PED-Board

just add power

Power Electronics & Drives Board

for National Instruments System on Module - sbRIO-9651®

LabVIEW® Drivers and Demo Programs

Table of contents

I. Limited Warranty

This document is provided ‘as is’ and is subject to being changed, without notice, in future editions. For the latest version, refer to ped-board.com.

PED-Board team reviews this document carefully for technical accuracy; however, PED-Board team MAKES NO EXPRESS OR IMPLIED WARRANTIES AS TO THE ACCURACY OF THE INFORMATION CONTAINED HEREIN AND SHALL NOT BE LIABLE FOR ANY ERRORS.

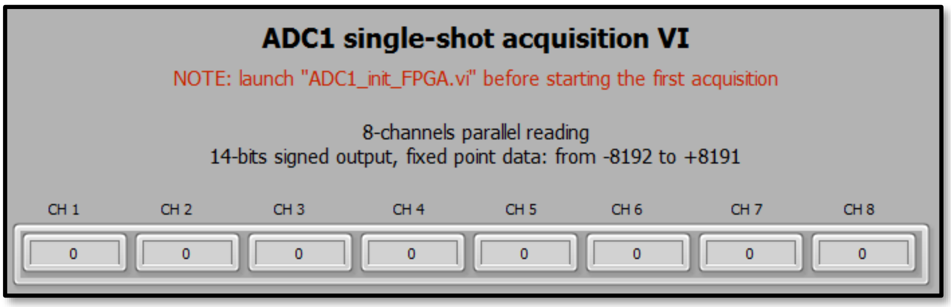
Trademarks and service marks of third parties are the property of their respective owners.

II. LabVIEW Drivers

PED-Board is provided with dedicated LabVIEW drivers to speed up the development process of specific applications. Each driver can be used to configure and run the related peripheral. Demo programs are also provided to demonstrate PED-Board capabilities.

A. ADC1 driver

ADC1 driver package consists in two VIs named **ADC1_init_FPGA.vi** and **ADC1_read_FPGA.vi**. File **ADC1_init_FPGA.vi** is necessary to initialize the required pins and must be run only once before the first acquisition.
ADC1_read_FPGA.vi performs a single acquisition and providing the output of each channel (after the simultaneous sampling of the 8 channels). Data output is represented by a 14-bits signed fixed-point number for each channel.
In order to acquire signals from ADC1 it is therefore necessary to launch **ADC1_init_FPGA.vi** once and then **ADC1_read_FPGA.vi** can be run whenever is needed without delays between the executions.
Figure 1 shows the Front Panel of **ADC1_read_FPGA.vi**. It is important to note that this is not a standalone VI and that it must be only used as a sub-VI after the execution of the initialization task. The following section shows a basic example about how to use these drivers.



ADC1 single-shot acquisition VI

NOTE: launch "ADC1_init_FPGA.vi" before starting the first acquisition

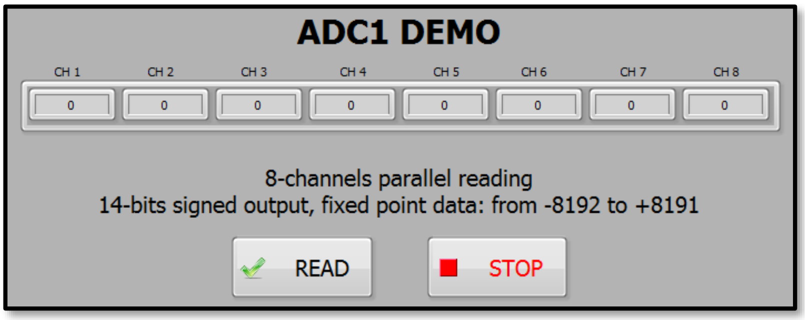
8-channels parallel reading
14-bits signed output, fixed point data: from -8192 to +8191

CH 1	CH 2	CH 3	CH 4	CH 5	CH 6	CH 7	CH 8
0	0	0	0	0	0	0	0

Figure 1. Font panel of the sub-VI ADC1_read_FPGA.vi.

- *ADC1 demo*

A demo VI named **DEMO_FPGA_ADC1.vi** is provided to show the correct use of the driver package. Provided simple VI performs the initialization task running once **ADC1_init_FPGA.vi**. After that, it executes **ADC1_read_FPGA.vi** each time the user presses the READ button, showing the acquired data on the front panel. The execution ends pressing the STOP button. Figure 2 and Figure 3 show, respectively, the front panel and the block diagram.



ADC1 DEMO

CH 1	CH 2	CH 3	CH 4	CH 5	CH 6	CH 7	CH 8
0	0	0	0	0	0	0	0

8-channels parallel reading
14-bits signed output, fixed point data: from -8192 to +8191

Figure 2. DEMO_FPGA_ADC1.vi front panel.

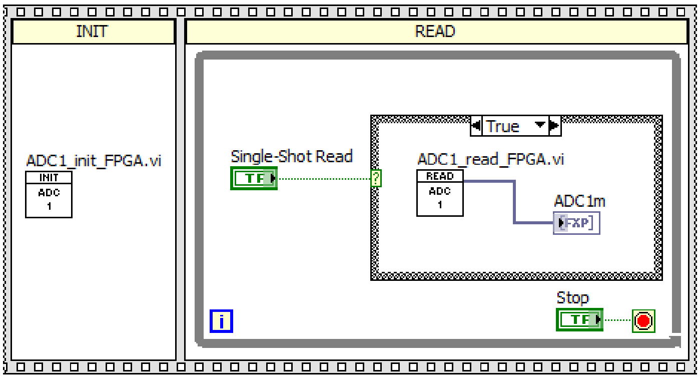


Figure 3. DEMO_FPGA_ADC1.vi block diagram.

B. ADC2 driver

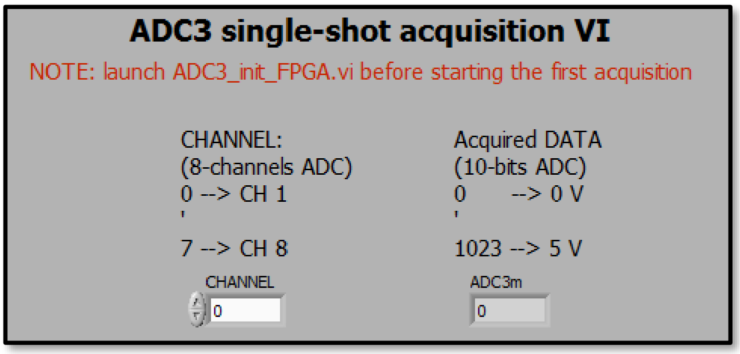
ADC2 interface is similar to the previously depicted ADC1, except for the low-pass filtering path that is composed by a simple RC filter with impedance matching circuit.

- *ADC2 demo*

Driver package and demo VI provided for ADC2 are identical to the ones given for ADC1. Please refer to the previous sections for detailed description.

C. ADC3 driver

ADC3 driver package consists in two VIs named **ADC3_init_FPGA.vi** and **ADC3_read_FPGA.vi**.
ADC3_init_FPGA.vi is necessary to initialize the required pins and must be run only once before the first acquisition.
ADC3_read_FPGA.vi performs a single acquisition and provides the output for the specified channel. Channel input must be a 3-bits fixed-point number where 0 corresponds to Channel-1 and 7 to Channel-8. Data output is a 10-bits number but it is represented by a 16-bits unsigned integer.
In order to acquire signals from ADC3 it is therefore necessary to launch **ADC3_init_FPGA.vi** once and then **ADC3_read_FPGA.vi** can be run whenever is needed without delays between the executions.
Figure 4 shows the front panel of **ADC3_read_FPGA.vi**. It is important to note that this is not a standalone VI and that it must be used only as a sub-VI after the execution of the initialization task.
The following section shows a basic example about how to use these drivers.



ADC3 single-shot acquisition VI

NOTE: launch ADC3_init_FPGA.vi before starting the first acquisition

CHANNEL:
(8-channels ADC)
0 --> CH 1
'
7 --> CH 8

Acquired DATA
(10-bits ADC)
0 --> 0 V
'
1023 --> 5 V

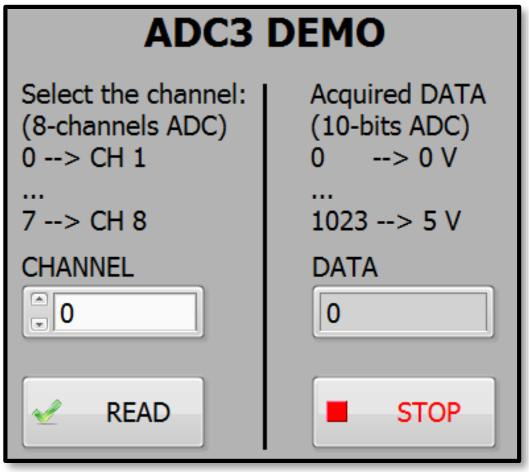
CHANNEL: 0

ADC3m: 0

Figure 4. ADC3_read_FPGA.vi front panel.

- *ADC3 demo*

A demo VI named **DEMO_FPGA_ADC3.vi** is provided to show the correct use of the driver package.
This simple VI performs the initialization task running once **ADC3_init_FPGA.vi** and then it executes **ADC3_read_FPGA.vi** each time the user presses READ button”. Acquired data from the specified channel are shown on the front panel. The execution ends pressing STOP button. Figure 5 and Figure 6 show, respectively, the front panel and the block diagram.



ADC3 DEMO

Select the channel:
(8-channels ADC)
0 --> CH 1
...
7 --> CH 8

Acquired DATA
(10-bits ADC)
0 --> 0 V
...
1023 --> 5 V

CHANNEL: 0

DATA: 0

Figure 5. DEMO_FPGA_ADC3.vi front panel.

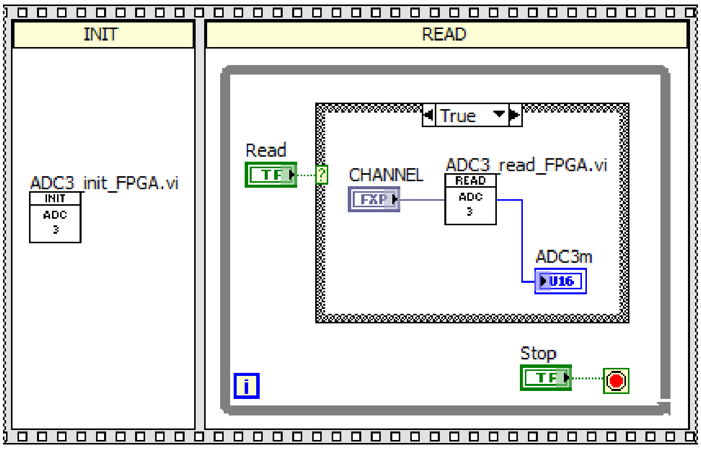
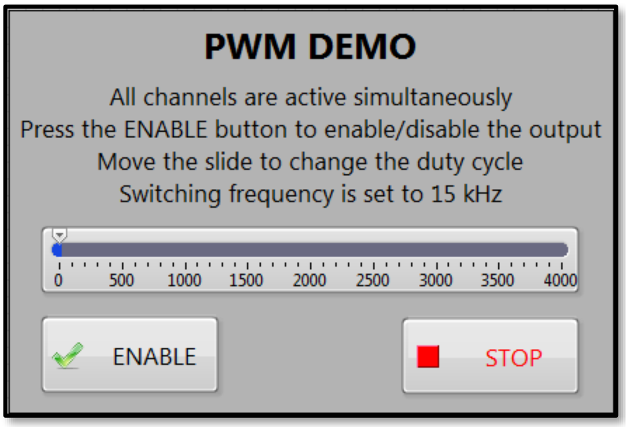


Figure 6. DEMO_FPGA_ADC3.vi block diagram.

D. PWM output

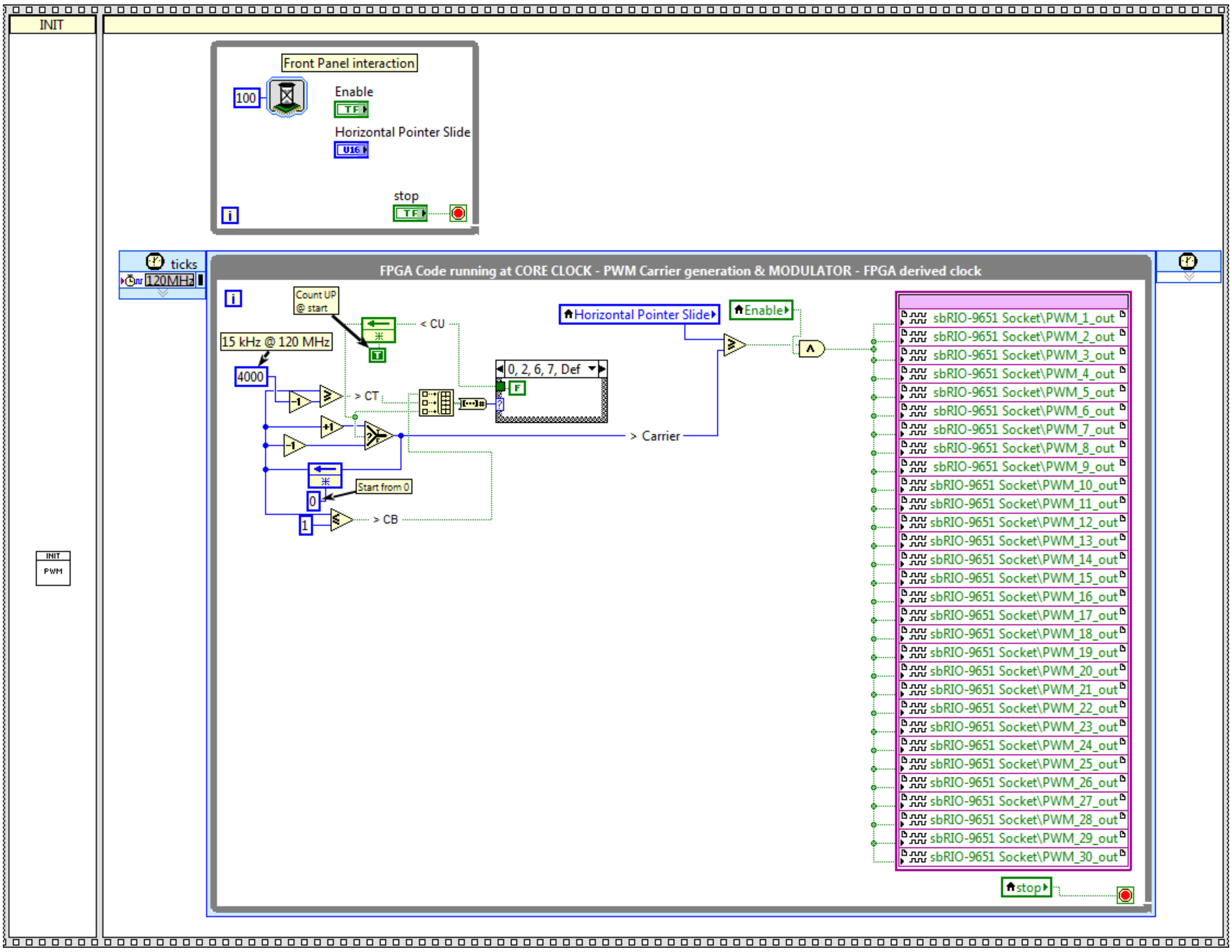
A demo program is provided to test PWM capabilities. Reference VI is **DEMO_FPGA_PWM.vi**.



PWM DEMO

All channels are active simultaneously
Press the ENABLE button to enable/disable the output
Move the slide to change the duty cycle
Switching frequency is set to 15 kHz

Figure 7. PWM DEMO front panel.



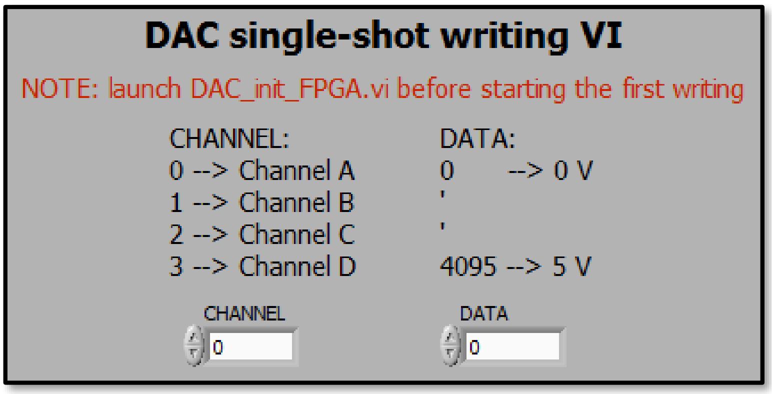
INIT
Front Panel interaction
100
Enable
Horizontal Pointer Slide
U16
stop
ticks
120MHz
FPGA Code running at CORE CLOCK - PWM Carrier generation & MODULATOR - FPGA derived clock
Count UP @ start
< CU
15 kHz @ 120 MHz
4000
> CT
Horizontal Pointer Slide
Enable
0, 2, 6, 7, Def
> Carrier
Start from 0
0
1
> CB
sbRIO-9651 Socket\PWM_1_out
sbRIO-9651 Socket\PWM_2_out
sbRIO-9651 Socket\PWM_3_out
sbRIO-9651 Socket\PWM_4_out
sbRIO-9651 Socket\PWM_5_out
sbRIO-9651 Socket\PWM_6_out
sbRIO-9651 Socket\PWM_7_out
sbRIO-9651 Socket\PWM_8_out
sbRIO-9651 Socket\PWM_9_out
sbRIO-9651 Socket\PWM_10_out
sbRIO-9651 Socket\PWM_11_out
sbRIO-9651 Socket\PWM_12_out
sbRIO-9651 Socket\PWM_13_out
sbRIO-9651 Socket\PWM_14_out
sbRIO-9651 Socket\PWM_15_out
sbRIO-9651 Socket\PWM_16_out
sbRIO-9651 Socket\PWM_17_out
sbRIO-9651 Socket\PWM_18_out
sbRIO-9651 Socket\PWM_19_out
sbRIO-9651 Socket\PWM_20_out
sbRIO-9651 Socket\PWM_21_out
sbRIO-9651 Socket\PWM_22_out
sbRIO-9651 Socket\PWM_23_out
sbRIO-9651 Socket\PWM_24_out
sbRIO-9651 Socket\PWM_25_out
sbRIO-9651 Socket\PWM_26_out
sbRIO-9651 Socket\PWM_27_out
sbRIO-9651 Socket\PWM_28_out
sbRIO-9651 Socket\PWM_29_out
sbRIO-9651 Socket\PWM_30_out
stop
INIT
PWM

Figure 8. PWM DEMO block diagram.

E. DAC driver

DAC driver package consists in two VIs named **DAC_init_FPGA.vi** and **DAC_write_FPGA.vi**.
DAC_init_FPGA.vi is necessary to initialize the required pins and must be run only once before the first execution of **DAC_write_FPGA.vi**.

DAC_write_FPGA.vi performs a single writing and provides the output on the specified channel. Channel input must be a 2-bits fixed-point number where 0 corresponds to Channel-A and 3 to Channel-D. Data input must be a 12-bits fixed-point number.
In order to use the DAC, it is therefore necessary to launch **DAC_init_FPGA.vi** once and then **DAC_write_FPGA.vi** can be run whenever is needed without delays between the executions.
Figure 9 shows the Front Panel of **DAC_write_FPGA.vi**. It is important to note that this is not a standalone VI and that it must be used only as a sub-VI after the execution of the initialization task.
The following section shows a basic example about how to use these drivers.



DAC single-shot writing VI

NOTE: launch DAC_init_FPGA.vi before starting the first writing

CHANNEL:
0 --> Channel A
1 --> Channel B
2 --> Channel C
3 --> Channel D

DATA:
0 --> 0 V
'
'
4095 --> 5 V

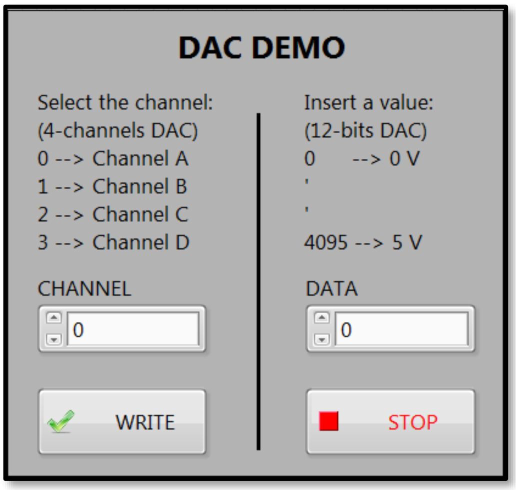
CHANNEL: 0

DATA: 0

Figure 9. DAC_write_FPGA.vi front panel.

- *DAC demo*

A demo VI named **DEMO_FPGA_DAC.vi** is provided to show the correct use of the driver package.
This simple VI performs the initialization task running once **DAC_init_FPGA.vi** and then it executes **DAC_write_FPGA.vi** each time the user presses WRITE button. Channel and Data output can be modified using the controls on the front panel. The execution ends pressing STOP button.
Figure 10 and Figure 11 show, respectively, the block diagram and the front panel.



DAC DEMO

Select the channel:
(4-channels DAC)
0 --> Channel A
1 --> Channel B
2 --> Channel C
3 --> Channel D

Insert a value:
(12-bits DAC)
0 --> 0 V
'
'
4095 --> 5 V

CHANNEL: 0

DATA: 0

Figure 10. DEMO_FPGA_DAC.vi block diagram.

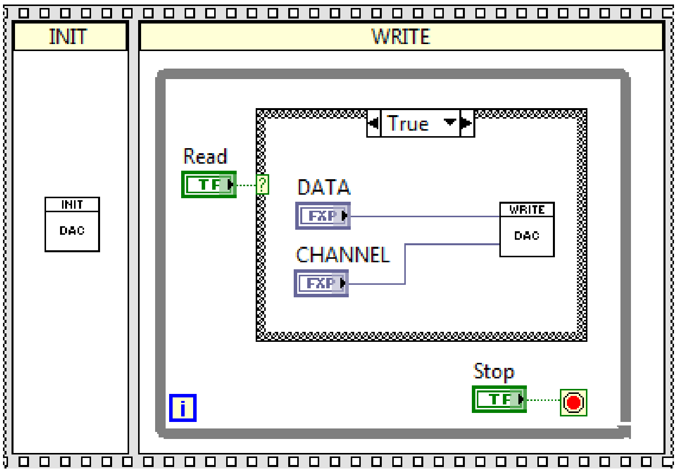
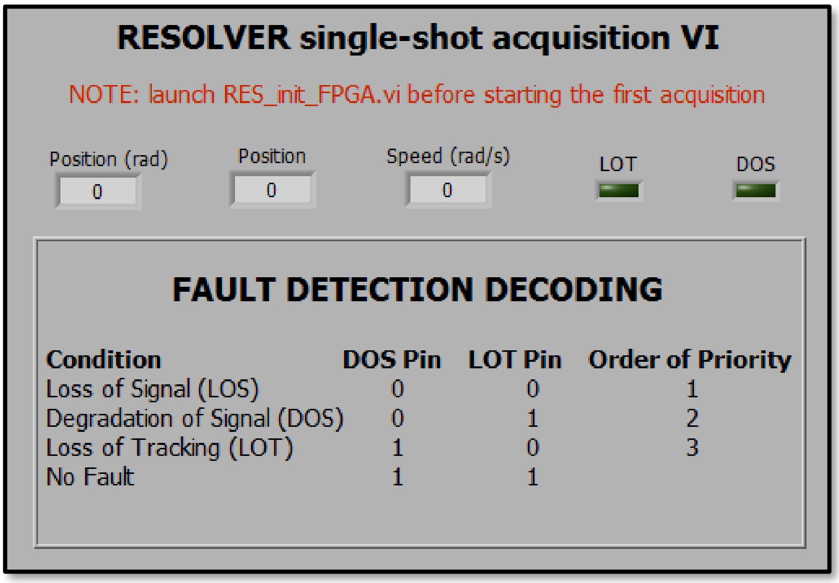


Figure 11. DEMO_FPGA_DAC.vi front panel.

F. *RESOLVER driver*

RES driver package consists in two VIs named **RES_init_FPGA.vi** and **RES_read_FPGA.vi**.
RES_init_FPGA.vi is necessary to initialize the required pins and must be run only once before the first acquisition.
RES_read_FPGA.vi performs a single acquisition of Speed (rad/s) and Position (rad) and provides error signals (LOT and DOS). Position data output is represented by a <10,3> unsigned fixed-point number. Speed data output is represented by a <14,11> signed fixed-point number. Position output is also provided as 12 bits FXP data <12,12> unsigned as the discretized value achieved from the resolver-to-digital converter. When the angle is equal to ZERO, Position is ZERO; whereas, for an angle of 360°, Position is 4095 (2^{12}-1).
LOT and DOS error signals are represented by two Booleans where the no-fault state is represented by the logical 1. Please refer to the Analog Devices AD2S1205YST data-sheet for detailed information about fault detection.

In order to acquire signals from the resolver it is therefore necessary to launch **RES_init_FPGA.vi** once and then **RES_read_FPGA.vi** can be run whenever is needed without delays between the executions.
Figure 12 shows the front panel of **RES_read_FPGA.vi**. It is important to note that this is not a standalone VI and that it must be used only as a sub-VI after the execution of the initialization task. The following section shows a basic example on how to use these drivers.



RESOLVER single-shot acquisition VI

NOTE: launch RES_init_FPGA.vi before starting the first acquisition

Position (rad)	Position	Speed (rad/s)	LOT	DOS
0	0	0		

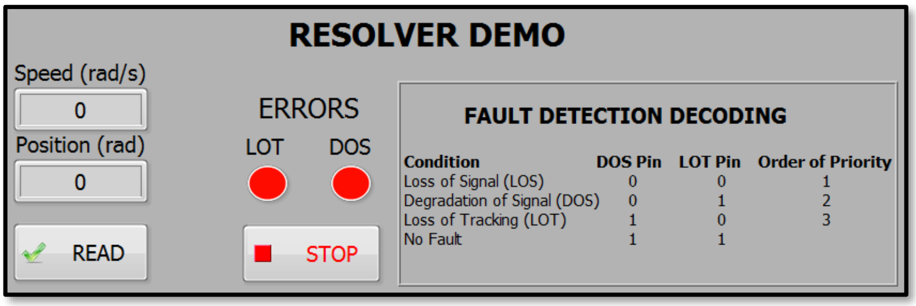
FAULT DETECTION DECODING

Condition	DOS Pin	LOT Pin	Order of Priority
Loss of Signal (LOS)	0	0	1
Degradation of Signal (DOS)	0	1	2
Loss of Tracking (LOT)	1	0	3
No Fault	1	1	

Figure 12. RES_read_FPGA.vi front panel.

- *RESOLVER demo*

A demo VI named **DEMO_FPGA_RES.vi** is provided to show the correct use of the driver package.
This simple VI performs the initialization task running once **RES_init_FPGA.vi** and then it executes **RES_read_FPGA.vi** each time the user presses READ button. Acquired data are shown on the front panel. The execution ends pressing the STOP button. Figure 13 and Figure 14 show, respectively, the front panel and the block diagram.



RESOLVER DEMO

Speed (rad/s): 0

Position (rad): 0

ERRORS
LOT ● DOS ●

READ

STOP

FAULT DETECTION DECODING

Condition	DOS Pin	LOT Pin	Order of Priority
Loss of Signal (LOS)	0	0	1
Degradation of Signal (DOS)	0	1	2
Loss of Tracking (LOT)	1	0	3
No Fault	1	1	

Figure 13. DEMO_FPGA_RES.vi front panel.

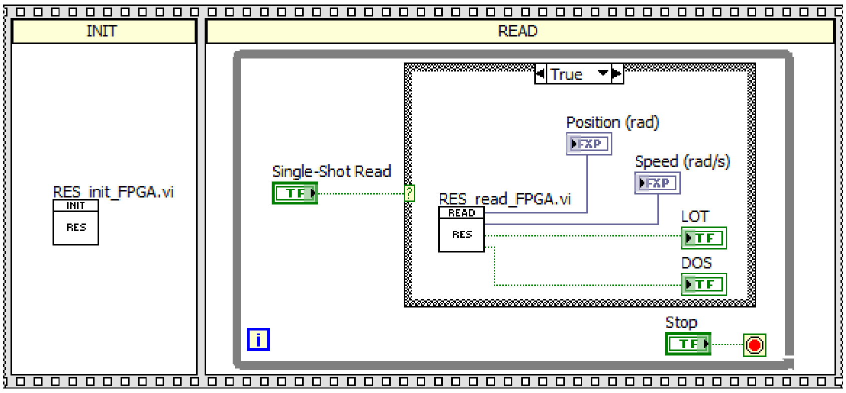


Figure 14. DEMO_FPGA_RES.vi block diagram.

G. FPGA main scheduler

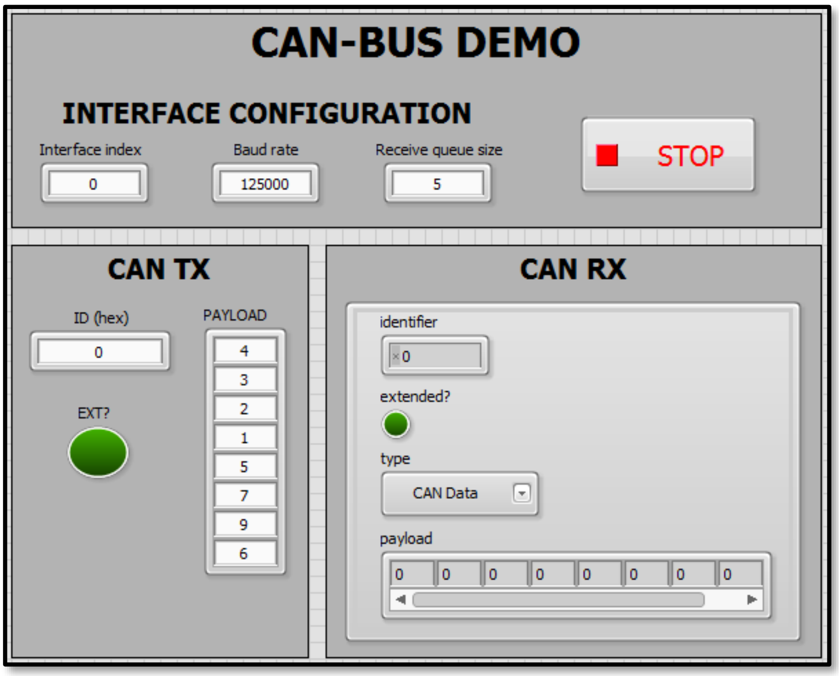
FPGA_MAIN_VI.vi contains a basic scheduler, a PWM modulator, initialization and the main task that is called when the PWM period starts. It can be used as starting VI for specific application.

H. Real-Time main VI

RT_MAIN.vi is linked to **FPGA_MAIN_VI.vi** and can be used to develop the Real-Time application.

- *CAN-Bus demo (RT_CAN_MAIN.VI)*

Using the generated CLIP, CAN-bus is directly controller by the Real-Time (RT) target, without any need to program the FPGA. Front panel of the provided CAN-bus demo in shown in Figure 15. In the RT target, a black FPGA VI can be loaded and run as shown in Figure 16.


CAN-BUS DEMO

INTERFACE CONFIGURATION

Interface index: 0
Baud rate: 125000
Receive queue size: 5

STOP

CAN TX

ID (hex): 0
PAYLOAD: 4, 3, 2, 1, 5, 7, 9, 6
EXT? ●

CAN RX

identifier: x0
extended? ●
type: CAN Data
payload: 0 0 0 0 0 0 0 0

Figure 15. CAN-bus Real-Time front panel.

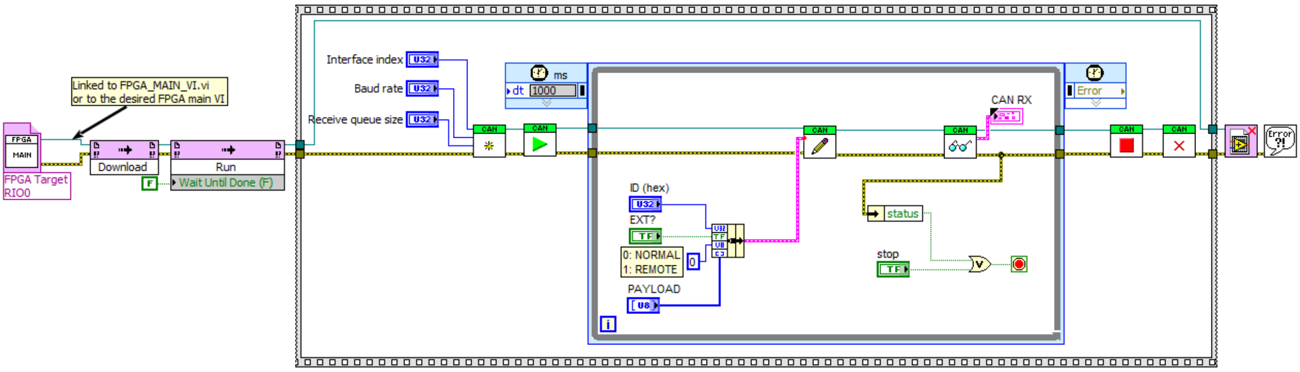


Figure 16. CAN-bus Real-Time block diagram.

- *RS-485 demo (RT_RS485_MAIN.VI)*

PED-Board is equipped with an isolated RS-485 transceiver. Sample project provides a Real-Time VI named **RT_RS845_MAIN.VI**, which can be used to test the port capabilities. Interface 5 should be selected for the on-board RS-485 port generated from the LabVIEW CLIP.