PulseBlasterESR-PRO-200-cPCI™

Owner's Manual

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 aim 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. Introduction.....5

II. Device Description and Specifications.....6

Device Overview.....6
System Architecture.....6
Output Signals.....6
Timing Characteristics.....7
Instruction Set (Flow Control).....7
On-Board Clock.....7
Device Memory.....7
External Inputs.....7

Summary.....8
Output Signals.....8
Timing Characteristics.....8
Instruction Set (Program Flow).....8
On-Board Clock.....8
Device Memory.....8
External Input Specifications.....8

III.Installation.....9

Installing the PulseBlasterESR-PRO-200-cPCI.....9

IV. Programming PulseBlaster Devices.....10

Introduction.....10

Programming Paradigm.....10

PulseBlaster Interpreter.....11

PulseBlaster.NET.....12

MATLAB GUI.....13

LabVIEW Extensions.....14

C/C++ Programming.....15

V. Connecting to PulseBlaster Devices.....16

Connector Information.....16
BNC Header Connectors.....16
IDC Headers.....17
HWTrig/Reset Header.....19

VI. Related Products and Accessories.....21

VI.Contact Information.....22

I.Document Information.....22

I. Introduction

The PulseBlasterESR-PRO-200-cPCI is a programmable multichannel pulse/delay generator in the CompactPCI form factor. This device is capable of generating pulses and delays with duration ranging from 5 ns to 2.3 x 10^7 seconds (260 days). The PulseBlasterESR-PRO-200-cPCI features outputs on BNC connectors and IDC headers, highly flexible program flow control, and can accommodate pulse programs containing up to 4096 instructions (if more instructions are necessary, please contact SpinCore Technologies, Inc. for more information).

The PulseBlasterESR-PRO-200-cPCI is capable of generating complex output pulse sequences featuring very short and very long pulses and delays in the same sequence. Pulses/delays can be as short as one clock cycle, to as long as 2^{52} clock cycles. Regardless of the pulse/delay length, timing resolution is only one clock cycle.

The intelligence of the PulseBlasterESR-PRO-200-cPCI comes from its proprietary PulseBlaster processor core. Unlike general-purpose processors, the PulseBlaster processor core features a highly optimized instruction set designed for timing applications. A unique feature of the PulseBlaster processor core is the user can vary the execution time of instructions.

User interaction with the PulseBlasterESR-PRO-200-cPCI can be accomplished in several different ways. Many Graphical User Interfaces (GUI) are offered by SpinCore Technologies that allow a user with little to no programming experience to create, edit, save and run pulse programs. Development environments like LabVIEW, MATLAB, and C/C++, are supported by using the SpinAPI package, a dedicated Application Programming Interface (API) package. The SpinAPI package can be used with most Windows programming environments and can be used with other operating systems, including Linux. Software for the PulseBlasterESR-PRO-200-cPCI is available on our website: http://www.spincore.com/support/.

II. Device Description and Specifications

Device Overview

System Architecture

The major building blocks of the PulseBlaster processor core are the SRAM memory, the microcontroller core (uPC), the integrated bus controller (IBC), the counter, and the output buffers. All components are located on a single silicon chip, making the design a System-on-a-Chip (SOC). User control of the device is provided through the integrated bus controller (IBC) using the CompactPCI bus. The figure below shows the block diagram of the PulseBlaster processor core.

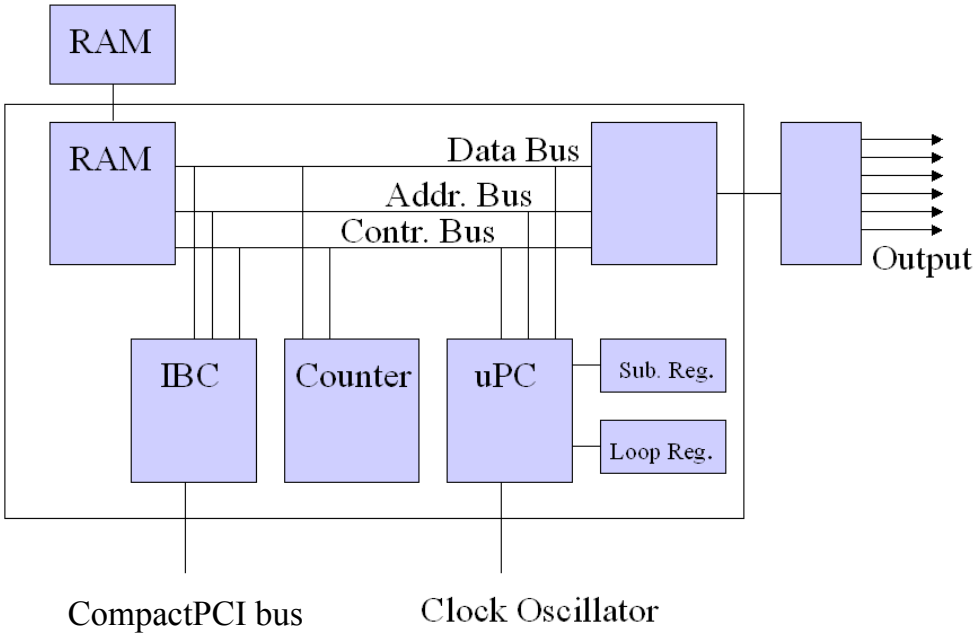

Figure 1: Block Diagram of the PulseBlaster processor core. All the components are placed on a single silicon chip, making the design a System-on-a-Programmable-Chip (SOPC). The clock oscillator signal is derived from an on-chip PLL circuit typically using a 50 MHz on-board reference clock.

Output Signals

Pulse sequences are output digitally using 3.3 V Low Voltage TTL (LVTTL). If the channel is on, then the device will output 3.3 V unterminated, and if the channel is off, the device will output 0.0 V, unterminated. Each channel is capable of delivering up to ±25 mA per channel. If more output current is necessary, the individual channels can be driven in parallel.

Outputs are available on BNC connectors and IDC headers. The BNC connectors are impedance matched to 50 Ω and are located on the mounting bracket. All outputs are available on IDC headers located

on the board's surface. Status bits are also available on IDC headers, allowing the user or an external device to monitor the status of the PulseBlaster device. For more information about the connections available, please see “V. Connecting to PulseBlaster Devices.”

Timing Characteristics

The innovative architecture of the PulseBlaster processor core allows pulses/delays to be as short as one clock cycle (5 ns) and last up to 2^{52} clock cycles (260 days). Regardless of the timing duration used, the timing resolution is one clock cycle. A program can have both long and short pulses/delays in the same program, and both will be accurate to a single clock cycle.

Instruction Set (Flow Control)

The PulseBlaster devices features a set of commands for highly flexible program flow control. The specialized microcontroller allows for programs to include branches, subroutines, and loops up to 8 nested levels deep. These commands allow the user to perform repetitious events with ease.

Instruction execution time can be set by the user. The minimum instruction execution time is five clock cycles (25 ns), and the maximum is 2^{52} clock cycles (260 days). To create pulses with duration shorter than five clock cycles, the Short Pulse Feature will need to be used. The Short Pulse Feature allows pulse duration to be as short as one clock cycle (5 ns); however, at least five clock cycles are still required for the PulseBlaster processor core to process the instruction. For more information about PulseBlaster processor core architecture, please see the “Instruction Set Architecture” of Using SpinAPI in C/C++ in PulseBlaster Programming, found at: http://www.spincore.com/support/spinapi/using_spin_api_pb.shtml.

On-Board Clock

The PulseBlaster device accepts an on-board 50 MHz oscillator. The on-board oscillator's frequency is internally multiplied to 200 MHz by using a Phase-Locked Loop (PLL). The device can be externally clocked by removing the oscillator and attaching an equivalent external source to the oscillator mount. The PulseBlaster device does not have on-board termination for the clock input signal. Applying less than 0.0 V or more than 3.3 V to the clock input pins will damage the PulseBlaster device.

CAUTION: *Incorrectly attaching an external clock source will damage the PulseBlasterESR-PRO-200-cPCI. Contact SpinCore Technologies, Inc. if you would like information on how to use an external clock source.*

Device Memory

The memory will hold up to 4096 instructions. Programs do not need to fill the memory; they can be as short as desired. If larger device memory is required, please contact SpinCore Technologies, Inc.

External Inputs

The PulseBlaster device has 2 external inputs for device control: HW_Trig and HW_Reset. If HW_Trig is activated, then the device will start running the program (the PulseBlaster device must be programmed first). If HW_Reset is activated, then the device will be stopped. The two separate lines combine the convenience of triggering (e.g., in cardiac gating) with the safety of a "stop/reset" line. External inputs are described further in “V. Connecting to PulseBlaster Devices.”

Summary

Output Signals

- 4 bracket-mounted BNC connectors, impedance matched to 50 Ω. 3.3 V LVTTL.
- 21 individually controlled digital output signals on IDC headers. 3.3 V LVTTL.
- ±25 mA output current per output line.
- 4 status outputs on IDC header.

Timing Characteristics

- Shortest pulse/delay: 1 clock cycle (5.0 ns).
- Longest pulse/delay: 2^{52} clock cycles (260 days).
- Pulse resolution: 1 clock cycle (5.0 ns), regardless of pulse length.

Instruction Set (Program Flow)

- User-programmable instruction execution time.
- Subroutines can be nested up to 8 levels deep.
- Loops can be nested up to 8 levels deep.
- 20-bit loop counters (maximum of 1,048,576 repetitions).
- Branch range includes the entire memory.
- Latency after trigger (WAIT state) – 8 clock cycle latency (40 ns at 200 MHz), adjustable to 20 seconds in duration.
- External trigger and reset provide external control of the PulseBlaster device.

On-Board Clock

- 50 MHz on-board oscillator.
- 200 MHz internal clock frequency by use of a Phase-locked Loop.
- External clock source may be used (contact SpinCore Technologies, Inc. for information on using an external clock source).

Device Memory

- Up to 4096 instructions.

External Input Specifications

- External triggering and reset. Tolerance: 0.0 V minimum, 3.3 V maximum.

III. Installation

Installing the PulseBlasterESR-PRO-200-cPCI

To install the board you must uninstall any previous versions of SpinAPI and complete the following:

1. Install the latest version of SpinAPI found at: http://www.spincore.com/support/spinapi/.
    - SpinAPI is a custom Application Programming Interface developed by SpinCore Technologies, Inc. for use with the PulseBlaster board. It can be utilized using C/C++ or graphically using the options in the next section below. The API will also install the necessary drivers.
    - There is also a package with example programs available to download.

2. Shut down the computer, unplug the power cord, insert the PulseBlaster card into an available CompactPCI computer bus and fasten the card securely in place.

3. Plug the power cord back in, turn on the computer and follow the installation prompts.

We recommend running example programs after you installed the PulseBlasterESR to verify that your device is functional. These example files can be found at: http://www.spincore.com/support/spinapi/spinapi_examples.shtml. Examples can be downloaded individually or all at once using the Complete SpinAPI Examples Installer.

Be sure to download either the 32-bit or 64-bit version which matches the operating system of your computer. Save the .exe file to your Desktop when prompted to select a location and run the file. The installer will begin and ask for a location to save the example files. It is recommended to save these examples under “C:\SpinCore\SpinAPI\” for better organization. A new folder “examples” can be created within SpinAPI for this purpose. After selecting a destination folder, the installer will place the selected example files at that location.

IV. Programming PulseBlaster Devices

Introduction

SpinCore Technologies provides several Graphical User Interfaces (GUIs) for creating/editing/saving programs, programming PulseBlaster devices, and starting/stopping programs. The GUIs are the *PulseBlaster Interpreter, PulseBlaster.NET*, MATLAB GUI, LabVIEW extensions, and C/C++ interface.

All SpinCore Technologies, Inc. software is available for free at our website: http://www.spincore.com/support.

Programming Paradigm

The PulseBlasterESR-PRO-200-cPCI can be programmed with an arbitrary sequence of intervals. Each interval can be of unique length, and up to 4096 intervals can be accommodated per sequence. Because each interval can be a pulse or a delay, each interval involves the loading of two basic parameters: the output state (logical 0 or 1), and the duration of the state (in nanoseconds, microseconds, milliseconds).

The low-level interaction is accomplished through a dedicated Application Programming Interface (API) package called SpinAPI. SpinAPI is available for download on SpinCore Technologies' website: http://www.spincore.com/. Virtually any higher-level application package (Matlab, LabVIEW etc.) can interact with the board through the provided SpinAPI functions.

PulseBlaster Interpreter

PulseBlaster Interpreter features large buttons for program editing and execution, while using text to input instructions. *PulseBlaster Interpreter* is included with the SpinAPI software suite. When the SpinAPI software suite is installed, a shortcut to *PulseBlaster Interpreter* is automatically placed on the desktop. More information on *PulseBlaster Interpreter* is available at: http://www.spincore.com/support/SPBI/.

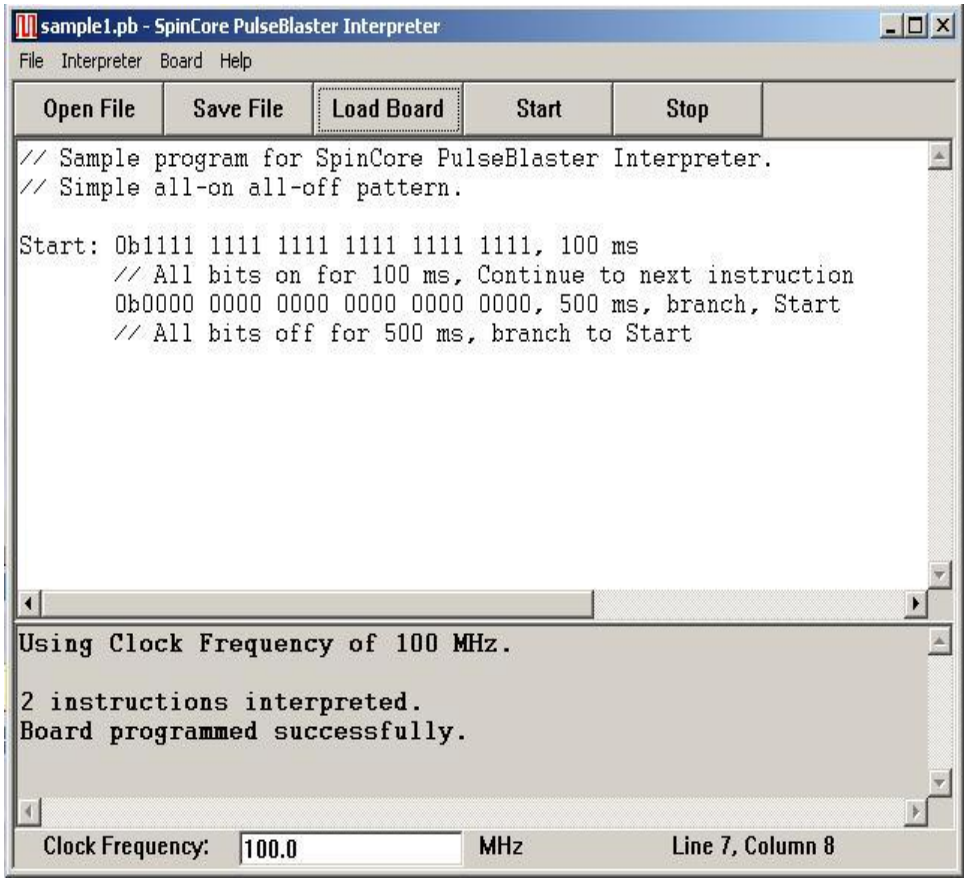

Figure 2: Screenshot of the PulseBlaster Interpreter. The example shown creates a pulse that toggles all output bits on for 100 ms, then off for 500ms, and repeats. PulseBlaster Interpreter is included with the SpinAPI software suite.

PulseBlaster.NET

The *PulseBlaster.NET* GUI features an easy-to-understand interface. Programming is no longer done in text, but instead presented as a simple visual metaphor. Outputs are set by using check-boxes, and program flow instructions are set by using a drop-box. This GUI is ideal for users with no programming experience. *PulseBlaster.NET* is capable of saving and loading files, allowing work to be saved and/or transferred to another computer. Programming errors (e.g., invalid OpCode) are indicated to the user immediately, reducing debugging time.

The latest version of the *PulseBlaster.NET* and supporting documentation is available on our website at http://www.spincore.com/support/net/.

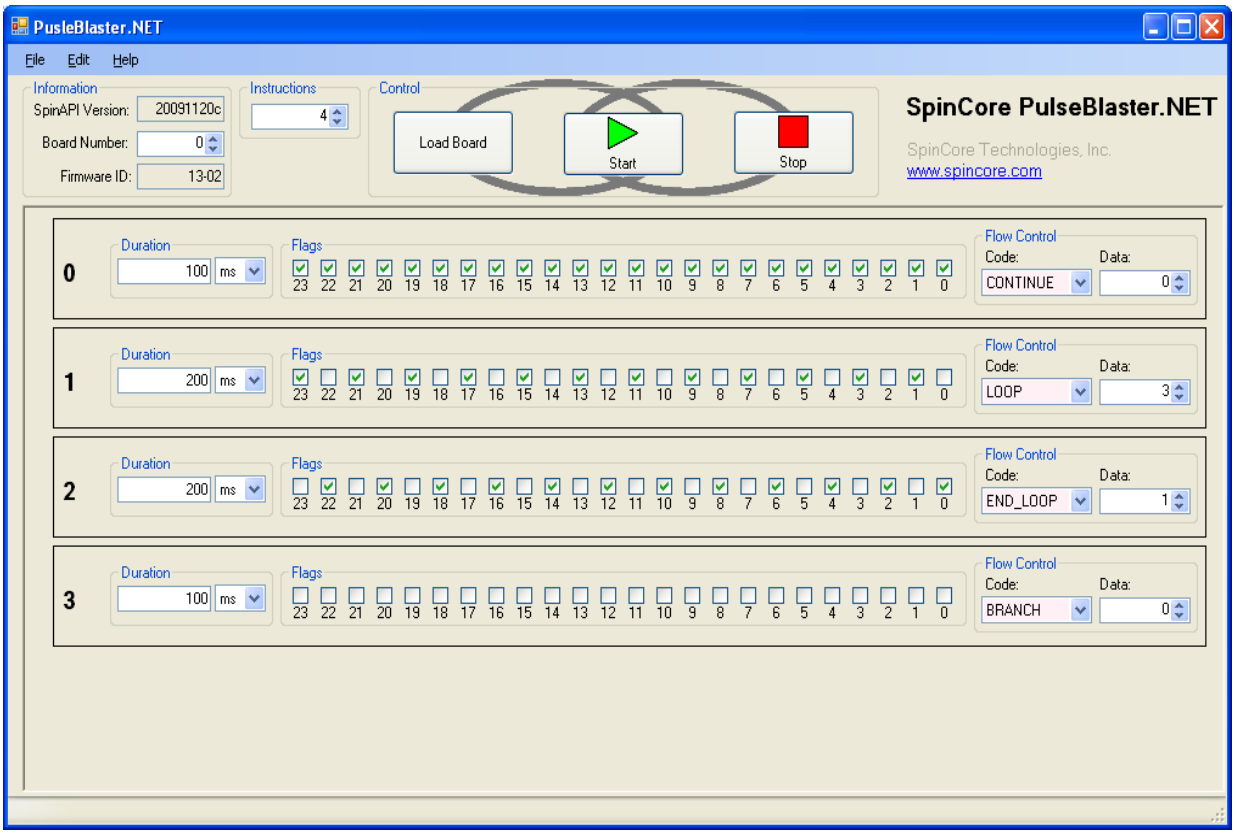

Figure 3: The PulseBlaster.NET interface. This interface is easy to use, featuring buttons, checkboxes, dropboxes, and very little text use. This GUI is ideal for users with no programming experience. Pulse programs can be saved, opened, and edited with ease.

MATLAB GUI

The MATLAB GUI provides a simple yet powerful interface for controlling PulseBlaster devices. Similar to the *PulseBlaster.NET* interface, outputs are set by using check-boxes, program flow instructions are selected from a drop-box, and instruction durations are typed in. The full instruction set can be utilized, and multiple devices can be controlled. Device selection is done by typing in the board's number, and device control is done by clicking large buttons at the top of the GUI. The MATLAB GUI is compatible with programs from *PulseBlaster.NET.*

MATLAB is required for using this GUI. The latest version of MATLAB GUI and download information can be found at: http://www.spincore.com/support/PulseBlasterMatlabGUI/

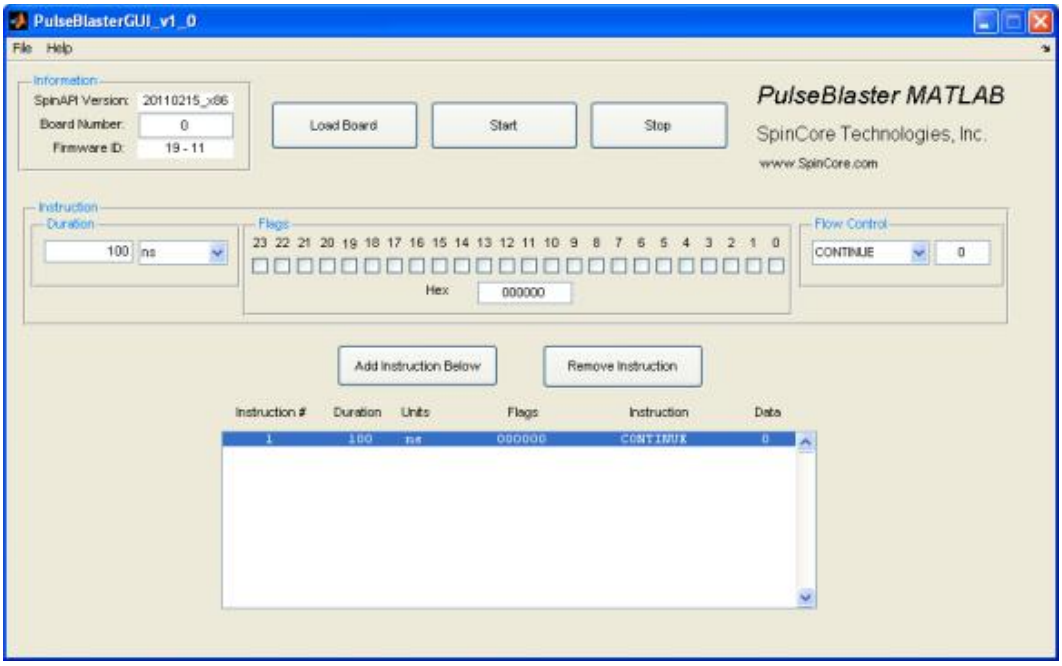

Figure 4: The MATLAB GUI. This GUI controls outputs by using checkboxes, program flow instructions are performed through drop boxes, and instruction durations are input by using text. This GUI can control multiple devices, and device control is done by clicking large buttons. Programs can be saved/loaded/edited with ease.

LabVIEW Extensions

This GUI features large buttons for programming the device and starting/stopping the pulse program. Pulse sequence generation is done by turning a channel on or off by left clicking on the button. The only text used in this GUI is to set the program flow instruction (continue, branch, etc.) and the duration of the instruction. This is another GUI that is ideal for users with no programming experience.

For users with LabVIEW and programming experience, we've provided basic sub-VIs (Virtual Instruments) that add PulseBlaster interaction with your own LabVIEW programs, allowing users to create a custom interface. LabVIEW can utilize the SpinAPI library for more functionality.

This GUI can be run on the free LabVIEW 9.0 Runtime Engine. For supporting documentation and download information, please visit: http://www.spincore.com/support/PBLV/.

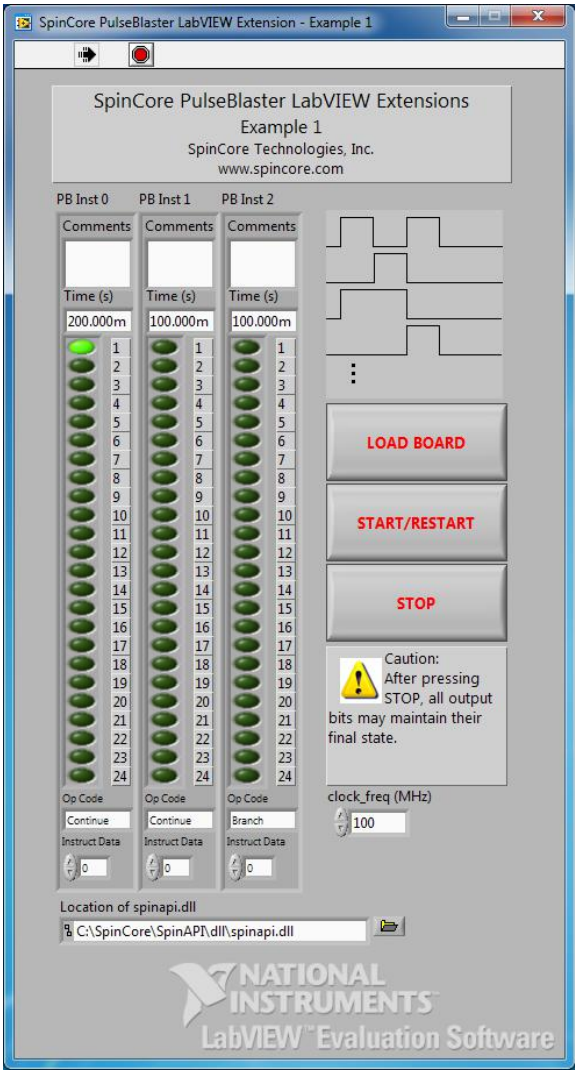

Figure 5: PulseBlaster LabVIEW Extensions User Interface. This GUI features very little text. This is ideal for users with no programming experience.

C/C++ Programming

Programming PulseBlaster devices using C/C++ is easier than ever. SpinCore Technologies offers a pre-configured C/C++ compiler, and the SpinAPI library provides functions for programming PulseBlaster devices. Programming using C/C++ allows the user to fully utilize the device, including the use of interrupt features. Additionally, repetitive instructions may be easier to program using C/C++ because you can copy and paste lines of code, which may be faster than performing many instructions using a GUI.

SpinCore Technologies provides many example programs that demonstrate features of PulseBlaster devices. Example programs are available by going to "**Start>Programs>SpinCore>SpinAPI**." Inside this folder, open the folder for your device (e.g., PBESR-PRO for PulseBlasterESR-PRO devices). To run an example program, double-click on the executable file (*.exe). A PulseBlaster device must be installed for the program to run properly.

An easy method of creating programs using C/C++ is to modify an existing example program, and recompile. To recompile, select the "Rebuild All" button (see the figure below). The pre-configured compiler will create an executable file that will handle device programming, and will start the pulse program. Double click on the executable file (*.exe) to run the program.

Download information and installation instructions are available at our website at: http://www.spincore.com/support/spinapi/. A description of SpinAPI and the included C functions can be found at: http://www.spincore.com/support/spinapi/using_spin_api_pb.shtml

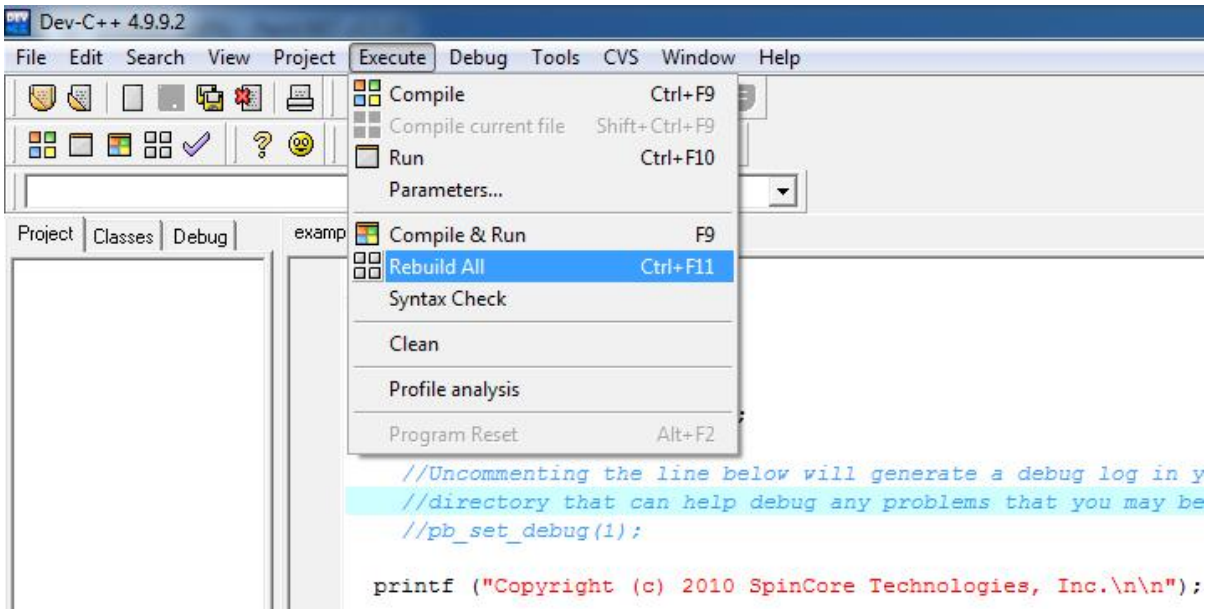

Figure 6: Compiling a C program to run PulseBlaster devices is easy! Programs can be made quickly by using the available pre-configured compiler and the supplied example programs. Open an example program with the supplied compiler, modify to your needs, then click "Rebuild All." The compiler will create an executable file that will handle device programming, and starting your program. The compiler is available at http://www.spincore.com/support/spinapi.

V. Connecting to PulseBlaster Devices

Connector Information

BNC Header Connectors

The four bracket-mounted BNC connectors outputs the first four bits of the output word (bit 0 to bit 3). BNC 0 outputs bit 0, BNC 1 outputs bit 1, etc. The figure below shows the location of the BNC connectors on the mounting-bracket.

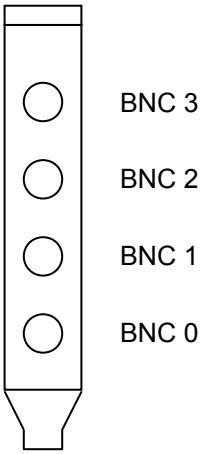

Figure 7: BNC connector locations. These connectors are located on the mounting bracket. The BNC connectors are impedance-matched to 50 Ω.

If using a high input impedance oscilloscope with BNC connecters to monitor the PulseBlaster's output via the BNC connectors, place a resistor that matches the characteristic impedance of the transmission cable in parallel with the coaxial cable at the oscilloscope input (e.g., a 50 Ω resistor with a 50 Ω transmission cable, see Figures 8 and 9 below). When using an oscilloscope with an adjustable bandwidth, set the bandwidth to as large as possible. Failure to do so may yield inaccurate readouts on the oscilloscope.

Figure 8: Left: BNC T-Connector and Right: BNC 50 Ohm resistor

Figure 9: BNC T-Connector on oscilloscope with coaxial transmission line connected on the left and BNC 50 Ohm resistor connected on the right, to terminate the line.

IDC Headers

Three IDC headers on the PulseBlaster device provide access to all of the output bits. The IDC headers are labeled Flag0..11_Out, Flag12..23_Out and Flag24..35_Out. On each IDC header, the top row of pins (pins 14-26) are grounds, and the bottom row of pins (pins 1-13) are signals. The IDC header pinout is shown in the figure below.

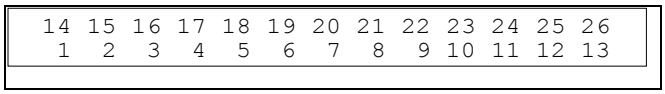

Figure 10: IDC header pinout

Each pin of an IDC header corresponds to a bit in the Output Pattern and Control field of an instruction (Each bit corresponds to a channel). The association between bits and pins are shown in the table below.

Pin Assignments			
Pin#	Flag0..11	Flag12..23	Flag24..35
1	Bit 0	Bit 12	Stopped
2	Bit 1	Bit 13	Reset
3	Bit 2	Bit 14	Running
4	Bit 3	Bit 15	Waiting
5	Bit 4	Bit 16	Unused
6	Bit 5	Bit 17	Unused
7	Bit 6	Bit 18	Unused
8	Bit 7	Bit 19	Unused
9	Bit 8	Bit 20	Unused
10	Bit 9	Bit 21	Unused
11	Bit 10	Bit 22	Unused
12	Bit 11	Bit 23	Unused
13	Unused	Unused	Unused
14-26	Ground	Ground	Ground

Table 1: IDC header pinout description. Every bit of the Output Pattern and Control field correspond to a channel. Pins 1 – 4 on IDC header “Flag24...35” are status bits. They allow the user or an external device to monitor the state of the PulseBlaster device.

The PulseBlaster device features four extra output signals, called "Status" signals, which can tell the user or an external device what state the PulseBlaster device is in. Pin assignments for these signals are shown in the table above. The status bits are defined as follows:

- **Stopped**
  - Driven high when the PulseBlaster device has encountered a STOP OpCode during program execution.
- **Reset**
  - Driven low when the PulseBlaster device is in a RESET state. The device must be reprogrammed before code execution can begin again.
- **Running**
  - Driven high when the PulseBlaster device is executing a program. This pin is low when the PulseBlaster enters either a reset or idle state.
- **Waiting**
  - Driven high when the PulseBlaster device has encountered a WAIT OpCode. Activating a trigger (either hardware or software) will resume operation.

HWTrig/Reset Header

This is an input connector for hardware triggering (HW_Trig) and hardware resetting (HW_Reset). If HW_Trig is activated, then the device will start running the program (the PulseBlaster device must be programmed first). If HW_Reset is activated, then the device will be stopped. Pins 1 and 2 are the reset and trigger signal pins, respectively, and pins 3 and 4 are grounds. The header pinout is shown in the figure below.

Figure 11: HWTrig/Reset Header pinout. Pin 1 and 2 are the HW_reset and HW_Trig, respectively. These are pulled high by 10k Ω resistors. They can be activated by grounding the signal pin (pin 1 or 2) with a ground pin (pin 3 or 4).

CAUTION: *The PulseBlaster requires 3.3 V input signals.* ***Applying voltages to the input pins that are greater than 3.3 V or less than 0V will damage the PulseBlasterESR-PRO-200-cPCI****.*

The external inputs are activated by a transition from logical high to logical low. The input is activated as long as the voltage remains at logical low (e.g., if HW_Reset is held low, the device will stay in a reset state, regardless if software or hardware triggers are used). To activate the external inputs, the signal pin must be shorted to ground, causing the transition. Both of these signals are pulled high to 3.3 V via 10k Ω resistors. Ground pins are provided next to the signal pins, so activating HW_Trig and HW_Reset is as easy as connecting the signal pin to a ground pin.

HW_Trigger (pin 2): When low voltage is detected (e.g., when shorting pins 2 and 4), one of two events will happen. If the hardware trigger is activated when the program is idle because of a:

- WAIT OpCode, then the program will continue to the next instruction.
- STOP OpCode or from HW_Reset activation, then the program will restart execution from the beginning of the program. If the STOP OpCode was used, a HW_Reset or software reset needs to be applied before the HW_Trigger.

Figure 12 shows an example of the HW_Trigger signal.

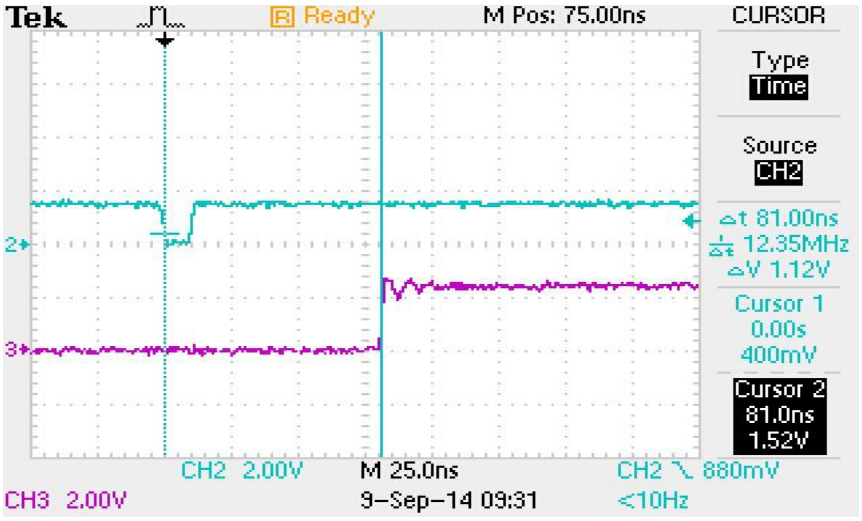

Figure 12: Demonstration of HW_Trigger signal. The blue shows the HW_Trigger signal, the pink shows one of the output flags. A latency of 80 ns is shown in this figure. Please refer to the Instruction Set Architecture section in Appendix I for more details on programming the duration of the WAIT latency. To trigger once, the trigger signal must begin at high voltage (between 2V and 3.3V), then must be pulled low (to ground) and stay low for at least 50 ns before returning to high voltage. The PulseBlaster will continue to trigger or reset for as long as the HW_Trigger or HW_Reset signals stay at ground. **Caution: applying voltages to the input pins that are greater than 3.3V or less than 0V will damage the board.**

HW_Reset (pin 1): When low voltage is detected (e.g., when shorting pins 1 and 3), program execution will be halted. The device resets itself back to the beginning of the program. Program execution can be resumed by either a software start command or by a hardware trigger.

Clock Oscillator Header

The PulseBlasterESR-cPCI comes with a crystal oscillator mounted at the designated “Clock” header to provide a timing signal for the board. If required, it is possible to remove the oscillator that comes standard, and instead drive the PulseBlasterESR-cPCI with an external clock signal. The oscillator module can be removed from the board, and an external signal can be input through the header pins. Do not attempt to drive a PulseBlaster board with an external clock while an oscillator module is also connected. The standard clock oscillator’s orientation should be noted - if the clock oscillator is reconnected, it must be inserted in the same orientation or board damage may occur. The external clock signal must be a TTL square wave, i.e. a digital signal of no more than 3.3 V. This is the absolute maximum allowable voltage, typically a voltage of 1.5-2 V is sufficient. Be aware that the TTL signal must be a positive-only signal, any negative voltage will damage the programmable-logic chip.

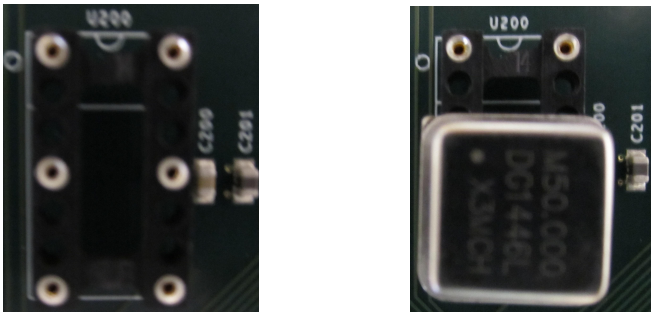

Figure 13: PulseBlaster ESR clock header. Both the bare header socket and the installed clock module are shown above. Please note the proper orientation of the 50 MHz clock.

Please take caution to provide a controlled signal at the correct frequency. The PulseBlasterESR-cPCI requires a 50 MHz signal. A reliable option for this purpose is the Oven Controlled Clock Oscillator available for purchase. This component will provide a precision low ripple signal for all PulseBlaster boards, and ensure that appropriate signal voltages are applied to the board. Information on this product can be found in the "Related Products and Accessories" section.

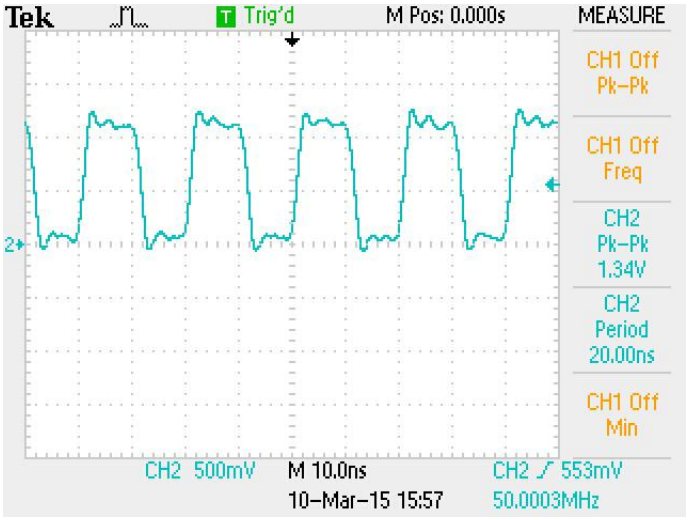

Figure 14: Example clock signal for SP19. Note that a small degree of voltage ripple is acceptable, so long as the voltage always remains above threshold for logical-high signals and below for logical-low signals.

NOTE: *The PulseBlasterESR-cPCI requires a 3.3V TTL input signal.* ***A signal that is more than 3.3V or less than 0V will damage the device****.*

VI. Related Products and Accessories

1. If you require additional interface options (custom cables for outputs or clocking, etc) or a custom product design, please inquire with SpinCore Technologies through our contact form, which is available at http://www.spincore.com/contact.shtml

2. If you require an Oven Controlled Clock Oscillator (with sub-ppm stability) or other custom features, please inquire with SpinCore Technologies through our contact form, which is available at http://www.spincore.com/contact.shtml

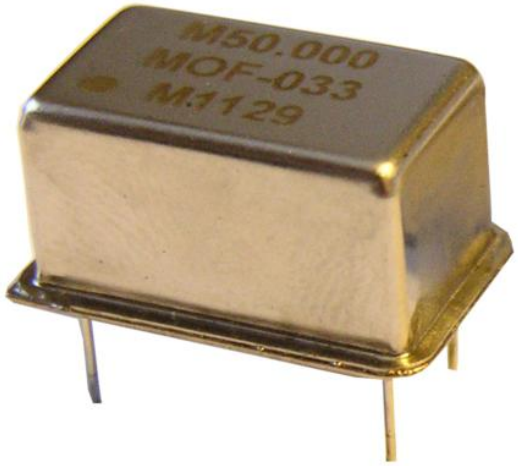

Figure 15: An Oven Controlled Clock Oscillator (or OCXO) with sub-ppm frequency stability is available for the PulseBlaster upon request.

3. PulseBlasterESR, PulseBlasterESR-PRO, and PulseBlasterESR-PRO-II – Alternate versions of the PulseBlaster that are capable of Higher Clock Frequencies (currently up to 500 MHz). For more information, please visit the individual Product URLs of the aforementioned products at http://www.spincore.com/products.shtml

4. PulseBlasterDDS – Built upon the PulseBlaster, the PulseBlasterDDS features programmable TTL outputs and RF Pulse Generation. For more information, please visit http://www.spincore.com/products/PulseBlasterDDS-300/

5. PulseBlasterUSB – The portable, stand-alone version of the PulseBlaster. For more information, please visit http://www.spincore.com/products/PulseBlasterUSB

VI. Contact Information

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

Telephone: **+1-352-271-7383**
Fax: **+1-352-371-8679**

Website: **http://www.spincore.com.**
Contact Email: **http://www.spincore.com/contact.shtml.**

VII. Document Information

Detailed revision history is available by contacting SpinCore Technologies, Inc.