Spark Driver Module Kit User's Manual

D000012 Rev E

January 5, 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

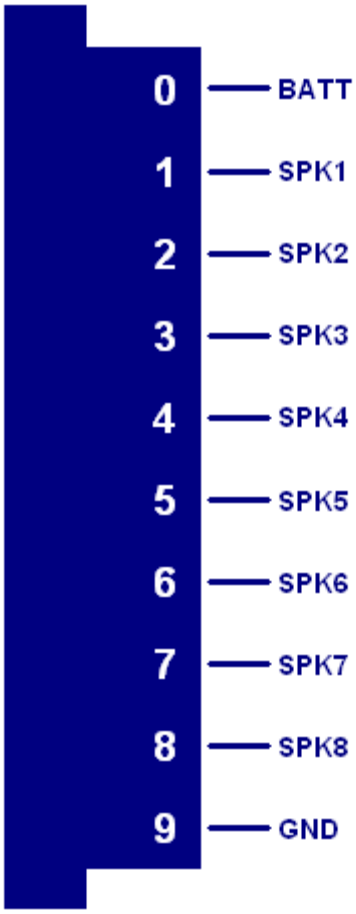
Introduction

The Spark Driver Module Kit provides a CompactRIO (cRIO) module for driving up to eight ignition coils. The kit includes a LabVIEW FPGA VI for controlling all driver channels. Each spark driver is individually controlled for timing and dwell.

Features:

- 8-channel inductive-type ignition coil driver
  - Software configuration for dual output coils (wasted spark)
  - Short circuit protection
  - Internally fuse protected at 10A
  - Module over-temperature protection
- External power supply of 6-16V (Rev C)
- External power supply of 6-32V (Rev D and later)

Pinout



Hardware

The Spark Driver Module Kit provides eight inductive-type ignition coil drivers in a National Instruments CompactRIO module.

Revision Notes:
Revision E of the Spark Driver Module Kit incorporates firmware and software changes that are not compatible with previous versions of the module. Software for revision E will not recognize modules of previous versions. Be sure to use the RevE FPGA VI with the RevE module.

Powering the Module

The Spark 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.

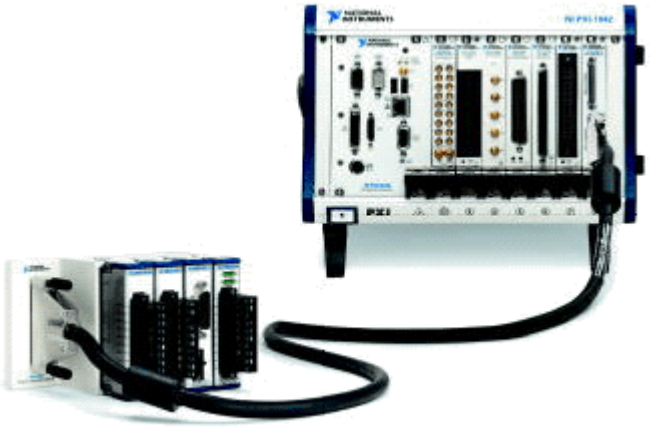


Figure 1b. PXI platform compatible with Drivven CompactRIO modules.

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 will not be supported by Drivven or National Instruments.

Spark Drivers

Each spark driver channel is capable of directly driving a single ignition coil and is independently controlled for timing and dwell. Each channel is capable of driving to a peak of 10 amps. During operation a channel is turned on, driving current through the primary winding of an ignition coil, thus storing magnetic energy. When the channel is turn off, the magnetic field of the primary quickly collapses causing the voltage across the secondary winding to spike, delivering a spark across the gap of a connected spark plug.

The amount of time that the driver channel is turned on, or dwell time, will, in part, determine the voltage that "can" be achieved at the spark plug gap. The gap length also determines the spark voltage. The dwell time required to achieve a certain spark energy will vary greatly with the characteristics of the ignition coil. For example, the dwell time required to achieve 20 kvolts at the spark gap using different coils could range from a few hundred microseconds to four milliseconds. The voltage actually required for combustion will also depend on engine parameters. This spark driver module is intended to be used to drive a wide variety of ignition coils, but the user must understand the proper range of dwell times for the coil used or damage to the coil and driver will occur. It should also be understood that longer dwell time does not necessarily correlate to better combustion. The dwell time only needs to be long enough to achieve a reliable spark under the engine operating conditions and battery voltage. More dwell time, over that which is needed to achieve reliable spark, wastes energy and needlessly heats the driver module.

Determining Dwell Time

The best way to determine required dwell time for a specific coil application is to measure the dwell of the coil within an OEM installation. While performing measurements, be sure to measure the dwell of at least three different battery voltages, if possible, in order to build a table of dwell times versus battery voltage.

It is possible to safely determine the required dwell time for a given ignition coil by performing a simple bench test. The test will require a variable-voltage power supply capable of 5-10 peak amps, an oscilloscope, a current probe and a spark tester device.

Using the built-in simulator of the Engine Position Tracking VI, the user can generate sparks on a test bench by connecting a test coil to a driver channel and a spark tester device to the coil. A spark tester device is a very low cost tool that fits into the boot of a coil or plug lead just like an ordinary spark plug. The gap of the tester is adjusted by a threaded rod with a pointed tip. The body of the spark tester is marked for approximate gap voltages. The tester shown in Figure 2 is made by Thexton (part #404) and is sold for approximately $10 USD.

Figure 2. Thexton adjustable spark tester. Part #404.

WARNING! Automotive ignition coils can generate voltages as high as 50,000 volts! The user should be extremely careful when handling the ignition coil, spark plug or spark testing device. Do not touch the coil or spark tester device while performing spark tests. Do not attempt to adjust the spark tester while performing spark tests. Remove power to the ignition circuit before handling any of these parts. Receiving an automotive ignition spark at any voltage can be very painful. Persons with medical devices or heart conditions should not be involved with testing or installing ignition systems.

Use a current probe to monitor the current through the primary while conducting dwell tests. Most passenger car ignition coils require 3-7 peak amps to achieve a consistent spark. If the coil manufacturer's recommended current rating is not known, then it is good to use the current probe to know if excessive current is required to achieve spark. If more than 10 amps are required to achieve spark then the module will not be able to drive the coil. The module is fuse protected for a total current of 10 amps.

Prepare an application for simulating a running engine using your EPT VI. There is an example FPGA application included with the Spark Driver Module Kit. The Spark Driver Module should be powered externally from the same supply (battery) voltage which feeds the test coil. Connect one side of the coil primary to battery voltage and **be sure to use a fuse in this connection**. We are all prone to making mistakes with connections and software settings! Start with a 5 amp fuse. It is better to blow an external fuse than the fuse within the module. Connect the other side (low side) of the coil primary to the spark driver module channel that you will be using for test. Connect the coil secondary to the spark tester device. This connection will vary with coil and test device. Be sure to connect the other side of the spark tester device back to battery ground.

The user should adjust the spark tester for an initial spark voltage of approximately 20 kvolts. Then starting with a dwell time of approximately 0.1 msec, increase the dwell until spark is achieved. Use the current probe to double check that current limits are not being exceeded as dwell is increased. For example, if the tester device were not connected properly for some reason, the user would see current increasing with dwell, but never see a spark, and appropriate measures could be taken to discover the problem. Once spark is achieved at 20 kvolts, then you can optionally increase the gap to approximately 25 to 30 kvolts and repeat the test. The dwell required for a 20 to 30 kvolt spark is most likely sufficient for achieving reliable combustion at moderate to high loads. Of course this requirement will vary with many different factors, but this test will provide the user with a typical dwell time for the given ignition coil. During engine operation, required dwell time will depend on battery voltage. As battery voltage decreases, dwell time must increase to achieve the same spark energy. The user can perform a bench test with a

variable power supply to determine how dwell time depends on battery voltage. These values can be implemented in a lookup table in LabVIEW using Drivven's table lookup VIs. Do not use a fixed dwell time in your application if you expect the battery voltage to vary. This could lead to weak sparks or blowing the module fuse.

Protections

The spark driver channels are short circuit protected. However, there is no short circuit fault detection reported via software. If a short circuit is present, the driver channel will immediately disable itself during the dwell command. Another driver attempt will be made at the next pulse. The driver channels are software limited to 5 msec of dwell, however, this may be much too long for a given ignition coil. Therefore it is important that the user perform the above described bench tests in order to fully understand the proper range of dwell time for a given ignition coil. Dwell time should also be limited to approximately 25% duty cycle. For example, if an engine is running at 8000 RPM, then the dwell time should be limited to 3.75msec. If wasted spark mode is enabled or re-strike pulses are enabled, the dwell time must be limited even more. For example, if an engine is running at 8000 RPM and wasted spark mode is enabled, then the dwell time should be limited to 1.875 msec. If using double ended coils in wasted spark mode, and dwell times are required to be greater than the maximum dwell calculated for wasted spark mode, then two driver channels can be used to drive the same double-ended ignition coil, with wasted spark mode disabled.

The spark driver module is internally protected from board temperature exceeding 85 degC. When this temperature is exceeded, all driver activity will be shutdown until power is cycled.

Ignition coil primaries should be wired to the Spark Driver Module according to Figure 3. A fast blow fuse is recommended to be used with powering coils individually or with a small group of coils. Better fusing protection will be achieved with a fewer number of coils per fuse.

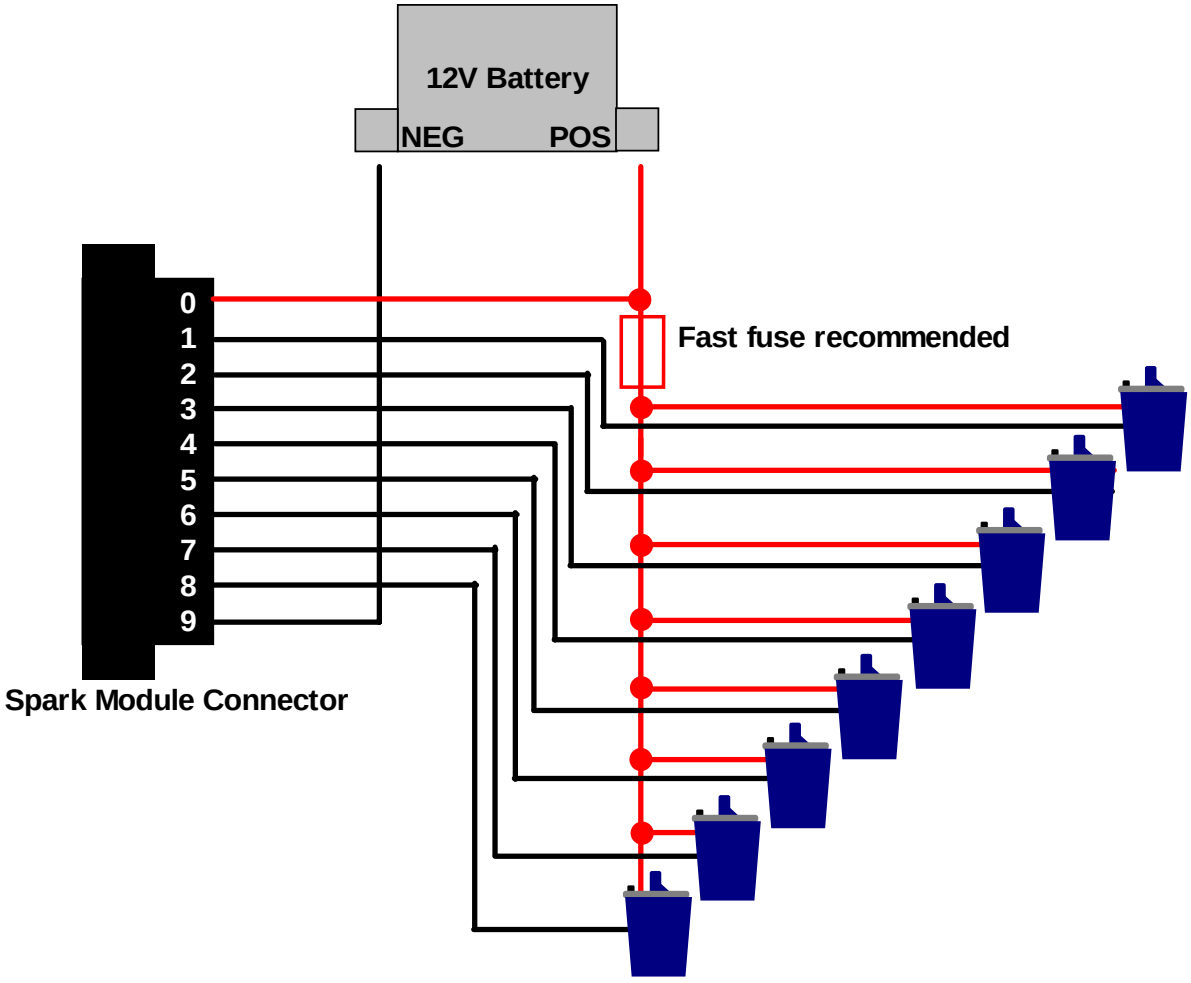


Figure 3. Connecting ignition coils to the driver module.

Software

The Spark Driver Module Kit is provided with 14 different LabVIEW FPGA VIs. Each VI includes an interface for controlling a different number of spark drivers with non-multiplexed or multiplexed internal spark control cores. This gives the user the ability to optimize FPGA resources depending upon maximum engine speed and dwell requirements. There is also a WasteSpark control input which allows the use of dual output ignition coils for pairs of opposing cylinders.

Figure 4 shows the icon which represents all supplied VIs, having identical clustered terminals. However, the contents of the SparkControl cluster will vary.

Figure 4. Spark 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.

Brief Glossary of Terms
CAD: Crank Angle Degrees. 360 CAD per two stroke cycle or one crankshaft rotation. 720 CAD per 4-stroke cycle, or two crankshaft rotations.

CAT: Crank Angle Ticks. Unit of angular measure reported by the CurrentPosition output of the EPT VI. Reported as a specified power-of-two angular ticks per crank tooth. For example, if using the N-M EPT VI, which has an extrapolation of 7, the number of CAT per crank tooth would be 2^7=128, and CurrentPosition would be incremented by 128 CAT from one tooth to the next. If a 60-2 pattern were used, the total number of CAT per crankshaft rotation (cycle) would be 60*128=7680. If the engine was a 4-stroke, the total number of CAT per cycle would be 120*128=15360.

MAX_CAT: Maximum Crank Angle Ticks per engine cycle.

The FPGA VIs supplied with this kit cannot generate spark commands without the supervision of an engine position tracking (EPT) VI from Drivven. The EPT VI provides the necessary output cluster to be wired to the FuelSparkSupervisor input cluster.

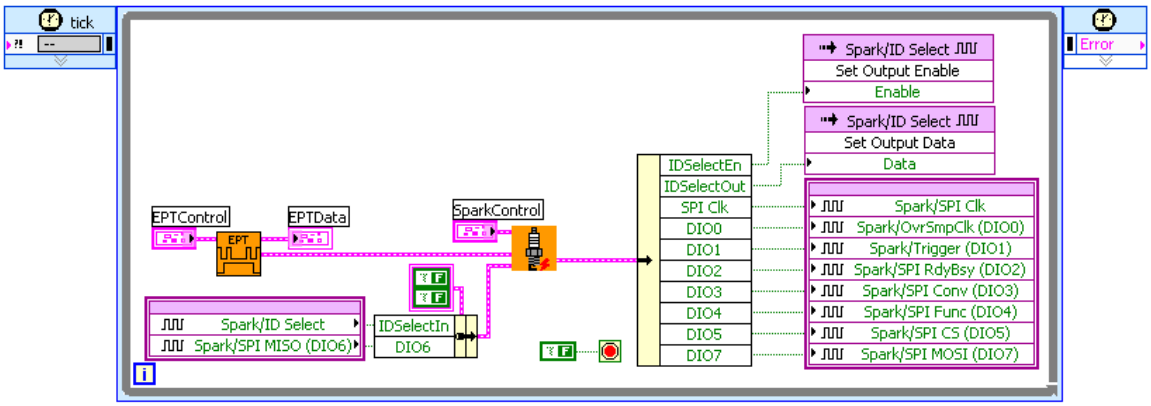


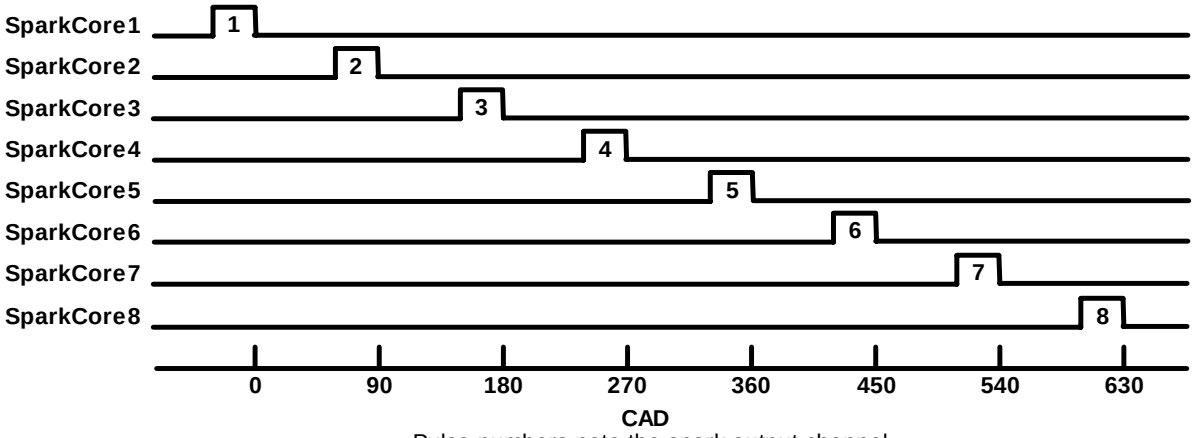
Figure 5. Example block-diagram implementation of Spark VIs.

Table 1 shows a list of all spark VIs included in this kit.

Table 1. Spark VI Configurations

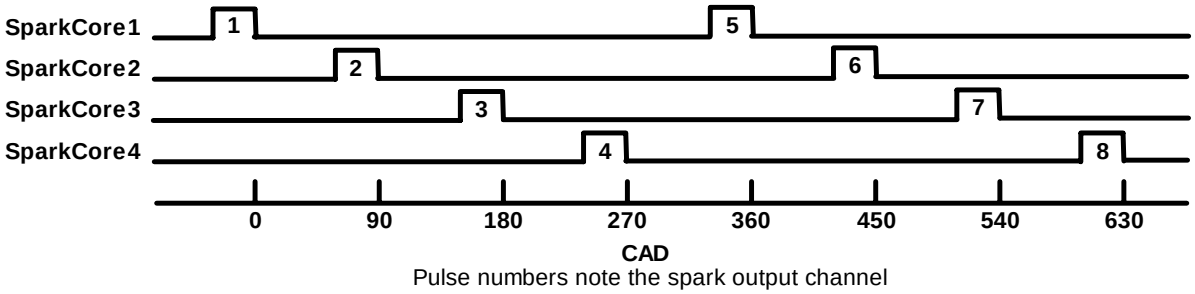
VI Name	Spark Cores Used	Number of Spark Outputs (WasteSpark = FALSE)
spark_vt_1mux1_revx.vi	1	1
spark_vt_1mux2_ revx.vi	1	2
spark_vt_2mux1_ revx.vi	2	2
spark_vt_1mux4_ revx.vi	1	4
spark_vt_2mux2_ revx.vi	2	4
spark_vt_4mux1_ revx.vi	4	4
spark_vt_1mux6_ revx.vi	1	6
spark_vt_2mux3_ revx.vi	2	6
spark_vt_3mux2_ revx.vi	3	6
spark_vt_6mux1_ revx.vi	6	6
spark_vt_1mux8_ revx.vi	1	8
spark_vt_2mux4_ revx.vi	2	8
spark_vt_4mux2_ revx.vi	4	8
spark_vt_8mux1_ revx.vi	8	8

As can be seen from Table 1, there are multiple VIs that support two, four, six and eight spark driver outputs. For example, if an engine control system was being prototyped for an eight cylinder engine with one coil per spark plug, then there are four different VIs to choose from which support eight spark outputs. One of those VIs internally contains eight individual spark cores, each controlling a single output. Another VI internally contains four spark cores, each multiplexed to control two outputs. Another VI internally contains two spark cores, each multiplexed to control four outputs. Finally, another VI contains a single spark core, multiplexed to control all eight outputs. The VI chosen will depend upon the maximum dwell required for the ignition coil and the maximum engine speed. The VI which has 8 spark cores will require much more FPGA resources than the VI which has a single spark core. If the non-multiplexing VI, spark_vt_8mux1_revx.vi, is chosen, then it is possible for all spark outputs to overlap. If a multiplexing VI is chosen, then consideration must be given to dwell and engine speed to ensure that consecutive spark pulses generated by the same spark core will not interfere. The plots in figures 6 - 9 show how the spark cores are multiplexed for each of the four eight-channel VIs.



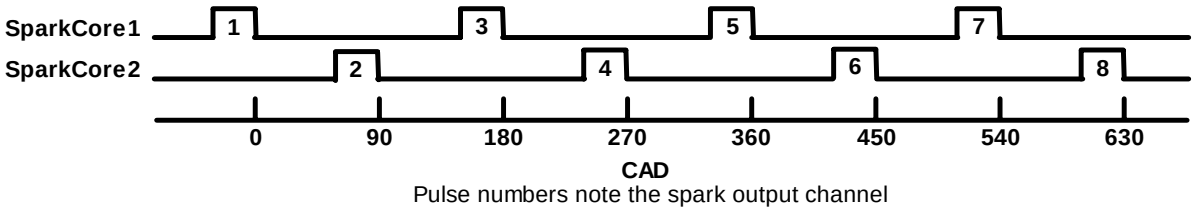
Pulse numbers note the spark output channel

Figure 6. spark_vt_8mux1_revx.vi contains 8 non-multiplexed spark cores.



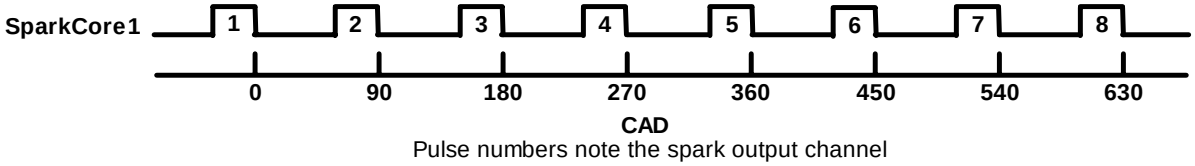
Pulse numbers note the spark output channel

Figure 7. spark_vt_4mux2_revx.vi contains 4 spark cores multiplexed into 2 outputs each.



Pulse numbers note the spark output channel

Figure 8. spark_vt_2mux4_revx.vi contains 2 spark cores multiplexed into 4 outputs each.



Pulse numbers note the spark output channel

Figure 9. spark_vt_1mux8_revx.vi contains 1 spark core multiplexed into 8 outputs.

NOTE: The pulse output numbers shown in the plots above do not correspond to cylinder numbers. There should be no attempt at cylinder number correlation to output channel number. The reason is that spark channels must be configured with timing values in sequential CAD order, NOT cylinder firing order.

In order to determine which spark VI can be used, the following calculations should be performed:

Assume EPT Stroke = 4, MUX = 8 (1mux8), Maximum engine speed = 6000 RPM

MinDwellWindow (sec) = (60 * STROKE) / (2 * MUX * MaxEngineSpeed(RPM))
= (60 * 4) / (2 * 8 * 6000)
= 0.002500 sec

If MinDwellWindow is calculated to be less than the actual dwell required by the ignition coil, then a VI should be selected with less multiplexing and the calculation should be performed again using the corresponding MUX value. Do not forget to consider minimum battery voltage when determining maximum dwell required by the coil.

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

SparkPinOutput (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 10a 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 10b 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.

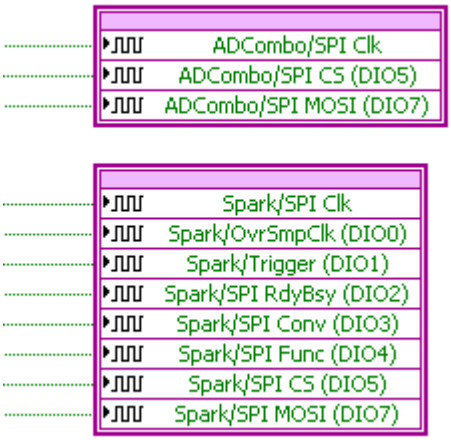


Figure 10a. Representative FPGA output nodes for ADCombo and Spark modules with correct output item selection.

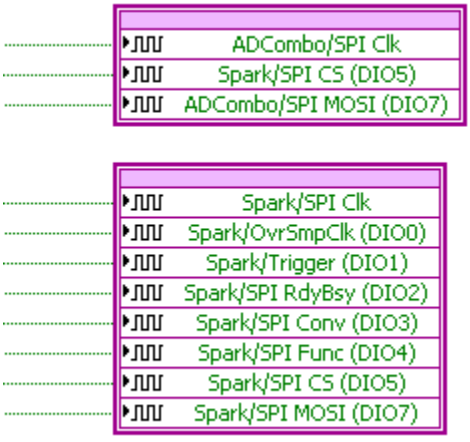


Figure 10b. Representative FPGA output nodes for ADCombo and Spark modules with incorrect output item selection within the ADCombo output node. The Spark (DIO5) output is selected in multiple nodes and therefore being driven by multiple sources. This will cause strange behavior or damage to the spark 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.

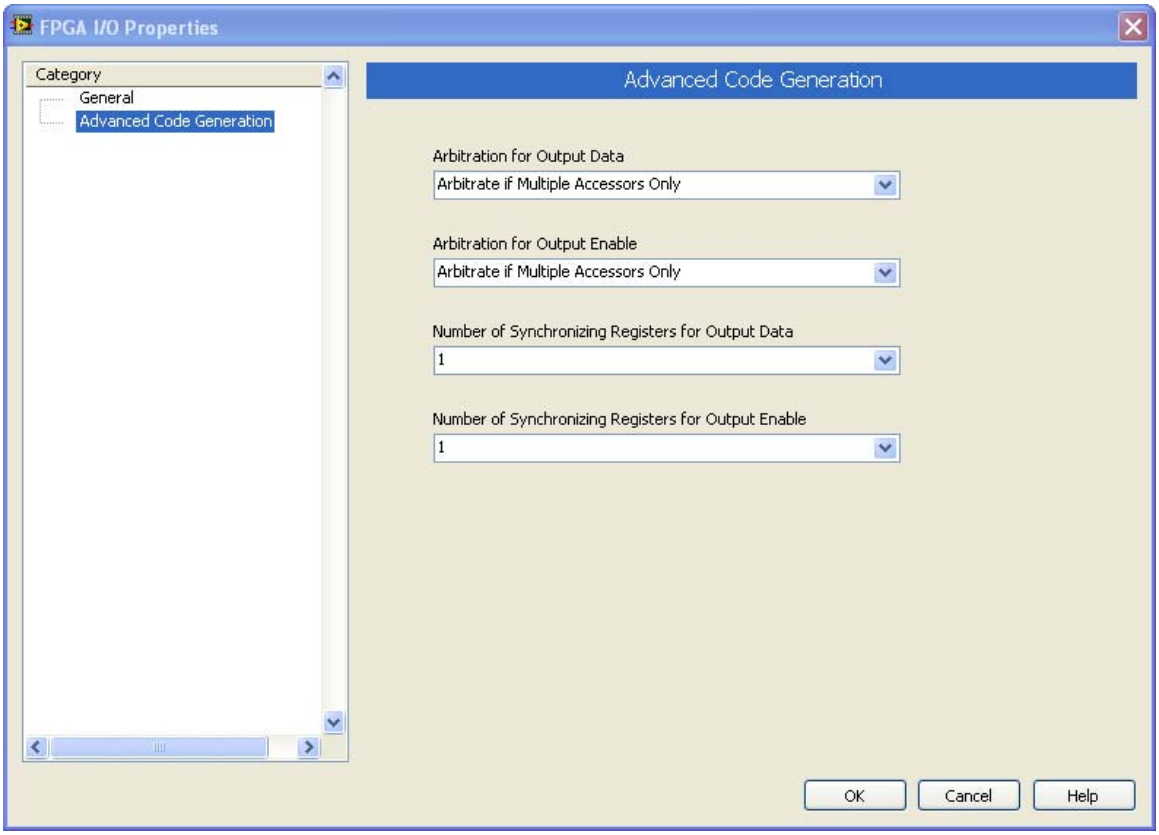
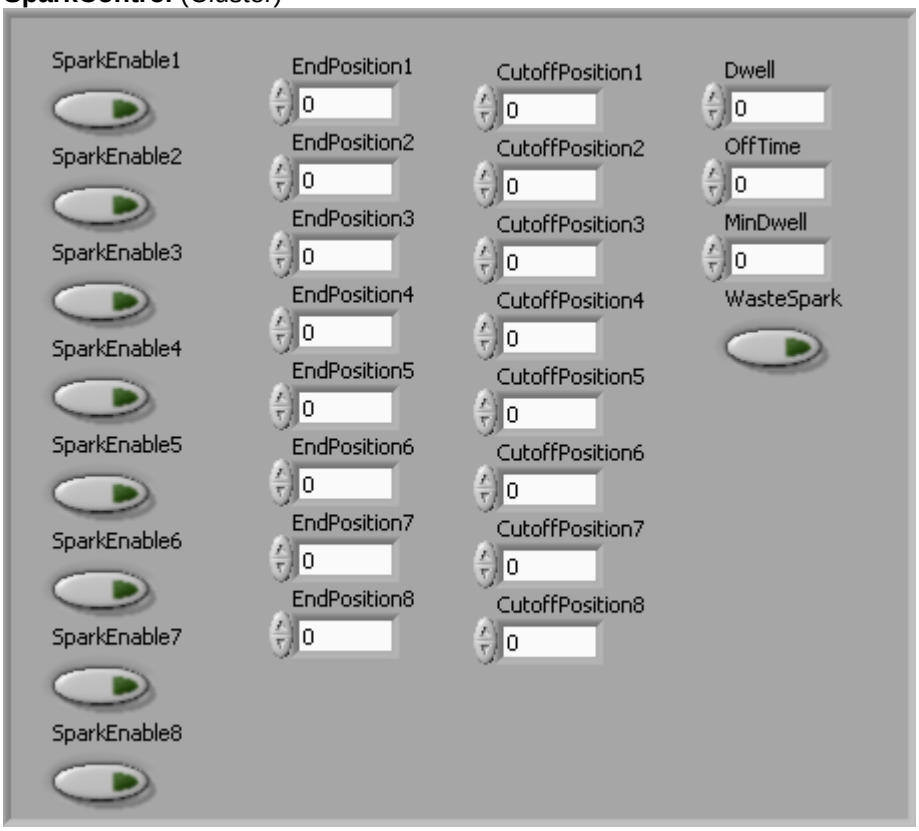


Figure 11. FPGA I/O Properties dialog configuration for cRIO modules.

SparkControl (Cluster)



SparkEnable (boolean): When TRUE, the spark command is enabled. When FALSE (default), the spark command is disabled.

EndPosition (uint16): Spark pulses are generated with a trailing edge coinciding with EndPosition. The length of the pulse will be according to Dwell. The leading edge will be determined by the current requested Dwell and the current engine speed. EndPosition will always take precedence over Dwell in the presence of engine speed fluctuations. If the engine speed increases after the spark pulse has started, then the actual dwell will be slightly shorter than the requested Dwell to ensure that correct spark timing is achieved. Likewise, if the engine speed decreases after the spark pulse has started, then the actual dwell be will slightly longer than the requested Dwell. However, MinDwell will take precedence over EndPosition to ensure that a spark occurs, even if it is late. Also, a maximum dwell of 5 msec is enforced to protect the driver circuit, even if the end of the spark pulse must occur before EndPosition. The units of EndPosition are CAT.

CutoffPosition (uint16): All spark pulse activity is "Cutoff" at CutoffPosition and reset for the next pulse. For multiplexed VIs, CutoffPositon is the position at which the next consecutive spark pulse parameters will be loaded for the given internal spark core. For multiplexed VIs, CutoffPosition should be set to a position close to, but after the most retarded spark timing position. This is an approximate rule and should be retarded out further if engine speed fluctuations are enough that pulses are being cut short. For non-multiplexed VIs, CutoffPosition MUST ALWAYS be at least 45 CAD after EndPosition. If the minimum spacing of 45 CAD is not maintained for non-multiplexed VIs, then spark commands will be generated incorrectly. The units of CutoffPosition are CAT.

Drivven provides a utility VI which can be implemented at the LabVIEW RT level for performing

the conversion of spark timing and cutoff timing values from TDC offsets to absolute CAT. The VI named Offset2CAT.vi can be used to convert advance, with respect to an absolute TDC, to CAT. This VI icon is shown in Figure 12.

Figure 12. Offset to CAT conversion VI.

Dwell (uint32): Determines the length of the spark command delivered to the driver circuit. Dwell is entered in terms of 40 MHz clock ticks and is internally limited to 200,000 ticks (5 msec). The value is also internally limited to 18 bits. Values larger than 18 bits will roll over from zero.

OffTime (uint16): Determines the length of inactivity following each spark pulse. It is recommended to set this value to at least 100 usec. OffTime provides a minimum amount of time for the command to be off to allow the actual spark to occur before turning the coil back on. OffTime is entered in terms of 40 MHz clock ticks and is internally limited to 16 bits. Values larger than 16 bits will roll over from zero.

MinDwell (uint32): Determines the minimum length of any spark pulse. It is possible that dwell could be cut short of the requested Dwell due to engine speed fluctuations or modifications to EndPosition. If MinDwell is not satisfied upon reaching EndPosition, then the pulse will be extended until MinDwell. This will ensure that a spark will always occur even if the timing is late. MinDwell should be set to a minimum value of dwell that will still guarantee a spark. MinDwell is entered in terms of 40 MHz clock ticks and is internally limited to 200,000 ticks (5 msec). The value is also internally limited to 18 bits. Values larger than 18 bits will roll over from zero. The following equation applies for converting dwell times in milliseconds to 40 MHz clock ticks.

Dwell(uint32 ticks) = Dwell(msec) * 40,000.

Drivven provides a utility VI which can be implemented at the LabVIEW RT level for performing this calculation. The VI named time2ticks.vi can be used to convert Time in milliseconds to uint32 ticks. This VI icon is shown in Figure 13.

Figure 13. Time to Ticks conversion VI.

WasteSpark (boolean): The WasteSpark input allows users to take advantage of dual output coils that operate on a pair of 4-stroke cylinders that are 360 degrees out of phase. While one cylinder is at top dead center compression, the pairing cylinder is at top dead center exhaust. A dual output coil will spark in both cylinders at the same time, sparking into a combustible mixture in one cylinder, while sparking into inert exhaust gases in the other. When WasteSpark is TRUE, the number of available spark channels will be divided by 2. For example, in the case of using an eight-channel spark VI, only the first four driver channels would be used instead of all eight. Pulse 1 will be paired with pulse 5 on channel 1, pulse 2 paired with pulse 6 on channel 2, pulse 3 paired with pulse 7 on channel 3, and pulse 4 paired with pulse 8 on channel 4. If using the waste spark mode, and the dwell time per channel is greater than 25% duty, then WasteSpark should be set to false and two spark channels should be used to drive the same coil primary.

FuelSparkSupervisor (Cluster)
This cluster input must be wired directly from the FuelSparkSupervisor cluster output of a Drivven EPT VI.

Position Conversion and Notes:

StartPosition, EndPosition and CutoffPosition are not entered as crank angle degrees (CAD), but crank angle ticks (CAT) which can be calculated from a CAD value. Each CAT represents a fractional CAD. A conversion factor referred to as Crank Angle Conversion (CAC) is required to convert between CAD and CAT. The units of CAC are degrees per tick. The following equations apply:

CAD (degrees) = CAT (ticks) * CAC (degrees per tick)
CAT (ticks) = CAD (degrees) / CAC (degrees per tick)

If the engine is a two-stroke then CAC is calculated as:

CAC (degrees per tick) = 360 (degrees) / MAX_CAT (ticks)

If the engine is a four-stroke then CAC is calculated as:

CAC (degrees per tick) = 720 (degrees) / MAX_CAT (ticks)

StartPosition, EndPosition and CutoffPosition are with respect to the absolute zero (0) CAT position which corresponds to the location of tooth 0 on the crank trigger wheel. Tooth 0 will be documented for each type of pattern within the EPT documentation.

When interfacing to the spark FPGA VIs from the CPU, the programmer can use the Offset2CAT VI for converting advance offset values, with respect to TDC, directly to CAT values. The Offset2CAT VI is shown in figure 12. This VI requires knowledge of the EPT Stroke parameter and MAX_CAT values, as well as the TDC of the particular cylinder being targeted.

Spark Command Scheduling:
The Spark VIs provide features that ensure the best possible spark command delivery, even while the CPU makes modifications to EndPosition and Dwell asynchronously to engine position.

Modifications to Dwell (Dwell, MinDwell, OffTime):

1. Dwell parameters can be modified at any time.
2. If Dwell is modified during the main spark pulse to a value less than the previous value of Dwell, then the pulse is continued until EndPosition.
3. If Dwell is modified to a value less than MinDwell (0, for example), then the pulse will still be started according to the value of Dwell and engine speed, but MinDwell will take precedence by sacrificing the requested EndPosition to ensure a spark occurs. This scenario should be avoided. Spark pulses should be disabled with the SparkEnable booleans.
4. If Dwell is modified during the main spark pulse to a value greater than the existing dwell, then the pulse is continued only until EndPosition, unless CutoffPosition or MAX_DWELL of 5 msec is encountered first.

Modifications to EndPosition:
EndPosition can be modified at any time. However, the value must not be advanced by more than 45 CAD within a single engine cycle. This value is referred to as the History Window. The spark VIs continually check the requested EndPosition with respect to the current crank position. If the EndPosition is modified by the CPU to a position in the past, the spark VIs use the History Window to determine whether a late spark pulse should be started.

1. For example, let's assume that a spark pulse is scheduled for an EndPosition of 200 Absolute CAD (ACAD) and a dwell time of 40,000 ticks (1.0 msec), such that the spark pulse is scheduled to turn on at 150 ACAD. Let's also assume that the CurrentPosition of the EPT VI is 140 ACAD when the CPU modifies EndPosition to 180 ACAD, which means that the required start of the pulse is now 130 ACAD. This new start is in the recent past by 10 CAD. Since this is less than the 45 CAD History Window, then the spark VI will immediately start the spark pulse even though it is late. The dwell will be turned off at EndPosition as long as MinDwell is satisfied.
2. As another example, let's assume that a spark pulse is scheduled for an EndPosition of 200 ACAD and a dwell time of 40,000 clock ticks (1.0 msec) such that the spark pulse is scheduled to turn on at 150 ACAD. Let's also assume that the CurrentPosition of the EPT VI is 140 ACAD when the CPU modifies EndPosition to 120 ACAD, which means that the required start of the pulse is now 70 ACAD. This new start is in the recent past by 70 CAD. Since this is greater than the 45 CAD History Window, the spark VI will not generate a late pulse, effectively skipping a cycle without a spark pulse. The following cycle will have a pulse delivered starting at 120 ACAD.

CutoffPosition must be set so that the following conditions are satisfied:

1. In the case of non-multiplexed spark VIs, CutoffPosition must be set at least 45 CAD after EndPosition. If this minimum spacing is not maintained, then spark commands will be generated incorrectly.

Examples

The following screen capture in Figure 14 shows a LabVIEW FPGA block diagram demonstrating the interface between EPT VIs and fuel/spark VIs. No external signals are wired to the EPTInSig cluster, thus requiring simulation of these signals. However, those signals may be provided from the AD Combo or VR/Hall Module Kits. This FPGA application is entirely contained within a single cycle loop, clocked at the required 40 MHz. Notice that the EPT VI provides the FuelSparkSupervisor cluster to be wired directly to the compatible fuel and spark VIs. Refer to the LabVIEW FPGA documentation for details about configuring cRIO I/O pins. A similar example VI is provided for any EPT, Fuel or Spark product ordered from Drivven.

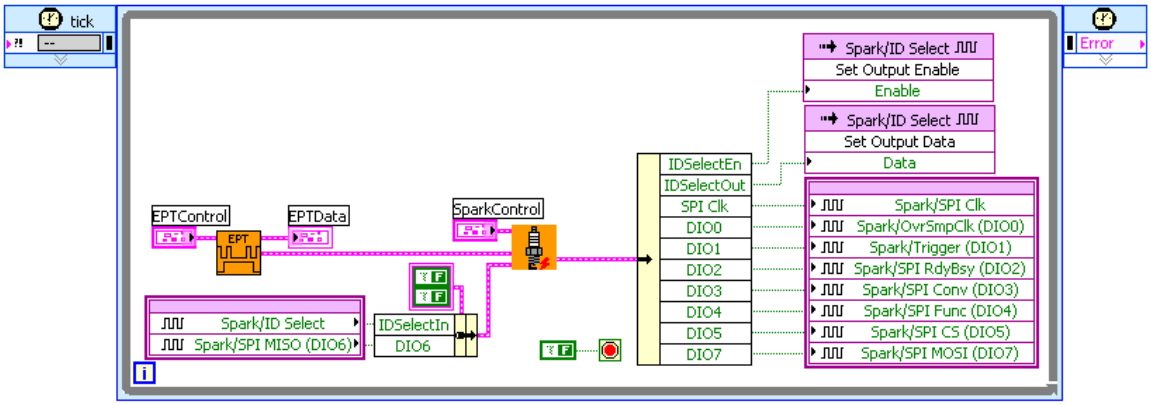


Figure 14. LabVIEW FPGA Block diagram example.