The output of the speed controller is a motor current command which is regulated by a lower level current PI controller.
- **Fault Status**: The on-screen LED will turn red whenever there is a fault signaled by the DRV830x. 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 two 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 DRV830x OCTWn pin is displayed using an on-screen LED. The LED can take two states:
  - **Yellow:** The DRV830x device temperature exceeds 130°C
  - **Green:** The DRV830x device temperature is below 130°C



Main Panel

The left portion of the **Main** GUI tab contains controls to vary the motor set-point or view various feedbacks.

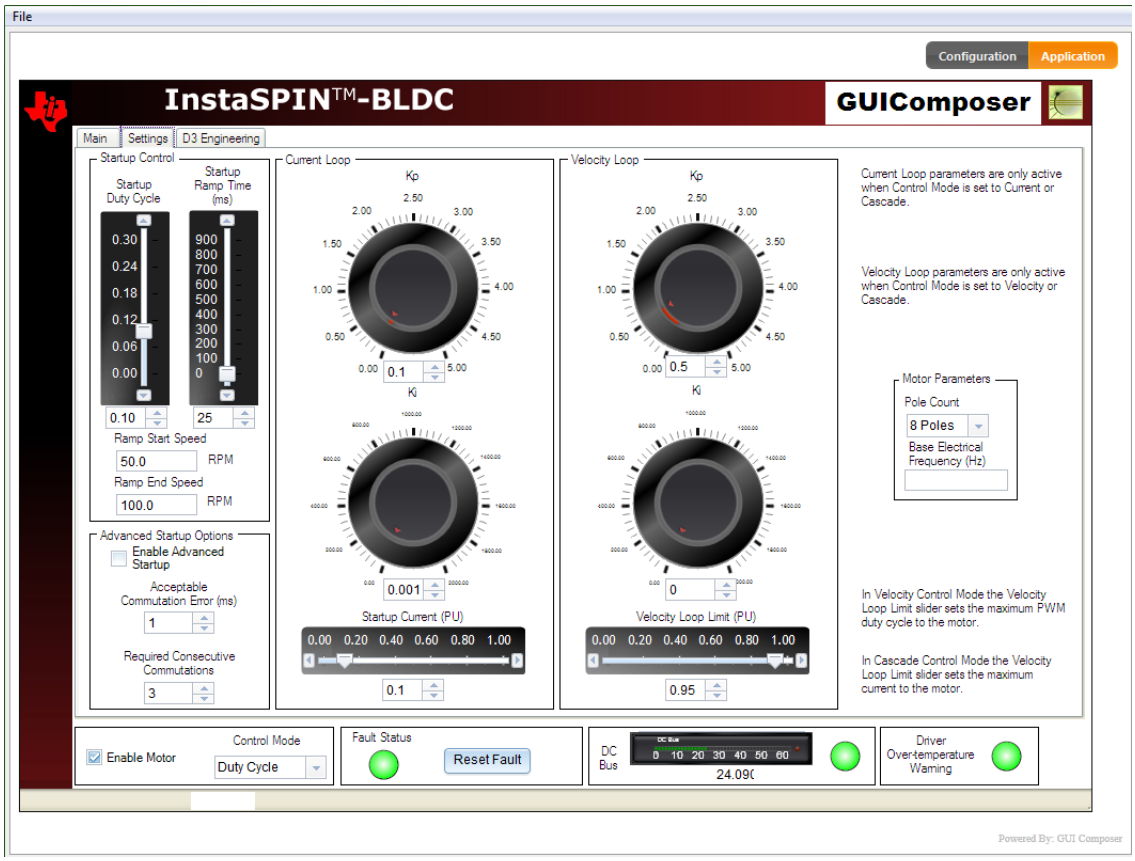
- The **Setpoint Knob** takes on a separate function for each control mode.
  - **Duty Cycle**: The knob adjusts the PWM duty cycle to the motor.
  - **Current**: The knob adjusts the per-unit (PU) commanded current through the motor.
    - **Note**: this is hardware drive board dependent, be careful when using a motor that cannot handle the full scale of the hardware (e.g. 82.5A on the DRV8301)
  - **Velocity**: The knob adjusts the per-unit (PU) motor commanded speed.
  - **Cascade**: The knob adjusts the per-unit (PU) motor commanded speed.
- The actual **Motor Speed (RPM)** is displayed through a digital display.
- The actual **Motor Current (PU)** is displayed using a linear gauge. The current should increase with motor load.

The right portion of the Main tab includes the graphing feedback:

- **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. 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.
    - This is useful to plot variables with similar scales in the same panel while displaying other variables on their own
- The time scale of the graphs can be adjusted by incrementing/decrementing the **Graph Time Scalar**
- Red: Displays the per-unit (PU) integrated motor flux.
- Green: Displays the per-unit (PU) Phase A BEMF waveform.
- Blue: Displays the per-unit (PU) Phase A current waveform.
- Speed can also be displayed if the number of graphs is two or more . This graph runs in a continuous strip chart mode and is not affected by the **Graph Time Scalar**.
- The **Flux Threshold** slider is used to adjust the motor's commutation point



Settings Tab

The **Settings** tab contains parameters affecting motor startup and control loop tuning.

- **Startup Control**: These parameters control how the motor initially ramps up under forced commutation. It is necessary to get the motor spinning and generating some BEMF in order for the sensorless algorithm to latch on and take over commutation.
  - **Startup Duty Cycle**: Sets the constant PWM duty cycle given to the motor during the forced commutation ramp up phase.
  - **Startup Ramp Time**: Sets the time taken to complete the forced commutation ramp up phase.
  - **Ramp Start Speed**: Sets the initial speed for the forced commutation ramp up phase.
  - **Ramp End Speed**: Sets the final speed for the forced commutation ramp up phase.
- **Motor Parameters**
  - **Pole Count**: Choose the number of poles for the motor under test.
  - **Base Electrical Frequency**: Sets the scaling of per-unit (PU) speed to motor electrical speed. For the settings shown above:

$$200Hz * \frac{1 mechanical\_rev}{\frac{8 poles}{2} electrical\_rev} * \frac{60\sec}{1\min} = 3000RPM \Rightarrow 1.0PU$$

- **Current Loop**: Contains parameters associated with the current control loop. These parameters are only active when Control Mode is set to Current or Cascade. Those are the only modes which make use of the current loop.
  - **Kp**: Sets the proportional gain for the current controller.
  - **Ki**: Sets the integral gain for the current controller.
  - **Startup Current**: When current control is active the motor is driven with constant current rather than a constant duty cycle during the forced commutation ramp up phase. This slider sets the current for this phase.
- **Velocity Loop**: Contains parameters associated with the velocity control loop. These parameters are only active when Control Mode is set to Velocity or Cascade. Those are the only modes which make use of the velocity loop.
  - **Kp**: Sets the proportional gain for the velocity controller.
  - **Ki**: Sets the integral gain for the velocity controller.
  - **Velocity Loop Limit**
    i. In Velocity Control Mode the Velocity Loop Limit Slider sets the maximum PWM duty cycle to the motor.
    ii. In Cascade Control Mode the Velocity Loop Limit Slider sets the maximum current to the motor.

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**.**