Low-Side Driver Module Kit User's Manual

D000030 Rev E

July 17, 2007

Drivven ,Inc. • 12001 Network Blvd, Bldg E, Suite 110 • San Antonio, Texas 78249 • USA
Phone : 210.248.9308
Web : www.drivven.com , E-mail : info@drivven.com

Introduction

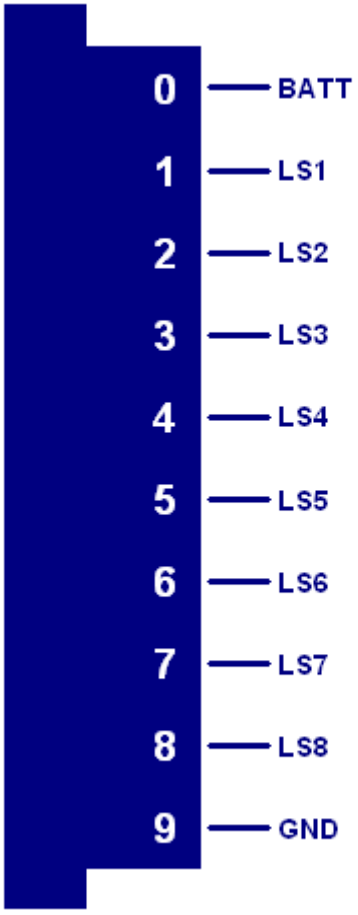
The Low-Side Driver Module Kit provides a CompactRIO (cRIO) module for driving general purpose automotive solenoid valves. The kit includes a LabVIEW FPGA VI for controlling all driver channels independently. Each general purpose low-side solenoid driver is PWM controlled, capable of 0-100% duty cycle operation. Channels 5-8 are capable of driving low impedance solenoids with a programmable peak hold current profile. The high and low current limits may be set independently for each channel.

Features:

- 8-channel general purpose low-side solenoid driver
  - 1.2A continuous duty (channels 1-4)
  - 2A continuous duty (channels 5-8)
  - Individual dual current limit profiles for channels 5-8
    - 50mA to 4A high current limit range (during peak time)
    - 50mA to 2A low current limit range
    - 2 mA current limit resolution
  - PWM controlled with 0-100% duty cycle operation
  - Open/short circuit detection and reporting with short circuit disable
- External power supply of 6-16V (Rev C)
- External power supply of 6-32V (Rev D and later)

Note: Channels 5 & 6 have a combined current rating of 3A continuous.
Note: Channels 7 & 8 have a combined current rating of 3A continuous.

Pinout



Hardware

The Low-Side Driver Module Kit provides eight general purpose low-side solenoid drivers in a National Instruments CompactRIO module.

Powering the Module

The Low-Side Driver module requires power from two different sources.

One source is from the CompactRIO backplane male high density D-Sub 15-pin (HD15) connector which mates with the module's female HD15 connector. This power source provides a regulated 5 volts and ground to various digital logic functions within the module. The CompactRIO 5V source is active whenever the CompactRIO or R-Series Expansion Chassis is properly powered. The module should only be powered at the HD15 connector by plugging it into a CompactRIO or R-Series Expansion Chassis. The module's HD15 connector should not be connected to any other device.

Another required power connection is at the external screw terminal connector. The terminals are labeled BATT (0) and GND (9). Typical power sources will be from automotive 12V or 24V battery systems. However, the module can accept power from a range of 6V to 32V.

The external battery power ground is completely isolated, within the module, from the CompactRIO 5V supply ground. However, the external battery ground and the CompactRIO ground may be connected externally.

The module will not be recognized by software without both power supplies active.

Note: Modules of revision C and earlier have a maximum power input of 16V.

Warning: The external battery supply input terminals are not reverse voltage polarity protected. Such protection would compromise certain features of the module. Connecting power to the module in reverse polarity will certainly damage the module.

Platform Compatibility

CompactRIO modules from Drivven are compatible within two different platforms from National Instruments. One platform is CompactRIO, consisting of a CompactRIO controller and CompactRIO chassis as shown in Figure 1a below.

Figure 1a. CompactRIO platform compatible with Drivven CompactRIO modules.

The other platform is National Instruments PXI which consists of any National Instruments PXI chassis along with a PXI RT controller and PXI-78xxR R-Series FPGA card. An R-Series expansion chassis must be connected to the PXI FPGA card via a SHC68-68-RDIO cable. The CompactRIO modules insert into the R-Series expansion chassis. This platform is shown in Figure 1b below.

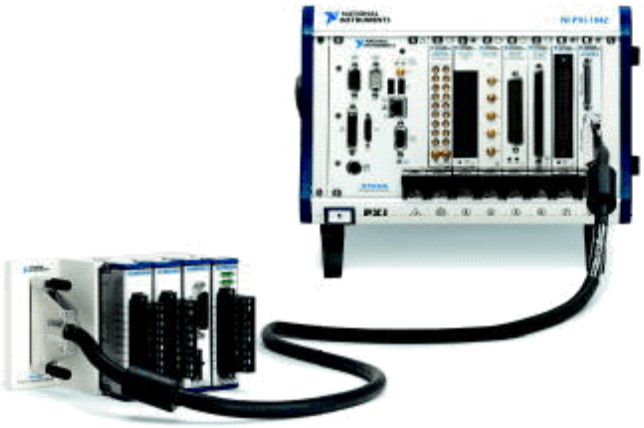


Figure 1b. PXI platform compatible with Drivven CompactRIO modules.

Drivven CompactRIO modules are not compatible with the National Instruments CompactDAQ chassis.

Drivven CompactRIO modules REQUIRE one of the hardware support systems described above in order to function. The modules may not be used by themselves and/or interfaced to third party devices at the backplane HD15 connector. These efforts cannot be supported by Drivven or National Instruments.

Low-Side Drivers

The low-side drivers are capable of driving a wide variety of automotive relays and actuators. Examples of actuators include EGR valves, turbo wastegate valves, fuel pressure regulators, and transmission solenoid actuators.

Channels 1-4 are low-side switches for inductive loads which can drive up to 1.2 amps.

Channels 5-8 are programmable current-limited drivers which can drive up to 4 amps during PeakTime and 2 amps in continuous hold mode. These channels are useful for low-impedance solenoids. Channels 5-8 operate identically to PFI channels 1-4 of the PFI driver module. One useful application of these channels would be for typical low impedance fuel injector valves which need to be operated according to duty cycle as opposed to crankshaft position. They may also be used for common rail fuel pressure regulator valves which often require current drives from 0 to 2A.

WARNING: Do not attempt to drive low impedance injectors or solenoids with the general purpose low-side driver channels 1-4. Channels 1-4 are only rated for 1.2 amps.

Each channel is independently controlled within the module via pulse-width modulation (PWM). PWM duty cycle from 0 to 100% is possible. Software on the RIO FPGA device communicates several parameters for each channel to operate. Those parameters are Enable, Period and PulseWidth. All parameters are communicated serially at 125 usec intervals. Each PWM controller utilizes a 20-bit timer operating at 4 MHz. This provides a resolution of 250 nsec and a minimum PWM frequency of 3.8 Hz. Since 0% and 100% duty cycles are possible, relays and actuators may be controlled in continuous on or off states.

Each low-side driver channel automatically tests for open circuit conditions. When the low-side switch is commanded open, the circuit is tested for an open condition. Upon detection of an open circuit, an open circuit error flag will be reported. The open circuit flag is automatically cleared when the condition is removed.

Each low-side driver channel automatically protects itself from an accidental short circuit to battery. When the low-side switch is commanded closed, the circuit is tested for a short. Upon detection of a short, a short circuit error flag will be reported for the channel and the channel will be disabled until software clears the error flag. The short circuit error flag for each channel will not be cleared while the switch is being commanded closed. It is strongly recommended that the user not continually clear the short circuit error flags, as this may lead to driver circuit damage in the event of an actual short circuit. Even though the short circuit detection and disable feature is present, it is strongly recommended that the wiring harness include a 7 amp fuse for four solenoids as shown in Figure 2.

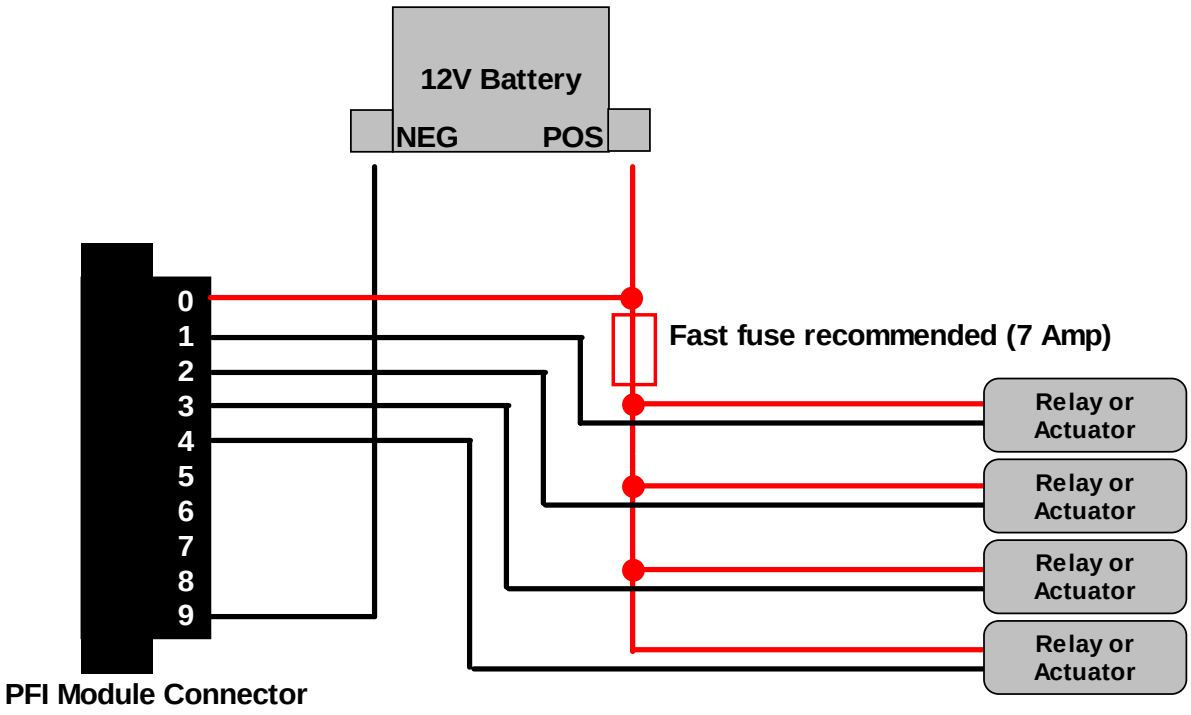


Figure 2. Connecting general purpose solenoids to the low-side driver channels

Current Limit Features of Channels 5-8

Channels 5-8 have the ability to limit the current according to programmable current setpoints. Each channel has a high and low current limit. The high current limit is effective when the drive pulse starts and extends for the time period specified by PeakTime for each channel. After the peak time expires, the low current limit setting takes effect until the pulse ends.

The high current limit setting has a useful range of 50mA to 4A. The low current limit setting has a useful range of 50mA to 2A. The ranges must not be exceeded or damage could occur to the channel.

Warning: Channels 5 and 6 have a combined total continuous current limit of 3A. Channels 7 and 8 have a combined total continuous current limit of 3A.

During normal operation, channels 5-8 will disable the channel when a short circuit is detected and will also set a short circuit flag in software. However, when current limit settings are below 700mA, the channels will falsely report a short circuit and disable the channel until the short circuit flag is cleared. When current limits are desired below 700mA then the ShortReportDisableX boolean must be set to TRUE so that short circuits will not be reported and the channel will not be disabled. However, the channel is still protected from short circuits because the current limit control feature is fast enough to prevent excessive currents.

Software

The Low-Side Driver Module Kit is provided with a LabVIEW FPGA VI for commanding each low-side switch individually via PWM.

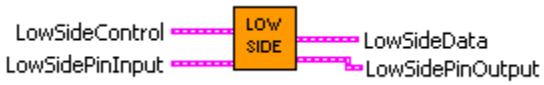


Figure 3. lowside_revx.vi icon with leads.

VERY IMPORTANT NOTES:

The FPGA VI requires:

- LabVIEW 8.2 Full Development or later
- LabVIEW RT Module 8.2 or later
- LabVIEW FPGA Module 8.2 or later
- NI-RIO 2.1 or later

The FPGA VI must be placed within a Single Cycle Loop (SCL) of a LabVIEW FPGA block diagram. The SCL must execute at the default clock rate of 40 MHz.

The FPGA VI requires a pre-synthesized netlist file having a matching name and an extension of .ngc. The netlist file must be located in the same directory as the matching VI.

The FPGA VI requires the installation of a special CompactRIO module support package called cRIO-generic. Please follow the steps below to install the cRIO-generic package:

1. Confirm that LabVIEW is closed.
2. Add the line **`cRIO_FavoriteBrand=generic`** to the LabVIEW INI file. The LabVIEW INI file is typically found at C:\Program Files\National Instruments\LabVIEW 8.0\LabVIEW.ini.
3. Upon restarting LabVIEW, the cRIO-generic module will appear in the list of available modules within the LabVIEW FPGA "New C Series Module" configuration dialog. All Drivven CompactRIO modules require adding an associated cRIO-generic module to your LabVIEW Project. Within the Project Explorer, A cRIO-generic module can be added to a PXI FPGA expansion chassis or a CompactRIO chassis. This is best understood by observing an example project provided with your module kit.

WARNING!

When writing values to an FPGA cluster from the RT level, every parameter within the cluster must be explicitly written. If any parameter is not explicitly written, then the default value for that particular data type will be used. This could cause unexpected behavior.

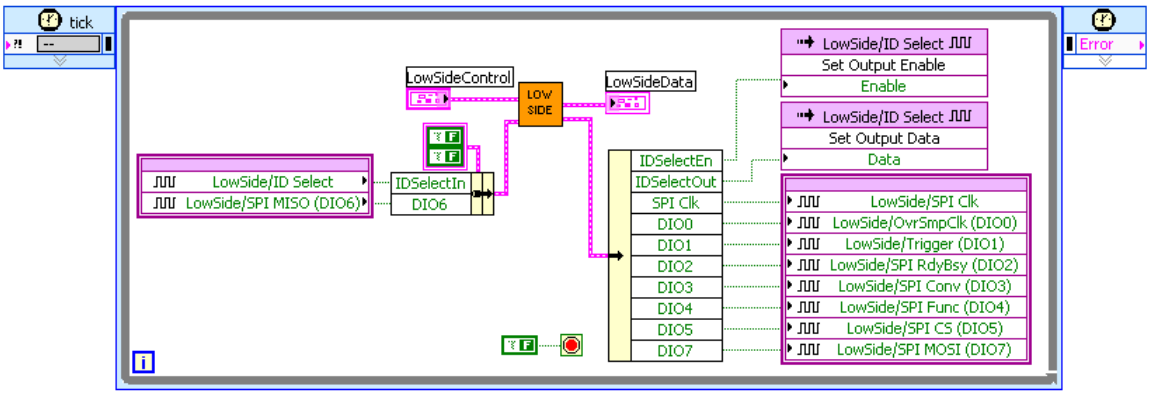


Figure 4. Example block diagram implementation of Low-Side VI.

LowSidePinInput (Cluster)
These boolean controls must be connected to their corresponding FPGA I/O Node input item.

LowSidePinOutput (Cluster)
The boolean indicator named IDSelectEn must be connected to a Set Output Enable method of an FPGA I/O Method Node. The boolean indicator named IDSelectOut must be connected to a Set Output Data method of an FPGA I/O Method Node. The remaining boolean indicators must be connected to their corresponding FPGA I/O Node output item.

WARNING!
Great care must be taken to ensure that LabVIEW FPGA I/O node output items are only wired from a single logic source. There is no circumstance in which FPGA I/O node output items should be driven by multiple sources when interfacing to cRIO modules, otherwise strange behavior or module damage could result. Two LabVIEW FPGA code snippets are shown below which illustrate this issue. Figure 5a shows the correct implementation of FPGA I/O node blocks, whereas a group of three outputs to an ADCombo module are controlled while another group of eight outputs to a Spark module are controlled. Each of the output items are selected only once in the entire block diagram. On the other hand, figure 5b shows a coding mistake that should be avoided. Notice the group of ADCombo output items where a Spark module output item is selected instead of the correct ADCombo module output item. The same Spark module output item is also selected in the Spark group below. This means that the Spark (DIO5) output is being driven by two different logic sources and will cause strange behavior of the spark module, or possible damage.

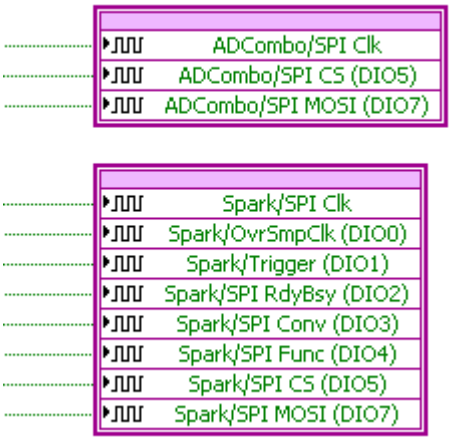


Figure 5a. Representative FPGA output nodes for ADCombo and Spark modules with correct output item selection.

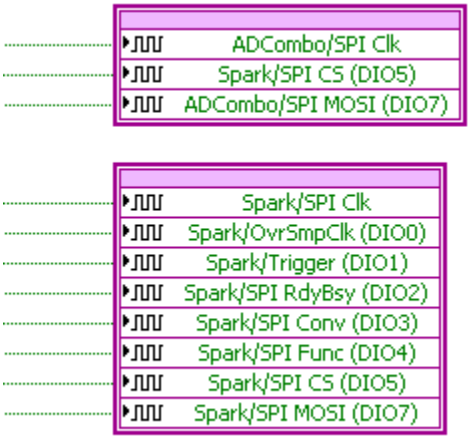


Figure 5b. Representative FPGA output nodes for ADCombo and Spark modules with incorrect output item selection within the ADCombo output node. The Spark (DIO5) output is selected in multiple nodes and therefore being driven by multiple sources. This will cause strange behavior or damage to the spark module.

One way to help prevent such coding mistakes is to prefix all FPGA I/O item names with an appropriate unique module name via the FPGA I/O creation dialog or via the project explorer, after the I/O item is created. This will make the coding mistake recognizable from the block diagram. Another way this situation can be prevented, even when a coding mistake is made, is by making sure that all FPGA output node items are configured to "Arbitrate if Multiple Accessors Only." When outputs are configured this way and they are used within a Single Cycle Loop (as is required by Drivven cRIO module kits), then a compile error will be generated if multiple sources are driving FPGA output node items. Then appropriate corrective action can be taken. FPGA output node items can be configured via the FPGA I/O properties dialog, by right clicking on the FPGA I/O item within the project explorer. FPGA output node properties should be set according to the following dialog screen shot.

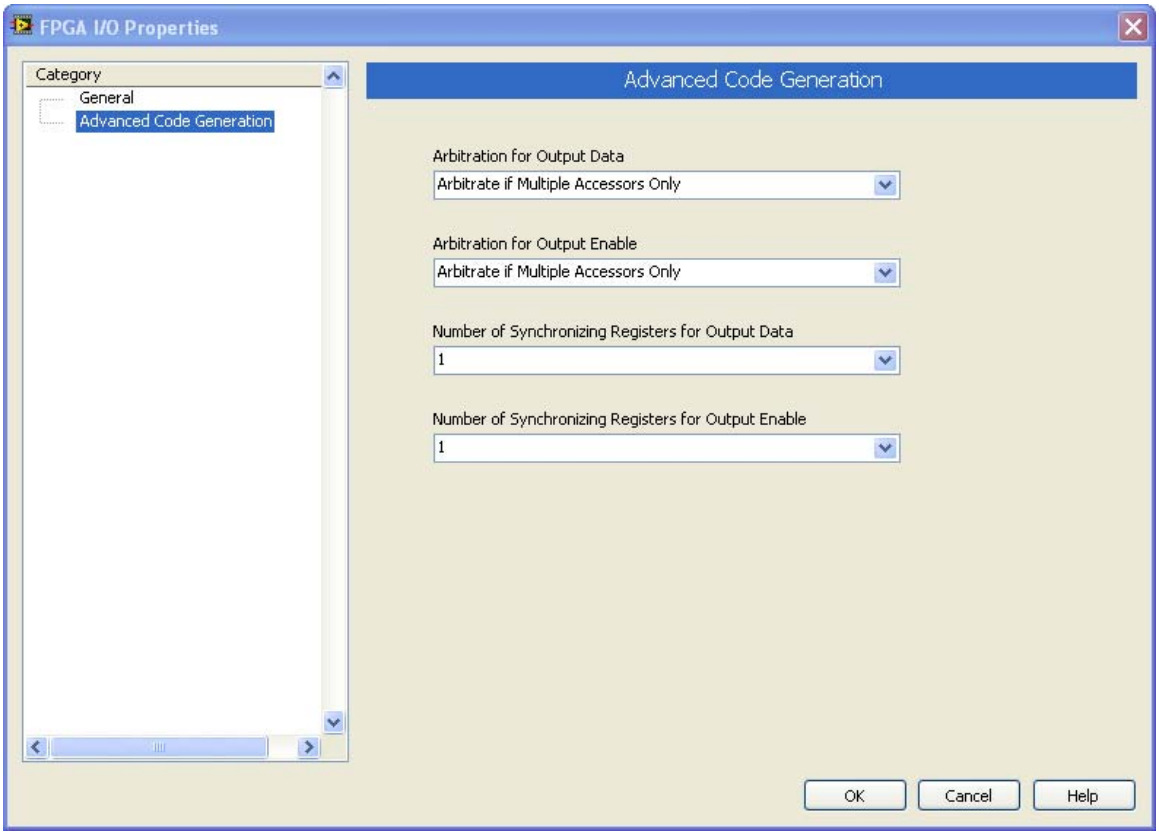
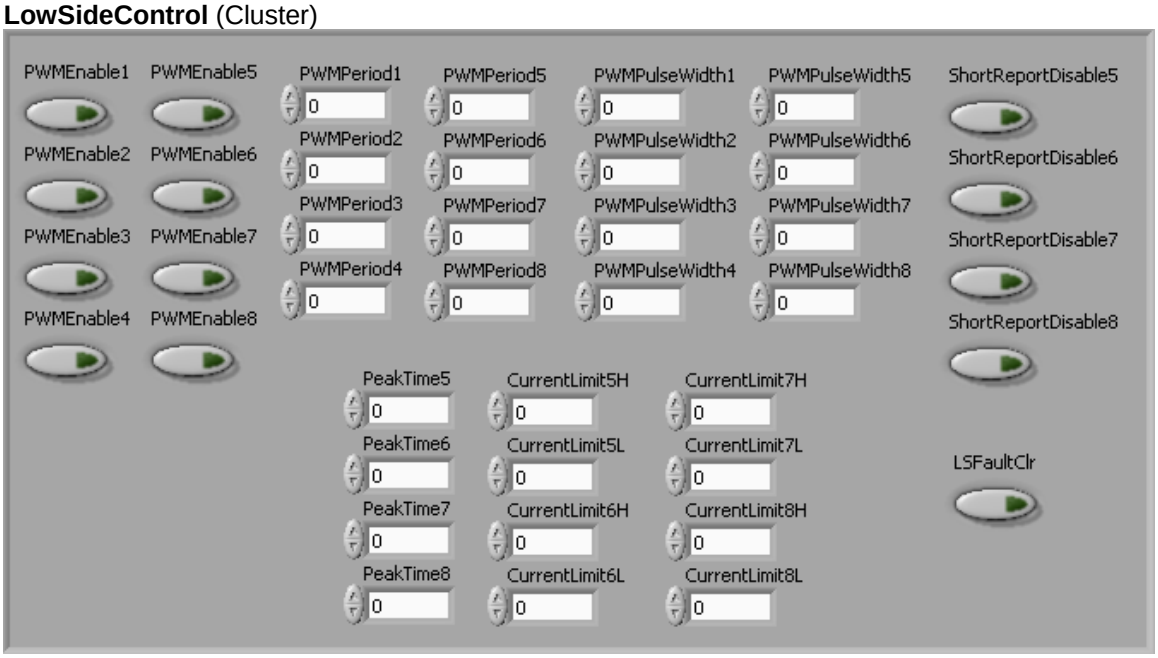


Figure 6. FPGA I/O Properties dialog configuration for cRIO modules.

LowSideControl (Cluster)



PWMEnableX (boolean): When TRUE, the PWM controller is enabled. When FALSE (default), the PWM controller is disabled.

PWMPeriodX (uint32): The time period between leading edges of low-side switch closure. PWMPeriod is entered in terms of 4 MHz clock ticks and is internally limited to 20 bits. This provide a maximum period of 0.26 seconds or a frequency of 3.8 Hz, and a resolution of 250 nsec. Values larger than 20 bits will roll over from zero.

PWMPulseWidthX (uint32): The time during each PWMPeriod in which the low-side switch is closed. The low-side switch is closed from the beginning of PWMPeriod until PWMPulseWidth expires. PWMPulseWidth is entered in terms of 4 MHz clock ticks and is internally limited to 20 bits. While PWMPulseWidth is 0 the low-side switch will remain open. While PWMPulseWidth is greater than or equal to PWMPeriod, the low-side switch will remain closed. Values larger than 20 bits will roll over from zero.

Drivven provides a utility VI, named lowside2ticks.vi, which can be implemented at the LabVIEW RT level for performing the conversion from frequency in hertz and duty cycle in percent to PWMPeriod ticks and PWMPulseWidth ticks. This VI icon is shown in Figure 7.

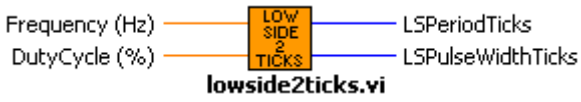


Figure 7. Lowside PWM conversion VI.

LSFaultClr (boolean): When TRUE, clears all low-side driver short circuit fault flags.

PeakTimeX (uint8): Determines the length of time that the driver circuit will use CurrentLimitXH as the current limit. PeakTime is entered as a uint8 value in terms of 4 MHz clock ticks divided by 64. Therefore, the maximum value entered for PeakTime of 255 equates to an effective value of 16320. At 4 MHz this corresponds to a maximum delivered peak time of 4.08 msec. The following equation applies:

PeakTime(uint8 ticks) = PeakTime(msec) * 62.5

Drivven provides a utility VI which can be implemented at the LabVIEW RT level for performing this calculation. The VI named peaktime2ticks.vi can be used to convert PeakTime in milliseconds to uint8 ticks. This VI icon is shown in Figure 8.

Figure 8. PeakTime to Ticks conversion VI.

PeakTime only applies to low-side channels 5-8.

CurrentLimitXH (uint16): The current limit setting in effect during PeakTimeX. CurrentLimitXH should NOT be set to a value greater than 2000 which corresponds to approximately 4A. Higher values may damage the channel.

CurrentLimitXL (uint16): The current limit setting in effect after PeakTimeX expires. CurrentLimitXL should NOT be set to a value greater than 1000 which corresponds to approximately 2A. Higher values may damage the channel.

Drivven provides a utility VI which can be implemented at the LabVIEW RT level for calculating the correct values for CurrentLimitXH and CurrentLimitXL based on requested current limits in amps. The calculations can also be scaled and trimmed to improve current accuracy. The VI named lowside_current_ctrl.vi icon is shown in figure 9. Drivven strongly recommends using this VI at all times for setting current limit values.

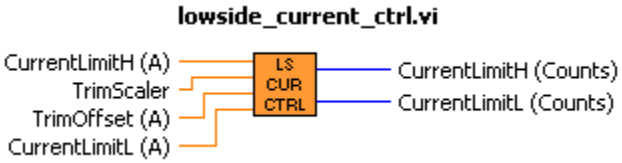


Figure 9. Current limit conversion VI.

PWM Controller Notes:
The control values for the independent PWM controllers can be updated at any time by the CPU. The FPGA updates the control values at 125 usec intervals. The PWM controllers ensure glitch free operation of the PWM signal to the low-side drivers. For example, if a PWM controller pulse width is updated to a larger value immediately after PWMPulseWidth expires, the PWM command to the driver will not be asserted again until the next period.

LowSideData (Cluster)

LSFaultOpen (boolean): Set to TRUE when an open circuit is detected for the low-side driver. The flag is cleared when the condition is removed.

LSFaultShort (boolean): Set to TRUE when a short circuit is detected for the low-side driver. A short circuit automatically disables the channel until the flag is cleared by the corresponding LSFaultClr.

Examples

The following screen capture in Figure 10 shows a LabVIEW FPGA block diagram with the Low-Side VI used for general purpose solenoid actuator control. This FPGA application is entirely contained within a single cycle loop, clocked at the required 40 MHz. The PinInput and PinOutput clusters are wired to LabVIEW FPGA I/O pins which are configured for a cRIO controller chassis or a cRIO R-Series expansion chassis. Refer to the LabVIEW FPGA documentation for details about configuring cRIO I/O pins.

This example makes all of the cluster controls and indicators available to the front panel, and therefore available to the CPU program. If some of the controls within clusters are to be constants and do not need to be made available to the CPU, then cluster bundle and unbundle functions should be used to wire in constants. This will save FPGA resources by limiting the number of visible controls to only those that are necessary at the CPU level.

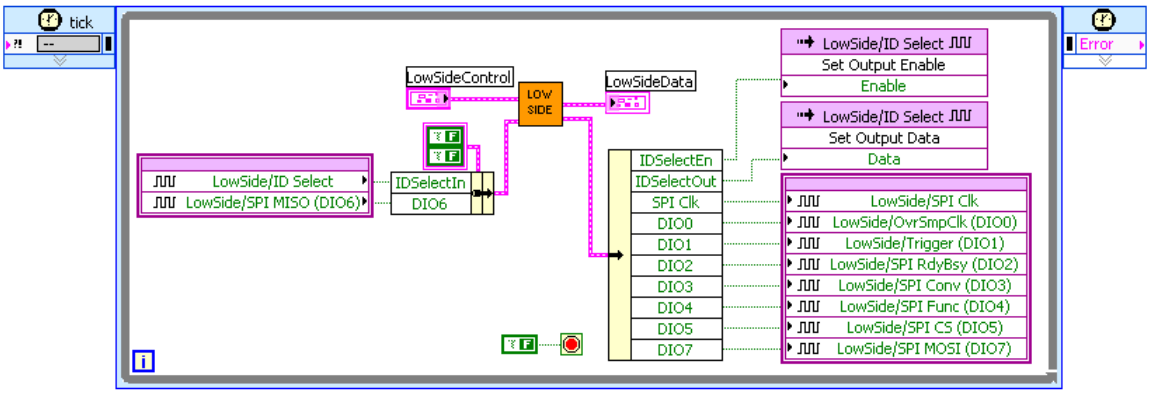


Figure 10. LabVIEW FPGA Block diagram example.