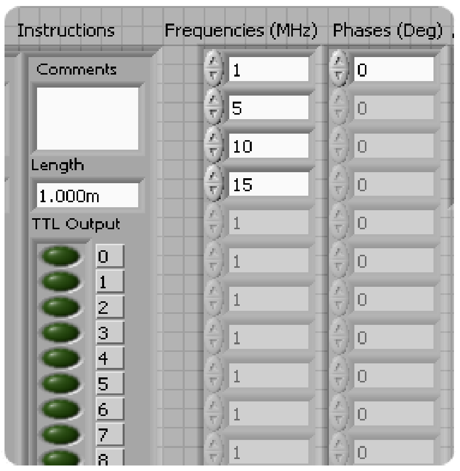
SpinCore PulseBlasterDDS LabVIEW Extensions

User's Manual



Frequencies (MHz)	Phases (Deg)
1	0
5	0
10	0
15	0
1	0
1	0
1	0
1	0
1	0
1	0
1	0

SpinCore Technologies, Inc.
http://www.spincore.com

Congratulations and *thank you* for choosing a design from SpinCore Technologies, Inc.

We appreciate your business!

At SpinCore we try to fully support the needs of our customers. If you are in need of assistance, please contact us and we will strive to provide the necessary support.

Table of Contents

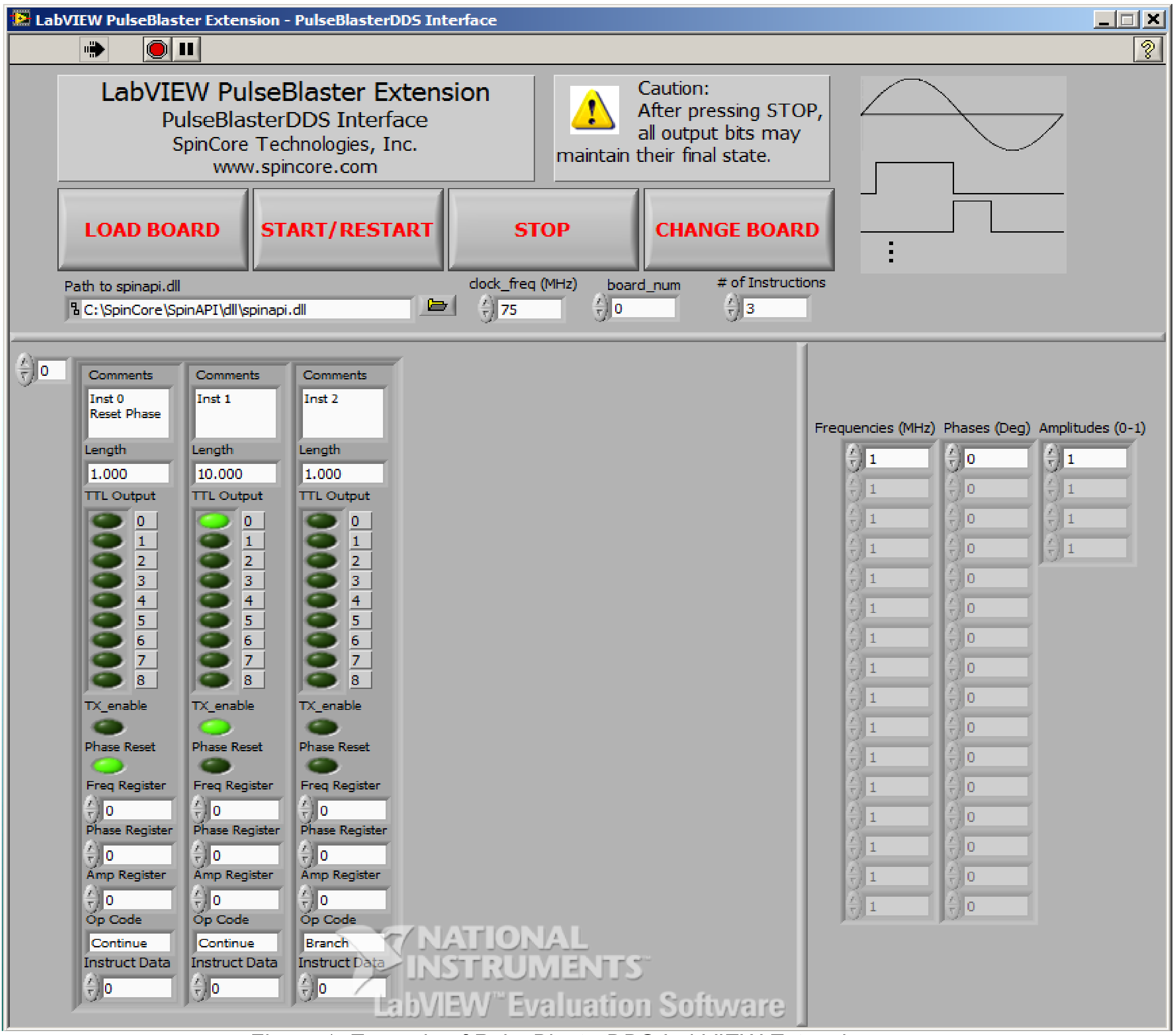
I. Overview

The SpinCore PulseBlasterDDS LabVIEW Extensions (PBLV-DDS) provide the functionality of programming and controlling digital pulse and RF generation in PulseBlasterDDS and RadioProcessor boards using the simple NI LabVIEW graphical programming interface. The package contains basic subVIs that can be used to include PulseBlasterDDS interaction from your own LabVIEW programs, as well as some complete example VIs. Additionally, all of the examples are available as stand-alone applications to control.

The PBLV-DDS is an intuitive graphical equivalent of the SpinAPI C functions. The GUI (known as the front panel) has all the inputs needed to access the PulseBlaster including instruction registers, clock frequency, and buttons for loading, starting, and stopping the board. The input is then used in the back-end code (known as the block diagram) to access the C functions that control and program the PulseBlasterDDS. The LabVIEW block diagram is a one-to-one equivalent of the corresponding C code, without having to write code. An example of the front panel and block diagram are shown in Figure 1 and Figure 2.

Note: For information on using the the Digital Pulse Generation functionality of the board in LabVIEW, please see the PulseBlaster LabVIEW Extensions documentation. (http://www.spincore.com/CD/PBLV/PBLV_Manual.pdf)

All example VIs and subVIs are described in detail below.



LabVIEW PulseBlaster Extension
PulseBlasterDDS Interface
SpinCore Technologies, Inc.
www.spincore.com

Caution:
After pressing STOP, all output bits may maintain their final state.

Path to spinapi.dll: C:\SpinCore\SpinAPI\dll\spinapi.dll
clock_freq (MHz): 75
board_num: 0
# of Instructions: 3

0

Comments	Inst 0 Reset Phase	Inst 1	Inst 2
Length	1.000	10.000	1.000
TTL Output 0	○	●	○
TTL Output 1	○	○	○
TTL Output 2	○	○	○
TTL Output 3	○	○	○
TTL Output 4	○	○	○
TTL Output 5	○	○	○
TTL Output 6	○	○	○
TTL Output 7	○	○	○
TTL Output 8	○	○	○
TX_enable	○	●	○
Phase Reset	●	○	○
Freq Register	0	0	0
Phase Register	0	0	0
Amp Register	0	0	0
Op Code	Continue	Continue	Branch
Instruct Data	0	0	0

Frequencies (MHz)	Phases (Deg)	Amplitudes (0-1)
1	0	1
1	0	1
1	0	1
1	0	1
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	

Figure 1: Example of PulseBlasterDDS LabVIEW Extensions User Interface

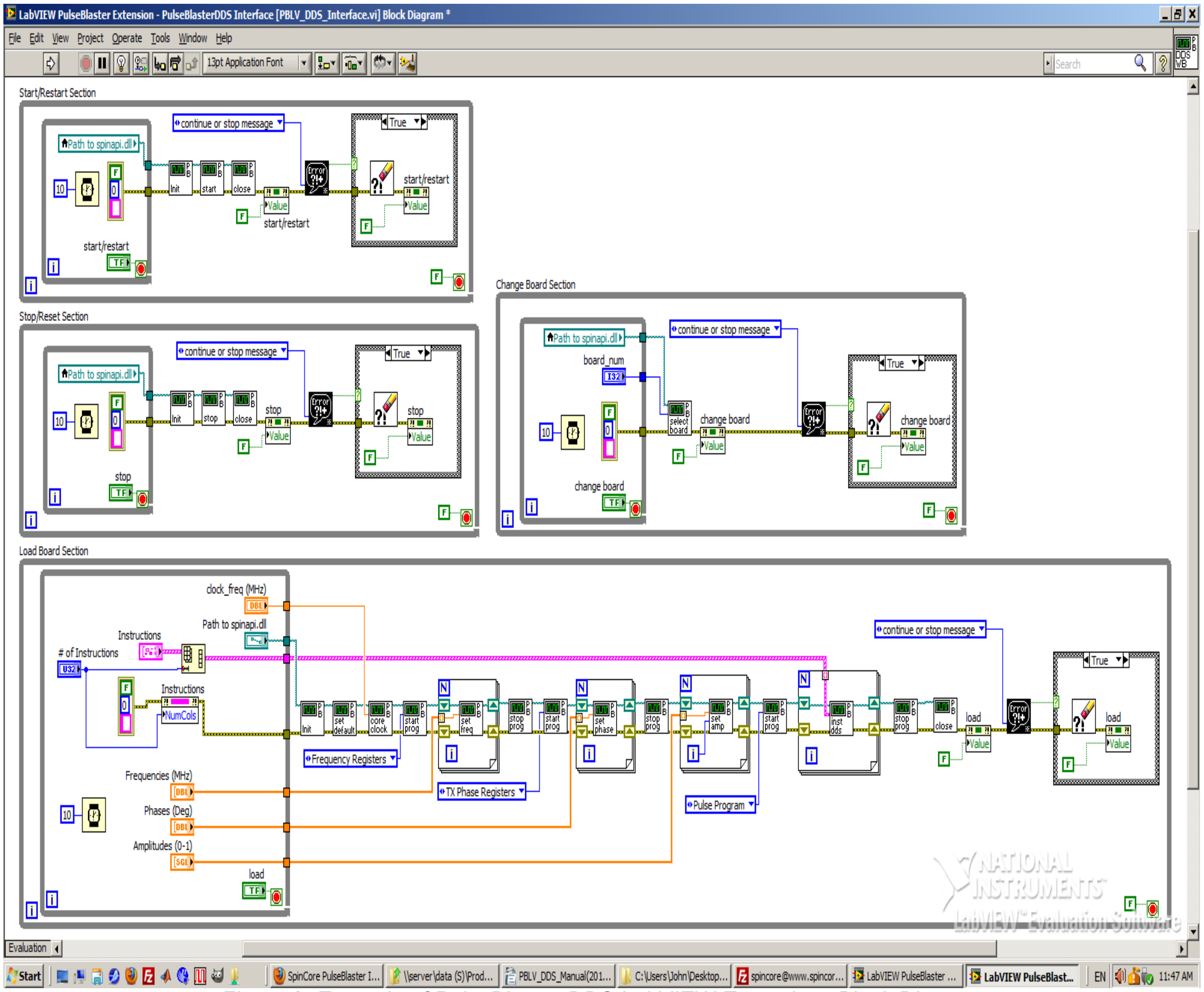


Figure 2: Example of PulseBlaster-DDS LabVIEW Extensions Block Diagram

II. Installation

There are two methods of using the PulseBlasterDDS LabVIEW Extensions. The first method is a set of stand-alone executables which will control the PulseBlasterDDS boards with a simple, intuitive interface with no other necessary knowledge of LabVIEW programming. The second method is a set of LabVIEW VIs which can be used with the LabVIEW Development platform to create custom programs using the PBLV interface.

Method 1: Stand-Alone Executables

In order for PBLV-DDS-II stand-alone executables to work, the following must be installed:

- SpinCore Driver Suite - Please see the SpinCore Driver Suite Installation Guide (http://www.spincore.com/support/spinapi/instructions/) for more information.
- National Instruments LabVIEW Run-Time Engine 2010 (http://www.spincore.com/support/PBLV/LVRTE2010std.exe) - Note if you have LabVIEW 2010 or later installed, this is not needed.
- LabVIEW PulseBlaster Extensions Stand-Alone executables located here.

Method 2: Customizable VIs

In order for PBLV-DDS-II customizable VIs to work, the following must be installed:

- SpinCore Driver Suite - Please see the SpinCore Driver Suite Installation Guide (http://www.spincore.com/support/spinapi/instructions/) for more information.
- National Instruments LabVIEW 8.6 or later - If you do not have LabVIEW 8.6 or later installed, you may download a 30-day evaluation (https://lumen.ni.com/nicif/us/lveval/content.xhtml) of the latest LabVIEW development software.
  (NOTE: Customizable VIs for versions of LabVIEW as old as LabVIEW 8.0 are available. For customizable VIs older than LabVIEW 8.6 please contact SpinCore Technologies via the web forum.)
- LabVIEW PulseBlaster Extensions located here.
  (NOTE: Customizable VIs use the C-calling convention. For customizable VIs that use the WINAPI calling convention please contact SpinCore Technologies via the web forum.)

III. General Information

Instruction Description

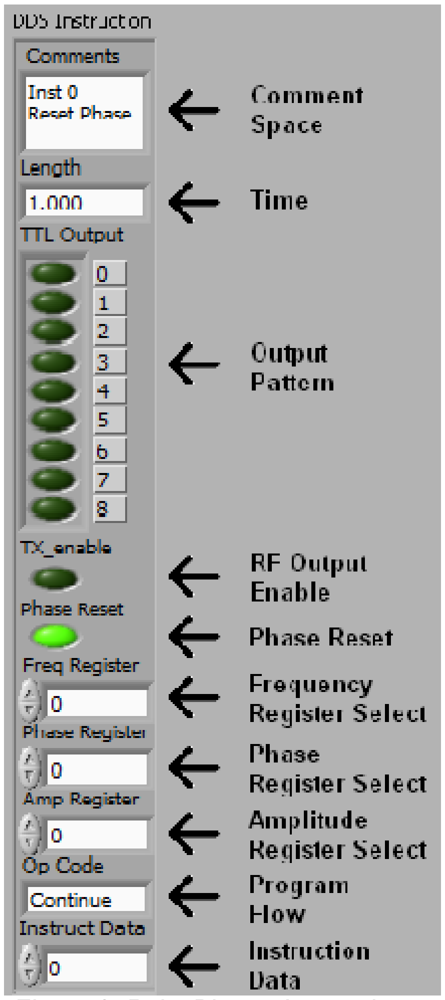


Figure 3: PulseBlaster Instruction

Throughout the PulseBlasterDDS LabVIEW Extension, the instruction word, corresponding to a given interval in the pulse sequence, is given as shown to the left. The instruction on the front panel (Figure 3) is broken up into 10 fields displayed from top to bottom:

- Comment space is an empty text box. This area is used to annotate your instruction and is not programmed onto the PulseBlasterDDS board.
- Time is the duration that the current instruction is to be held in microseconds. Range and resolution varies depending on what board is being used. This corresponds to “length” in SpinAPI.
- Output Pattern determines the state of each TTL output bit. If an LED is on then it's corresponding output bit is high, and if the LED is off then it's corresponding output bit is low. This corresponds to “flags” in SpinAPI.
- RF Output Enable tells the DDS board whether the RF output should be enabled during this instruction. This corresponds to “tx_enable” in SpinAPI.
- Phase Reset tells the DDS board to reset the phase of all DDS channels to their time=0 phase. They will stay in this state until the value of this bit returns to 0. This corresponds to “phase_reset” in SpinAPI.
- Frequency Register Select determines which frequency register will be used for the RF output during this instruction. Refer to your board manual for the number of frequency registers available. This corresponds to “freq” in SpinAPI.
- Phase Register Select determines which phase register will be used for the RF output during this instruction. Refer to your board manual for the number of phase registers available. This corresponds to “tx_phase” in SpinAPI.
- Amplitude Register Selects determines which amplitude register will be used for the RF output during this instruction. Refer to your board manual for the number of amplitude registers available. This corresponds to “amp” in SpinAPI.
- Program Flow determines the flow of program instructions *after* the bit pattern is displayed. This corresponds to “inst” in SpinAPI. Available instructions are:

  - CONTINUE - Program execution continues to next instruction.
  - STOP - Stop execution of program.
  - LOOP - Specify beginning of a loop. Execution continues to next instruction. Instruction data used to specify number of loops.
  - END_LOOP Execution returns to beginning of loop and decrements loop counter. Instruction data used to specify beginning of loop.
  - JSR - Program execution jumps to beginning of a subroutine. Instruction data used to specify address of first subroutine instruction.
  - RTS - Program execution returns to instruction after JSR was called.
  - BRANCH - Program execution continues at specified instruction. Instruction data specifies address of next instruction.
  - LONG_DELAY - For long interval instructions. Data field specifies a multiplier of the length field. Execution continues to next instruction.
  - WAIT - Program execution stops and waits for software or hardware trigger. Execution continues to next instruction after receipt of trigger. (Please see SpinAPI documentation for more information on limits of certain Op Codes)

- Instruction Data is the data to be used for certain instructions determined by the Op Code. This corresponds to “inst_data” in SpinAPI. Note for certain instructions this field is not used.

  Figure 4 Below shows detail of the corresponding instruction word as depicted in LabVIEW's block diagram.

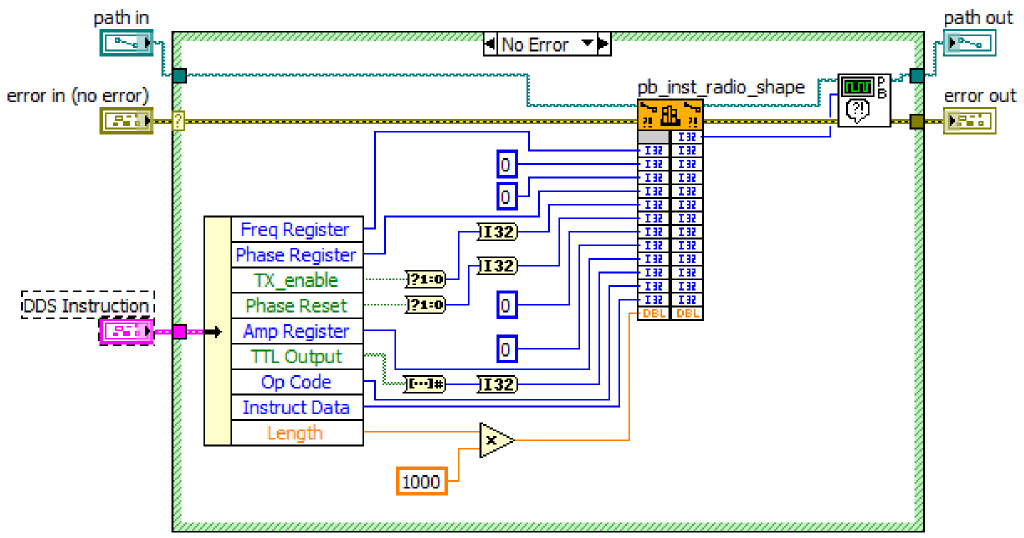


Figure 4: Example of how the instruction is created and sent to the board in the PBLV-DDS Block Diagram

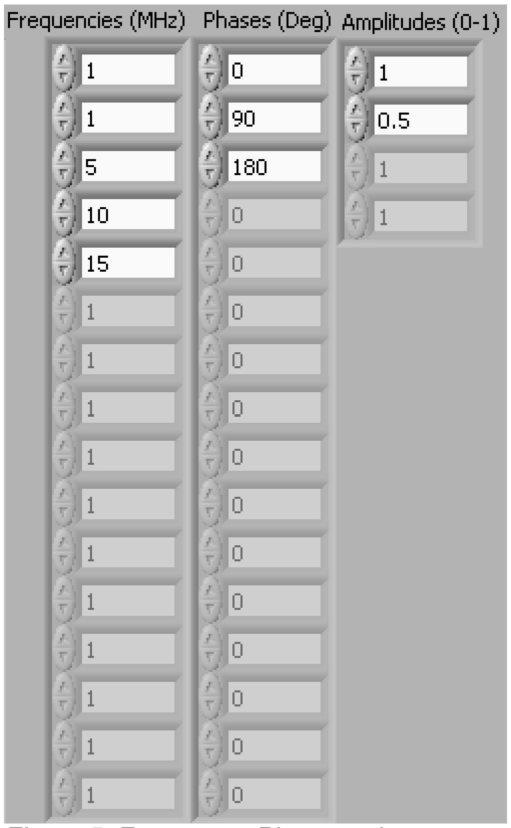
Register Descriptions

In each PBLV-DDS program, there are 3 banks of registers which must be set before programming the board. These are the frequency registers, phase registers and amplitude registers as shown in Figure 5.

Frequencies are specified in MHz and can be programmed in the range specified in your board manual. The registers shown on the front panel range from 0 at the top down to 15 at the bottom, but the number of frequency registers available vary depending on the board. See your board manual for the number of output frequency registers available. The specific frequency registers must be set in the register bank in order to be used in an instruction as described above. If the register box is white, then it will be programmed to the board and can be used. If it is grayed out, it will not be programmed and therefore cannot be used in an instruction. To make a grayed out register box white, simply click inside the box and type the value you want. To gray out a white register box, right-click on the box, select "Data Operations", and then select "Delete Element."

Phases are specified in degrees and can be set from 0.0 to 365.0 degrees. The registers shown on the front panel range from 0 at the top down to 15 at the bottom, but the number of phase registers available vary depending on the board. See your board manual for the number of output phase registers available. The specific phase registers must be set in the register bank in order to be used in an instruction as described above. If the register box is white, then it will be programmed to the board and can be used. If it is grayed out, it will not be programmed and therefore cannot be used in an instruction. To make a grayed out register box white, simply click inside the box and type the value you want. To gray out a white register box, right-click on the box, select "Data Operations", and then select "Delete Element."

Amplitudes are specified as a percentage of full power and ranges from 0.0 to 1.0. The registers shown on the front panel range from 0 at the top down to 3 at the bottom. There are only 4 amplitude registers available. The specific frequency registers must be set in the register bank in order to be used in an instruction as described above. If the register box is white, then it will be programmed to the board and can be used. If it is grayed out, it will not be programmed and therefore cannot be used in an instruction. To make a grayed out register box white, simply click inside the box and type the value you want. To gray out a white register box, right-click on the box, select "Data Operations", and then select "Delete Element."



Frequencies (MHz)	Phases (Deg)	Amplitudes (0-1)
1	0	1
1	90	0.5
5	180	1
10	0	1
15	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	

Figure 5: Frequency, Phase and Amplitude Registers

Path Terminals

All of the SubVIs have path input and output terminals. This is a reference to the path where spinapi.dll is installed on the PC. The default is C:\SpinCore\SpinAPI\dll\spinapi.dll however this may be changed depending on the installation. All subVIs should have the "path in" input connected. For ease of programming, the path terminals can be daisy chained since all subVIs point to the same dll. See Figure 6 and Figure 7 for an example of chaining these terminals.

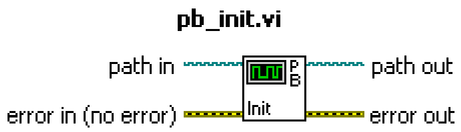


Figure 6: Path and error terminals

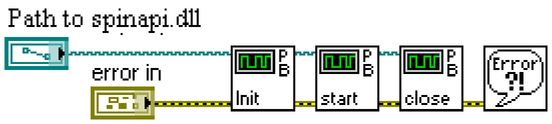


Figure 7: Example of chaining path and error terminals

Error Terminals

All of the SubVIs have error input and output terminals. When a VI is built, the "error out" terminal of a subVI should be connected to the "error in" terminal of the following subVI. These terminals are used to help facilitate sequential execution of the functions, as well as provide debugging information to the user if an error occurs in the VI. See Figure 6 and Figure 7 for an example of these terminals. When chaining, the order of the subVIs corresponds to the order in which the functions will be called.

LabVIEW Program Flow

The LabVIEW Block diagram is set up to independently control the four major functions (start/restart, stop/reset, load, change) using four while loops running in parallel. Within each loop is another loop that continuously waits for the specified button to be pressed. Once the button is pressed, the inner loop will exit and program flow will be passed to the chain of subVIs. After the chain of functions complete, program flow will return to the inner loop to wait for the button again. An example of this is shown in Figure 8.

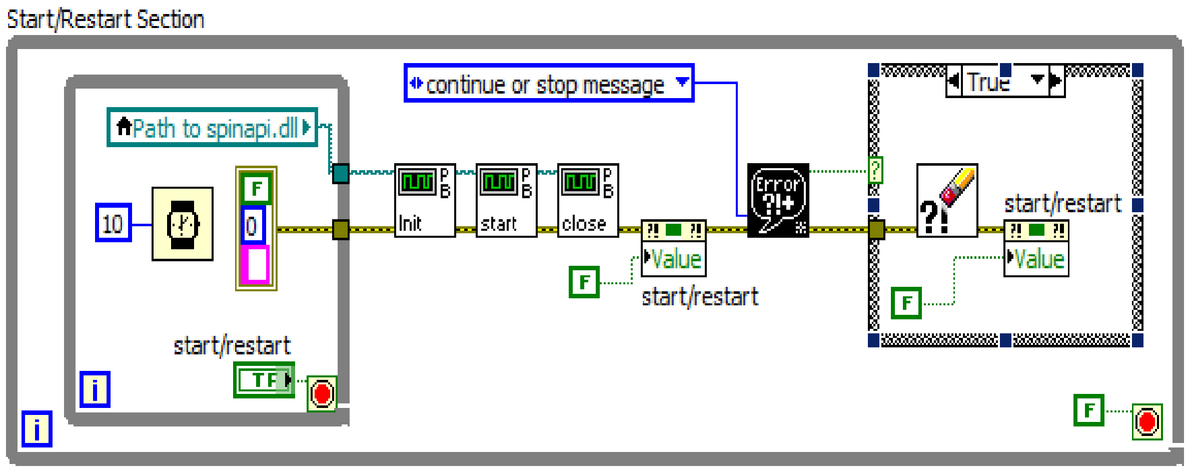


Figure 8: LabVIEW Program Flow Example

IV. VI descriptions

Main VI

PBLV_DDS_Interface.vi

The PBLV-DDS Interface VI allows the user complete control of the PulseBlasterDDS board.

To start the VI press the Run button at the top of the LabVIEW window (for stand-alone programs, the program is run automatically when opened). First you must insert the clock frequency value of the board into the "clock_freq (MHz)" field as well as the path to spinapi.dll as described above. Next, make any instruction set you want by editing the instruction fields. Use the "# of Instructions" field to increase or decrease the number of instructions you want to program. Once you have your instruction set, you can edit the frequency, phase, and amplitude registers to the values you want.

When the instructions and registers are set up correctly, you must first load the board by clicking the "LOAD BOARD" button and then wait until the button becomes inactive. To start the program click on the "START" button, and to stop the program click on the "STOP" button. If there are multiple boards connected to your computer, the "CHANGE BOARD" button will change the board that is being programmed to the one specified by the "board_num" field. All four functions are independent and may be run at any time, however an instruction sequence must be loaded into the board before it can be started. **Caution**: After pressing STOP, all digital output bits may maintain their final state. The RF output will return to 0V when the stop button is pressed.

See Figure 1 for a screen shot of the PBLV-DDS Interface.

Example VIs

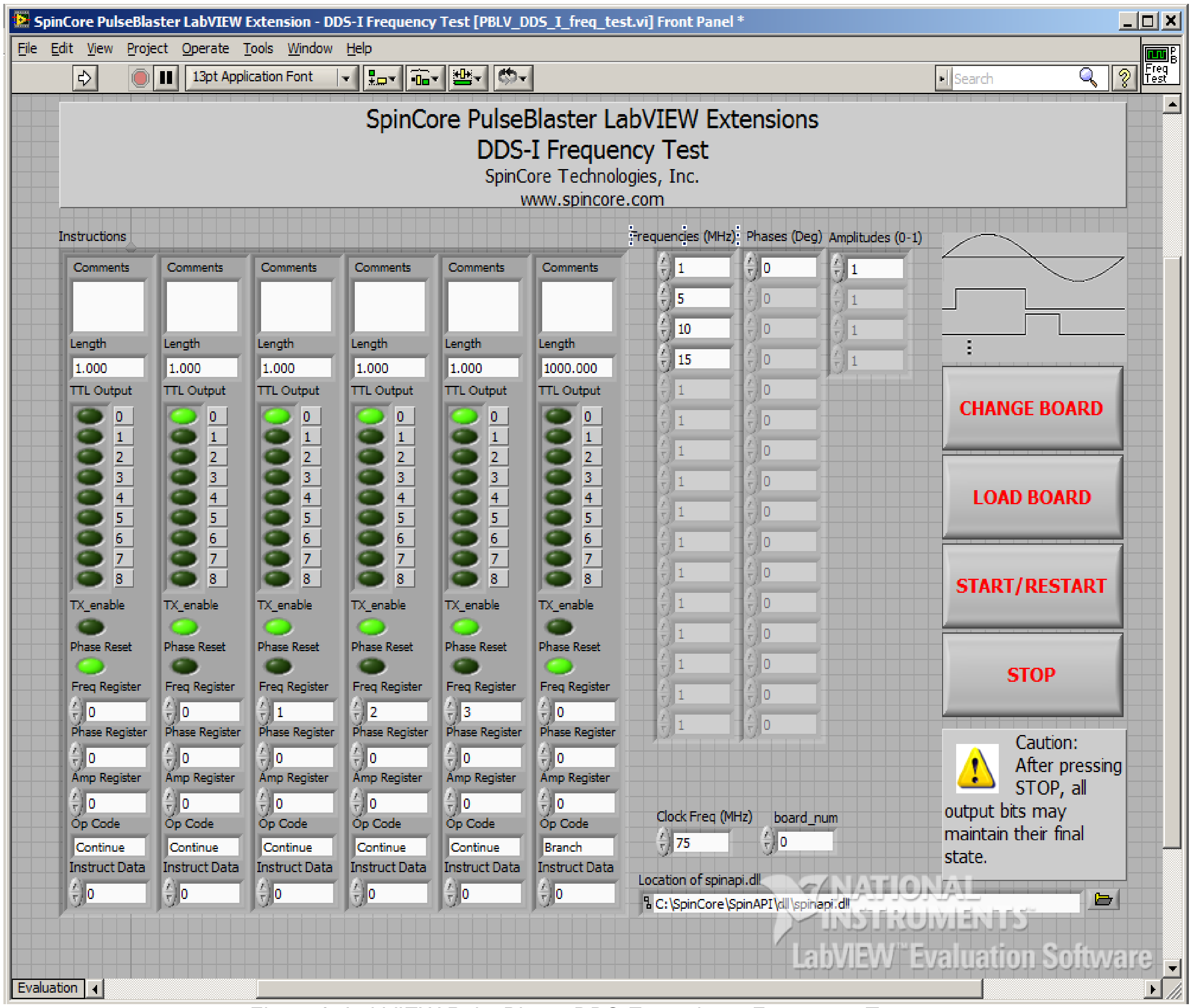
PBLV_DDS_freq_test.vi

The PBLV-DDS Frequency Test VI outputs a sine wave cycling through 4 different frequencies which are set in the frequency registers.

To start the VI press the Run button at the top of the LabVIEW window (for stand-alone programs, the program is run automatically when opened). First you must insert the clock frequency value of the board into the "clock_freq (MHz)" field as well as the path to spinapi.dll as described above. Next, the appropriate frequency, phase, and amplitude registers must be set. Initially frequency registers 0,1,2, and 3 are set to 1,5,10 and 15 MHz respectively, phase register 0 is set to 0 degrees, and amplitude register 0 is set to 1. Then the instructions must be set up appropriately. This VI programs the PBDDS board with 6 instructions. The first instruction resets the phase and the RF output is disabled. The next four instructions enable output for 1 us and frequency register 0, 1, 2, and 3 respectively. All instructions use phase register 0 and amplitude register 0. The last instruction disables the output for 1 ms to allow for oscilloscope triggering. You may change these instructions, and registers as desired.

When the instructions and registers are set up correctly, you must first load the board by clicking the "LOAD BOARD" button and then wait until the button becomes inactive. To start the program click on the "START" button, and to stop the program click on the "STOP" button. If there are multiple boards connected to your computer, the "CHANGE BOARD" button will change the board that is being programmed to the one specified by the "board_num" field. All four functions are independent and may be run at any time, however an instruction sequence must be loaded into the board before it can be started. **Caution**: After pressing STOP, all digital output bits may maintain their final state. The RF output will return to 0V when the stop button is pressed.

See Figure 9 for a screen shot of the PBLV-DDS Frequency Test.


SpinCore PulseBlaster LabVIEW Extension - DDS-I Frequency Test [PBLV_DDS_I_freq_test.vi] Front Panel *
File Edit View Project Operate Tools Window Help
13pt Application Font
Search
CHANGE BOARD
LOAD BOARD
START/RESTART
STOP
NATIONAL INSTRUMENTS LabVIEW™ Evaluation Software
Evaluation

SpinCore PulseBlaster LabVIEW Extensions
DDS-I Frequency Test
SpinCore Technologies, Inc.
www.spincore.com

Instructions

Comments						
Length	1.000	1.000	1.000	1.000	1.000	1000.000
TTL Output 0	○	●	●	●	●	○
TTL Output 1	○	○	○	○	○	○
TTL Output 2	○	○	○	○	○	○
TTL Output 3	○	○	○	○	○	○
TTL Output 4	○	○	○	○	○	○
TTL Output 5	○	○	○	○	○	○
TTL Output 6	○	○	○	○	○	○
TTL Output 7	○	○	○	○	○	○
TTL Output 8	○	○	○	○	○	○
TX_enable	○	●	●	●	●	○
Phase Reset	●	○	○	○	○	●
Freq Register	0	0	1	2	3	0
Phase Register	0	0	0	0	0	0
Amp Register	0	0	0	0	0	0
Op Code	Continue	Continue	Continue	Continue	Continue	Branch
Instruct Data	0	0	0	0	0	0

Frequencies (MHz)	Phases (Deg)	Amplitudes (0-1)
1	0	1
5	0	1
10	0	1
15	0	1
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	

Clock Freq (MHz): 75
board_num: 0
Location of spinapi.dll: C:\SpinCore\SpinAPI\dll\spinapi.dll

Caution: After pressing STOP, all output bits may maintain their final state.

Figure 9: LabVIEW PulseBlasterDDS Extension – Frequency Test

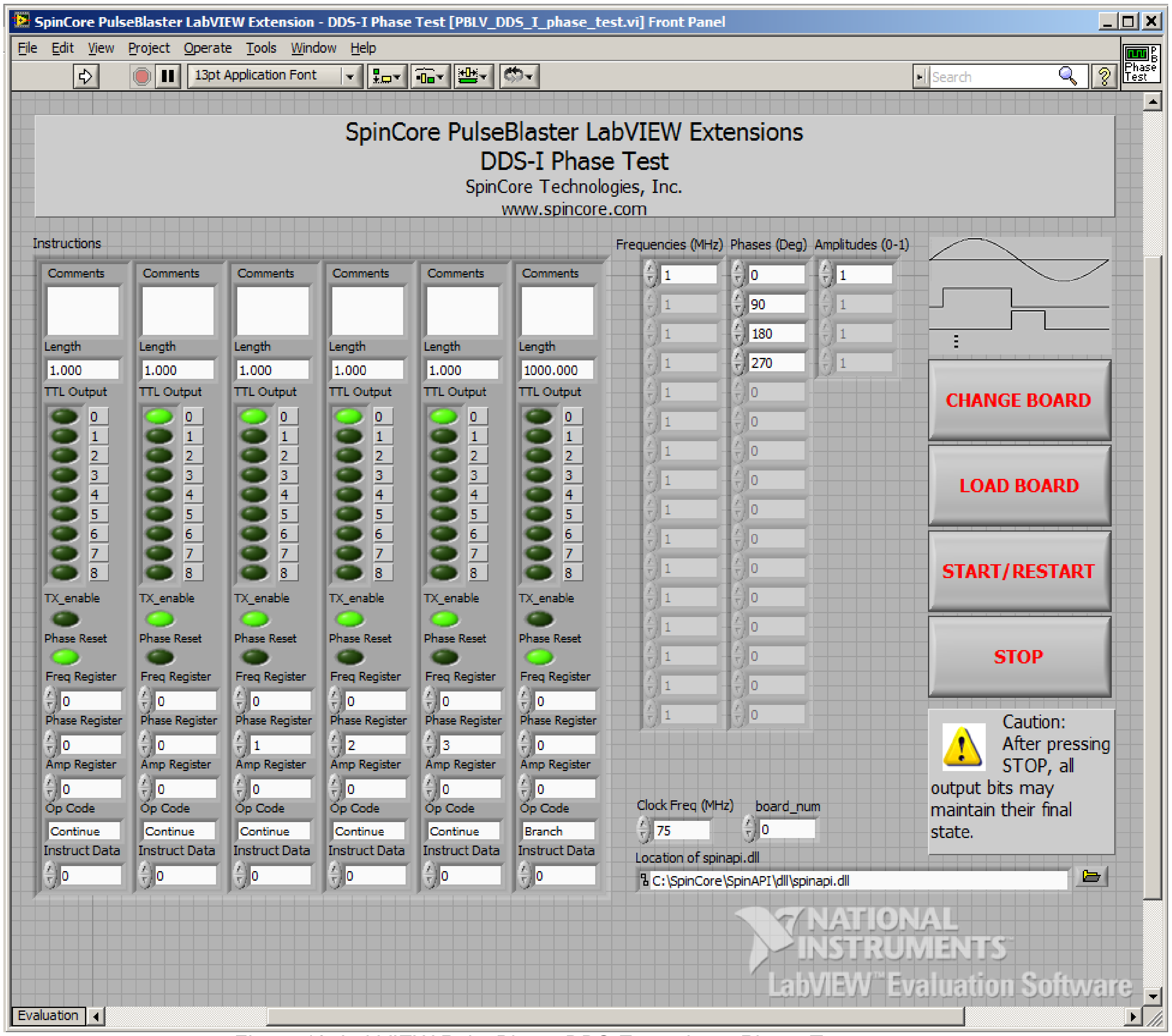
PBLV_DDS_phase_test.vi

The PBLV-DDS Phase Test VI outputs a sine wave cycling through 4 different phases which are set in the phase registers.

To start the VI press the Run button at the top of the LabVIEW window (for stand-alone programs, the program is run automatically when opened). First you must insert the clock frequency value of the board into the “clock_freq (MHz)” field as well as the path to spinapi.dll as described above. Next, the appropriate frequency, phase, and amplitude registers must be set. Initially phase registers 0, 1, 2, and 3 are set to 0°, 90°, 180°, and 270° respectively, frequency register 0 is set to 1 MHz, and amplitude register 0 is set to 1. Then the instructions must be set up appropriately. This VI programs the PBDDS board with 6 instructions. The first instruction resets the phase and the RF output is disabled. The next four instructions enable output for 1 us and use phase registers 0, 1, 2, and 3 respectively. All instructions use frequency register 0 and amplitude register 0. The last instruction disables the output for 1 ms to allow for oscilloscope triggering. You may change these instructions, and registers as desired.

When the instructions and registers are set up correctly, you must first load the board by clicking the “LOAD BOARD” button and then wait until the button becomes inactive. To start the program click on the “START” button, and to stop the program click on the “STOP” button. If there are multiple boards connected to your computer, the “CHANGE BOARD” button will change the board that is being programmed to the one specified by the “board_num” field. All four functions are independent and may be run at any time, however an instruction sequence must be loaded into the board before it can be started. **Caution**: After pressing STOP, all digital output bits may maintain their final state. The RF output will return to 0V when the stop button is pressed.

See Figure 10 for a screen shot of the PBLV-DDS Phase Test.



SpinCore PulseBlaster LabVIEW Extension - DDS-I Phase Test [PBLV_DDS_I_phase_test.vi] Front Panel

SpinCore PulseBlaster LabVIEW Extensions
DDS-I Phase Test
SpinCore Technologies, Inc.
www.spincore.com

Instructions

Comments						
Length	1.000	1.000	1.000	1.000	1.000	1000.000
TTL Output 0	●	🟩	🟩	🟩	🟩	●
TTL Output 1	●	●	●	●	●	●
TTL Output 2	●	●	●	●	●	●
TTL Output 3	●	●	●	●	●	●
TTL Output 4	●	●	●	●	●	●
TTL Output 5	●	●	●	●	●	●
TTL Output 6	●	●	●	●	●	●
TTL Output 7	●	●	●	●	●	●
TTL Output 8	●	●	●	●	●	●
TX_enable	●	🟩	🟩	🟩	🟩	●
Phase Reset	🟩	●	●	●	●	🟩
Freq Register	0	0	0	0	0	0
Phase Register	0	0	1	2	3	0
Amp Register	0	0	0	0	0	0
Op Code	Continue	Continue	Continue	Continue	Continue	Branch
Instruct Data	0	0	0	0	0	0

Frequencies (MHz)	Phases (Deg)	Amplitudes (0-1)
1	0	1
1	90	1
1	180	1
1	270	1
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	

Clock Freq (MHz): 75
board_num: 0
Location of spinapi.dll: C:\SpinCore\SpinAPI\dll\spinapi.dll

Caution: After pressing STOP, all output bits may maintain their final state.

Figure 10: LabVIEW PulseBlasterDDS Extension – Phase Test

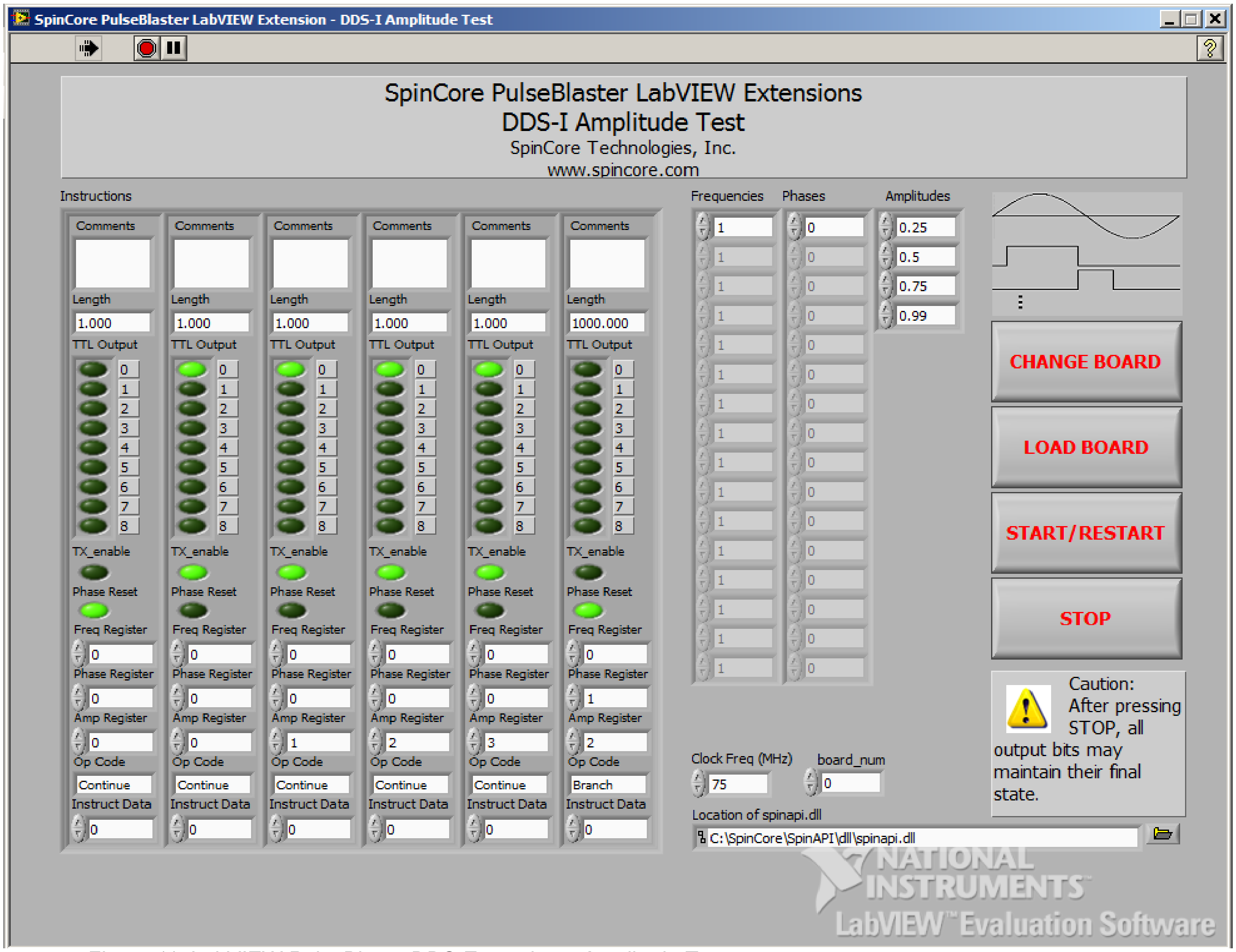
PBLV_DDS_amp_test.vi

The PBLV-DDS Amplitude Test VI outputs a sine wave cycling through 4 different amplitudes which are set in the amplitude registers.

To start the VI press the Run button at the top of the LabVIEW window (for stand-alone programs, the program is run automatically when opened). First you must insert the clock frequency value of the board into the “clock_freq (MHz)” field as well as the path to spinapi.dll as described above. Next, the appropriate frequency, phase, and amplitude registers must be set. Initially amplitude registers 0, 1, 2, and 3 are set to 0.25, 0.5, 0.75, and 0.99 respectively (this signifies 25%, 50%, 75% and 99% amplitude), frequency register 0 is set to 1 MHz, and amplitude register 0 is set to 1. *Note that due to an unknown issue in some boards, if the amplitude is set to 1 (100%) the output waveform is inverted.* Next the instructions must be set up appropriately. This VI programs the PBDDS board with 6 instructions. The first instruction resets the phase and the RF output is disabled. The next four instructions enable output for 1 us and use amplitude register 0, 1, 2, and 3 respectively. All instructions use frequency register 0 and phase register 0. The last instruction disables the output for 1 ms to allow for oscilloscope triggering. You may change these instructions and registers as desired.

When the instructions and registers are set up correctly, you must first load the board by clicking the “LOAD BOARD” button and then wait until the button becomes inactive. To start the program click on the “START” button, and to stop the program click on the “STOP” button. If there are multiple boards connected to your computer, the “CHANGE BOARD” button will change the board that is being programmed to the one specified by the “board_num” field. All four functions are independent and may be run at any time, however an instruction sequence must be loaded into the board before it can be started. **Caution**: After pressing STOP, all digital output bits may maintain their final state. The RF output will return to 0V when the stop button is pressed.

See Figure 11 for a screen shot of the PBLV-DDS Phase Test.


SpinCore PulseBlaster LabVIEW Extension - DDS-I Amplitude Test
CHANGE BOARD
LOAD BOARD
START/RESTART
STOP
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software

SpinCore PulseBlaster LabVIEW Extensions
DDS-I Amplitude Test
SpinCore Technologies, Inc.
www.spincore.com

Instructions

Comments						
Length	1.000	1.000	1.000	1.000	1.000	1000.000
TTL Output 0	○	●	●	●	●	○
TTL Output 1	○	○	○	○	○	○
TTL Output 2	○	○	○	○	○	○
TTL Output 3	○	○	○	○	○	○
TTL Output 4	○	○	○	○	○	○
TTL Output 5	○	○	○	○	○	○
TTL Output 6	○	○	○	○	○	○
TTL Output 7	○	○	○	○	○	○
TTL Output 8	○	○	○	○	○	○
TX_enable	○	●	●	●	●	○
Phase Reset	●	○	○	○	○	●
Freq Register	0	0	0	0	0	0
Phase Register	0	0	0	0	0	1
Amp Register	0	0	1	2	3	2
Op Code	Continue	Continue	Continue	Continue	Continue	Branch
Instruct Data	0	0	0	0	0	0

Frequencies	Phases	Amplitudes
1	0	0.25
1	0	0.5
1	0	0.75
1	0	0.99
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	
1	0	

Clock Freq (MHz) 75
board_num 0

Location of spinapi.dll
C:\SpinCore\SpinAPI\dll\spinapi.dll

Caution: After pressing STOP, all output bits may maintain their final state.

Figure 11: LabVIEW PulseBlasterDDS Extension – Amplitude Test

Basic SubVIs

These SubVIs use Call Library Function Nodes to access SpinAPI's C-functions. By default these nodes use the C-calling convention. For customizable VIs that use the WINAPI calling convention please contact SpinCore Technologies via the web forum.

pb_error_handler.vi

This VI will catch an error from SpinAPI and pass it into the LabVIEW code for debugging purposes. It takes the return code from any function call and determines if there is an error (return code is a negative number). If an error occurred, the VI calls pb_get_error from SpinAPI which returns the error text. This error text added to the LabVIEW error cluster and the error status bit is set true. All functionality will be bypassed if a previous error is passed into the VI.

pb_init.vi

Corresponds to the SpinAPI function int pb_init()

This VI initializes the SpinCore board. It calls the pb_init function from SpinAPI using the Call Library Function Node. It then checks for an error using the pb_error_handler. If there was an error at this stage, the VI determines if the board was not previously closed, if so the VI clears the error, closes the board and retries pb_init. All functionality will be bypassed if an error is passed into the VI.

pb_set_default.vi

Corresponds to the SpinAPI function int pb_set_default()

This VI sets the RadioProcessor/PBDDS to its default state. This function should generally be called after pb_init() to make sure the RadioProcessor is in a usable state. It is REQUIRED that this be called at least once after the board is powered on. It calls the pb_set_defaults function from SpinAPI using the Call Library Function Node. It then checks for an error using the pb_error_handler. All functionality will be bypassed if an error is passed into the VI.

pb_core_clock.vi

Corresponds to the SpinAPI function void set_clock(double clock_freq)

This VI sets the clock frequency of the board in MHz. It calls the pb_set_clock function from SpinAPI using the Call Library Function Node. All functionality will be bypassed if an error is passed into the VI.

pb_start_programming.vi

Corresponds to the SpinAPI function int pb_start_programming(int device)

This VI initializes the system to receive programming information. It calls the pb_start_programming function from SpinAPI using the Call Library Function Node. The Device Enum Control sets the device input of the pb_start_programming function. It then checks for an error using the pb_error_handler. All functionality will be bypassed if an error is passed into the VI.

pb_set_freq.vi

Corresponds to the SpinAPI function int pb_set_freq(double freq)

This VI will write the given frequency, in MHz to a frequency register on a DDS enabled board. and then check for errors. To do this, first call pb_start_programming.vi, and pass it "Frequency Registers". The first call pb_set_freq.vi will then program frequency register 0, the second call will program frequency register 1, etc. When you have programmed all the registers you intend to, call pb_stop_programming.vi. All functionality will be bypassed if an error is passed into the VI.

pb_set_phase.vi

Corresponds to the SpinAPI function int pb_set_phase(double phase)

This VI will write the given phase to a phase register on DDS enabled boards and then check for errors. To do this, first call pb_start_programming.vi, and specify the appropriate bank of phase registers (TX Phase Registers or RX Phase Registers) as the argument. The first call pb_set_phase.vi will then program phase register 0, the second call will program phase register 1, etc. When you have programmed all the registers you intend to, call pb_stop_programming.vi. The given phase value may be rounded to fit the precision of the board.. All functionality will be bypassed if an error is passed into the VI.

pb_set_amp.vi

Corresponds to the SpinAPI function int pb_set_amp(float amp, int addr)

This VI will write the given amplitude to the given amplitude register on DDS enabled boards and then check for errors. The VIs pb_start_programming.vi and pb_stop_programming.vi are not needed to set the amplitude. The given amplitude value may be rounded to fit the precision of the board. All functionality will be bypassed if an error is passed into the VI.

pb_inst_dds.vi

Corresponds to the SpinAPI function int pb_inst_radio_shape *(int freq, int cos_phase, int sin_phase, int tx_phase, int tx_enable, int phase_reset, int trigger_scan, int use_shape, int amp, int flags, int inst, int inst_data, double length) with shape functionality disabled.*

This VI sends one instruction of the DDS pulse program to the board. It calls the pb_inst_radio_shape function from SpinAPI using the Call Library Function Node. The instruction control is the 80-bit VLIW to be programmed to the board. It contains the 4 bit frequency register, 4 bit phase register, 1 bit tx_enable, 1 bit phase reset, 9 bit digital output, Op Code (which instruction to be executed), instruction data and length that the instruction should be carried out It then checks for an error using the pb_error_handler. All functionality will be bypassed if an error is passed into the VI.

pb_stop_programming.vi

Corresponds to the SpinAPI function int pb_stop_programming()

This VI tells the board that programming the is complete. It calls the pb_stop_programming function from SpinAPI using the Call Library Function Node. It then checks for an error using the pb_error_handler. All functionality will be bypassed if an error is passed into the VI.

pb_close.vi

Corresponds to the SpinAPI function int pb_close()

This VI releases SpinCore board. It calls the pb_close function from SpinAPI using the Call Library Function Node. It then checks for an error using the pb_error_handler. All functionality will be bypassed if a previous error is passed into the VI.

pb_start.vi

Corresponds to the SpinAPI function int pb_start()

Once board has been programmed, this instruction will start execution of pulse program. It calls the pb_start function from SpinAPI using the Call Library Function Node. It then checks for an error using the pb_error_handler. All functionality will be bypassed if an error is passed into the VI.

pb_stop.vi

Corresponds to the SpinAPI function int pb_stop()

This VI stops output of the board. It calls the pb_stop function from SpinAPI using the Call Library Function Node. It then checks for an error using the pb_error_handler. All functionality will be bypassed if an error is passed into the VI.

V. Contact Information

SpinCore Technologies, Inc.
4631 NW 53rd Avenue, SUITE 103
Gainesville, FL 32653
USA

Telephone (USA):	**352-271-7383**
Fax (USA):	**352-371-8679**
Website:	http://www.spincore.com
Web Form:	http://www.spincore.com/contact.shtml