PulseBlasterESR-PRO™

(PCI Board SP18A)
(USB Enclosure System SP47, SP51)
(Rackmount Front Panel SP51)
(PCIe Board SP49, SP56)

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.

SpinCore Technologies, Inc. makes every effort to verify the correct operation of the equipment. This equipment version is not intended for use in a system in which the failure of a SpinCore device will threaten the safety of equipment or person(s).

Table of Contents

I. Introduction

Product Overview

The PulseBlasterESR-PRO™ is a high-speed, intelligent pulse/pattern/delay generator designed for outputting precisely timed TTL patterns. The intelligence of the PulseBlasterESR-PRO comes from an embedded microprogrammed controller core nicknamed the PulseBlaster™. The controller is able to execute instructions that allow it to control program flow much like a general purpose microcontroller. The speed of the PulseBlasterESR-PRO comes from a maximum available clock of 500 MHz.

The PulseBlasterESR-PRO's microprogrammed controller core is different from the general-purpose microcontroller in that it contains a set of highly optimized instructions developed specifically for timing and control applications. A unique and distinguishing feature of the PulseBlasterESR-PRO processor is that the execution time for instructions is user programmable. This feature makes the PulseBlasterESR-PRO processor capable of executing complex timing patterns at greatly varying update rates, ranging from nanoseconds to months, with a constant setting accuracy of just one clock period.

Board Architecture

Block Diagram

Figure 1 presents the general architecture of the PulseBlasterESR-PRO system. The major building blocks are the SRAM memory, the PulseBlaster core, the integrated bus controller (IBC), the counter, and the output buffers. The entire logic design, including the SRAM memory and output buffers, is contained on a single silicon chip, making it a System-on-a-Chip design. User control to the system is provided through the IBC over the peripheral component interconnect (PCI), peripheral component interconnect express (PCIe) bus or universal serial bus (USB).

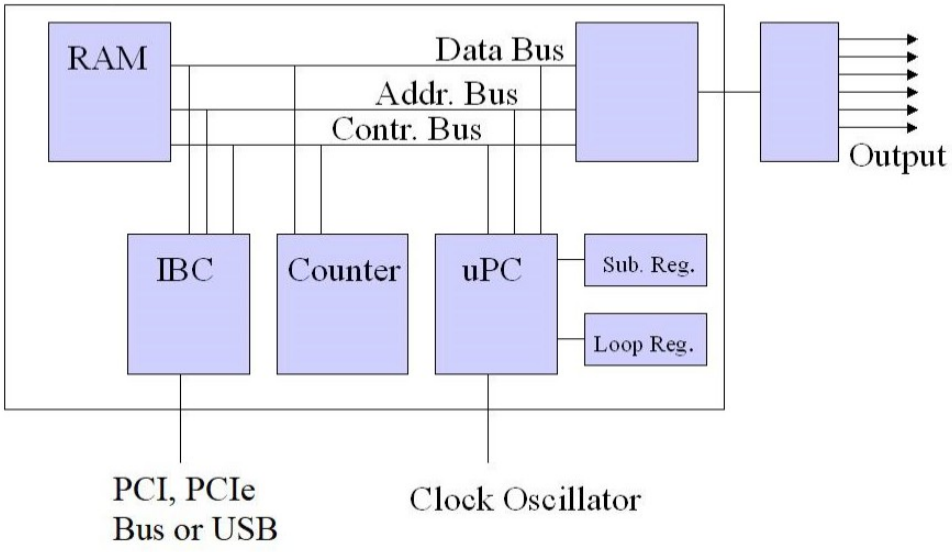


Figure 1: PulseBlasterESR-PRO Board Architecture. The clock oscillator signal is derived from an on-chip PLL circuit typically using a 50 MHz on-board reference clock.

Output Signals

The PulseBlasterESR-PRO allows for 21 digital output signal lines. On the PCI and PCIe boards, all 21 signal lines are routed to two sets of 26-pin IDC on-board connectors. In addition, the first four bits are routed to four bracket mounted BNC connectors and bits 4 to 11 are routed to SMA connectors. On the USB system, all 21 signal lines are routed to 21 BNC connectors. The output signals are impedance matched to 50 ohm.

The 21 individually controlled digital output bits comply with the 3.3V TTL-levels' standard, and are capable of delivering ±25 mA per bit/channel. Keep in mind that this is sufficient to provide a signal to a 132 ohm load, but if more current is necessary beyond this, the individual bits/channels can be driven in parallel.

Timing Characteristics

The PulseBlaster core's timing controller accepts an external (on-board) crystal oscillator of 50 MHz. The input frequency is internally multiplied. The PulseBlasterESR-PRO is available with up to 500 MHz internal clock frequency. The innovative architecture of the timing controller allows the processing of either simple timing instructions (with delays of up to 2^{32} clock cycles or 8.59 s at 500 MHz), or double-length timing instructions (up to 2^{52} clock cycles long – over 100 days at 500 MHz!). Regardless of the type of timing instruction, the timing resolution remains constant for any delay – just one clock period (e.g., 2 ns at 500 MHz).

The PulseBlaster core-timing controller has a very short minimum instruction time – only six clock periods[1]. This translates to a 12 ns machine instruction time at 500 MHz. The PulseBlasterESR-PRO is also capable of generating pulses on all outputs of lengths down to one clock cycle. For more information on this feature, please see the Short Pulse Feature section in Appendix I.

Instruction Set

The PulseBlaster core features a set of instructions for creating highly flexible pulse program flow control. The micro-programmed controller allows for programs to include branches, subroutines, and loops at up to 8 nested levels – all this to assist the user in creating dense pulse programs that cycle through repetitious events, especially useful in numerous multidimensional spectroscopy and imaging applications.

External Triggering

The PulseBlasterESR-PRO can be triggered and/or reset externally via dedicated hardware lines. These lines combine the convenience of triggering (e.g., in cardiac gating) with the safety of the "stop/reset" line (firmware-dependent).

Summary

The PulseBlasterESR-PRO is a versatile, high-performance pulse/pattern TTL signal generator operating at speeds of up to 500 MHz (see Firmware Designs) and capable of generating pulses ranging from 2 ns to 104 days per instruction at intervals ranging from 12 ns to 8.59 s per instruction (using a 500 MHz clock signal). It can accommodate pulse programs with highly flexible control commands of up to 32768 instruction words. Its high-current output logic bits are individually controlled with an unterminated output voltage of 3.3 V[2].

[1] NOTE: The PBESR-PRO-500-PCI (design 17-11) and PBESR-PRO-400-32k-PCIe (designs 34-10, 34-15, and 31-3) require a minimum instruction time of at least 7 clock-cycles.
[2] NOTE: For PBESR-PRO-PCIe, the unterminated output voltage is 3 V.

Specifications

TTL Specifications

- 21 individually controlled digital output lines (LVTTL levels, 3.3 V logical “one” unterminated)[3]
- 4 bracket mounted BNC connectors and 8 SMA connectors, impedance matched to 50 ohm, for PCI and PCIe board
- 24 BNC connectors, impedance matched to 50 ohm, for rackmount system and rackmount front panel, 21 of which are individually controlled output channels
- Variable pulses/delays for every TTL line
- 25 mA output current per TTL line

Pulse Parameters

- External triggering and reset – 3.3V LVTTL levels
- 500 MHz design
  - 2 ns shortest pulse
  - 12 ns shortest interval
  - 104 days longest pulse/interval (using the long delay instruction)
  - 2 ns pulse/interval resolution
  - 4096 instructions
- 400 MHz 32k memory word design (PCIe only, see Firmware Designs)
  - 2.5 ns shortest pulse
  - 17.5 ns shortest interval
  - 130 days longest pulse/interval (using the long delay instruction)
  - 2.5 ns pulse/interval resolution
  - 32768 instructions
- 400 MHz 4k memory word design (PCIe only, see Firmware Designs)
  - 2.5 ns shortest pulse
  - 15 ns shortest interval
  - 130 days longest pulse/interval (using the long delay instruction)
  - 2.5 ns pulse/interval resolution
  - 4096 instructions

Pulse Program Control Flow (Common)

- Loops, nested 8 levels deep
- 20 bit loop counters (max. 1,048,576 repetitions)
- Subroutines, nested 8 levels deep
- Wait for trigger – 8 clock cycle latency (16ns at 500 MHz), adjustable to 0.89 seconds in duration
- 15 MHz max. re-triggering frequency

[3] NOTE: For PBESR-PRO-PCIe, the unterminated output voltage is 3 V.

II. Installation

Installing the PulseBlasterESR-PRO

Whenever installing or uninstalling the PulseBlasterESR-PRO, always have it disconnected from the computer initially. Uninstall any previous version of SpinAPI.

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 PulseBlasterESR-PRO and most of SpinCore's other products. 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.

2. Shut down the computer, unplug the power cord.

3. Connect the product to the computer.

   a) For the PCI: Insert the PulseBlasterESR-PRO-PCI card into an available PCI slot and fasten the PC bracket securely with a screw.
   b) For the PCIe: Insert the PulseBlasterESR-PRO-PCIe card into an available PCIe slot and fasten the PC bracket securely with a screw.
   c) For the USB system: Plug one end of the USB cable into the PulseBlasterESR-PRO-USB and the other end into the host computer.
      - Next, power the rackmount using a C13 cable for the PBESR-PRO-USB-RM or a 4 pin molex (SP51 powers via +5 V DC) for the PBESR-PRO-USB-RM-FP.

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

5. The simplest way to test whether the device has been installed properly and can be controlled as intended is to run a simple test program. These example files can be found in the SpinAPI package.

6. To open the SpinAPI package on a Windows 10 PC, simply click the Window Start icon, and scroll down to find and open the "SpinCore" folder. Example .exe files and their C source code can be found in the folder /SpinAPI/examples. From there, you may select the “PulseBlasterESR-PRO” folder and run all .exe programs to test your PulseBlaster.

III. Programming the PulseBlasterESR-PRO

SpinCore Technologies Inc. is dedicated to providing an easy and efficient method of programming your board. Various control methods available are detailed below, making PulseBlaster products flexible for any number of applications.

Special consideration of the ESR-PRO Short Pulse feature must be taken when programming or operating this board. For signals of instruction time greater than five clock cycles, the Short Pulse feature must be disabled by setting output bits 21-23 to HIGH or LOW at all times (firmware-dependent). For more information, see Short Pulse Feature section in Appendix I.

The PulseBlaster Interpreter

The PulseBlasterESR-PRO can be programmed using *PulseBlaster Interpreter*, which is a free programming utility provided by SpinCore for writing pulse programs. This easy-to-use editor allows you to create, edit, save, and run your pulse sequence. Figure 2, below, shows the PulseBlaster Interpreter being used with an example program.

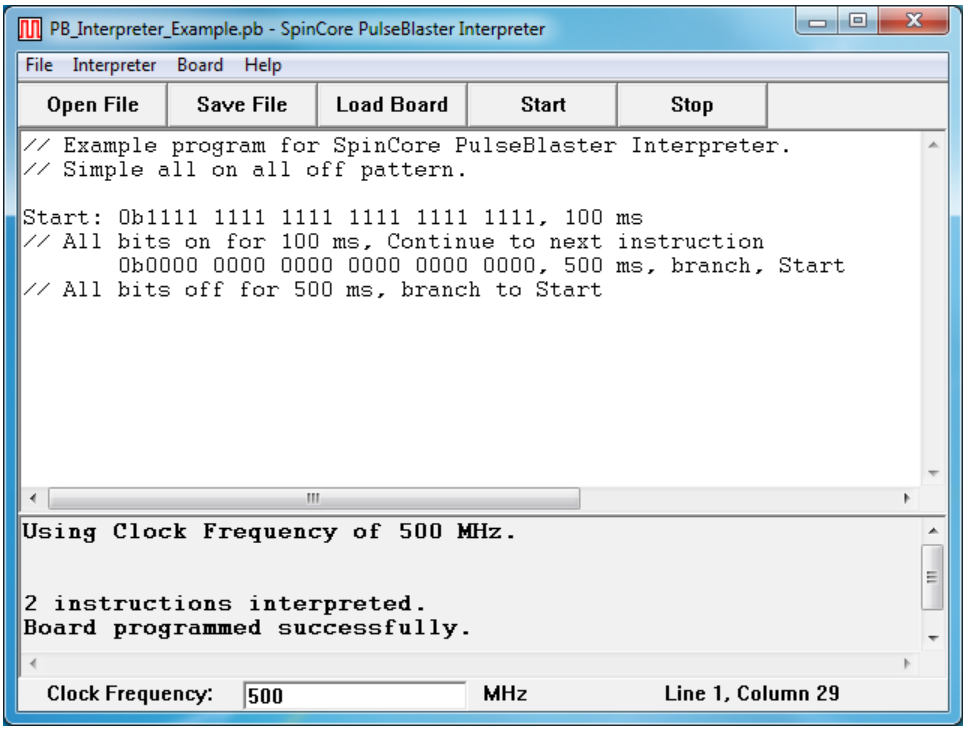


Figure 2: Graphical Interface of PulseBlaster Interpreter. The example shown creates a pulse that toggles all TTL bits on for 100 ms, then off for 500 ms, and repeats.

The PulseBlaster Interpreter is available as part of the SpinCore driver suite, and will be automatically installed during the setup process (setup process is described in Section II. Installation). For convenience, a shortcut to the PulseBlaster Interpreter will be added to your desktop. For more information on programming using the PulseBlaster Interpreter, see the manual located at http://www.spincore.com/support/SPBI/Doc/.

LabVIEW Extensions

The SpinCore PulseBlaster LabVIEW Extensions (PBLV) provide the ability to program and control the functionality of PulseBlaster boards using the simple National Instruments (NI) LabVIEW graphical programming interface. The package contains basic subVIs that can be used to include PulseBlaster 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.

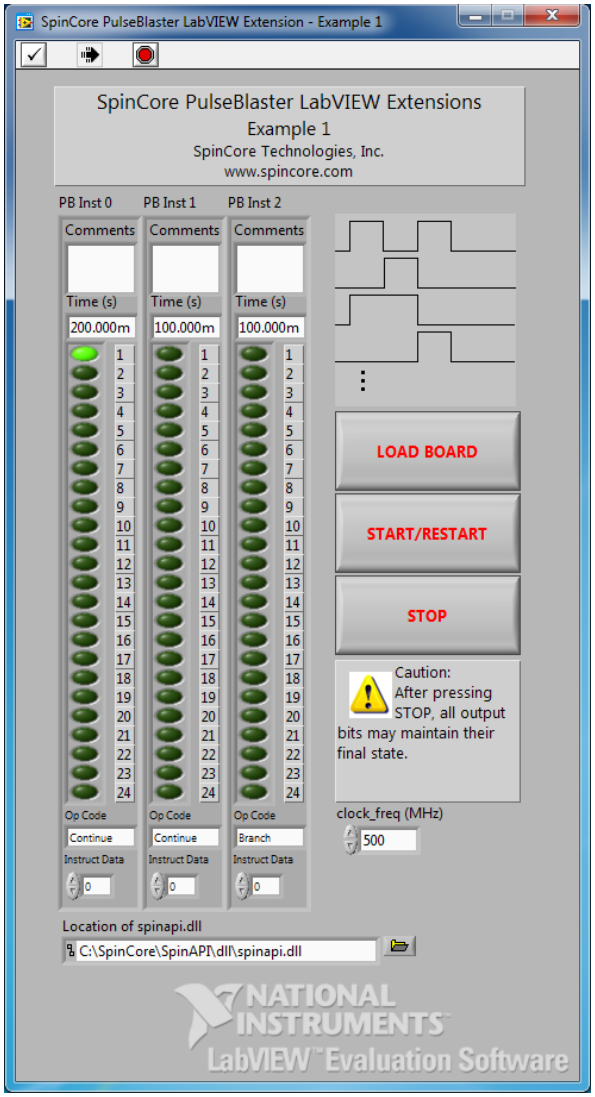


Figure 3: Example of PulseBlaster LabVIEW Extensions User Interface.

There are two versions of the LabVIEW extensions available free of charge on our website. The first is for those who do not have LabVIEW or who are not familiar with LabVIEW programming. This option is a stand-alone GUI (see Figure 3 above) that comes in executable form and utilizes the LabVIEW runtime environment. The second is for those who have LabVIEW and would like to make a custom interface for the PulseBlasterESR-PRO. For more information and downloads please visit:
http://www.spincore.com/support/PBLV/

C/C++ Programming

The most dynamic and flexible way to program the PulseBlasterESR-PRO board is with C/C++ using the SpinAPI package. The GUI based approaches to programming the board are designed for simplicity so they can be used by someone with no programming experience.

While GUI's are easier to use, coding in C/C++ allows you to better utilize the various features of the board and, in some cases, it may be easier to copy and paste lines of code than to make 100 instructions on a GUI. The instructions to compile on Windows can be found at http://www.spincore.com/support/spinapi/Windows_Help.shtml. After configuring the compiler, changing one of our example programs and recompiling the executable file for use with your PulseBlasterESR-PRO board is as easy as clicking “Rebuild All” (see Figure 4 below).

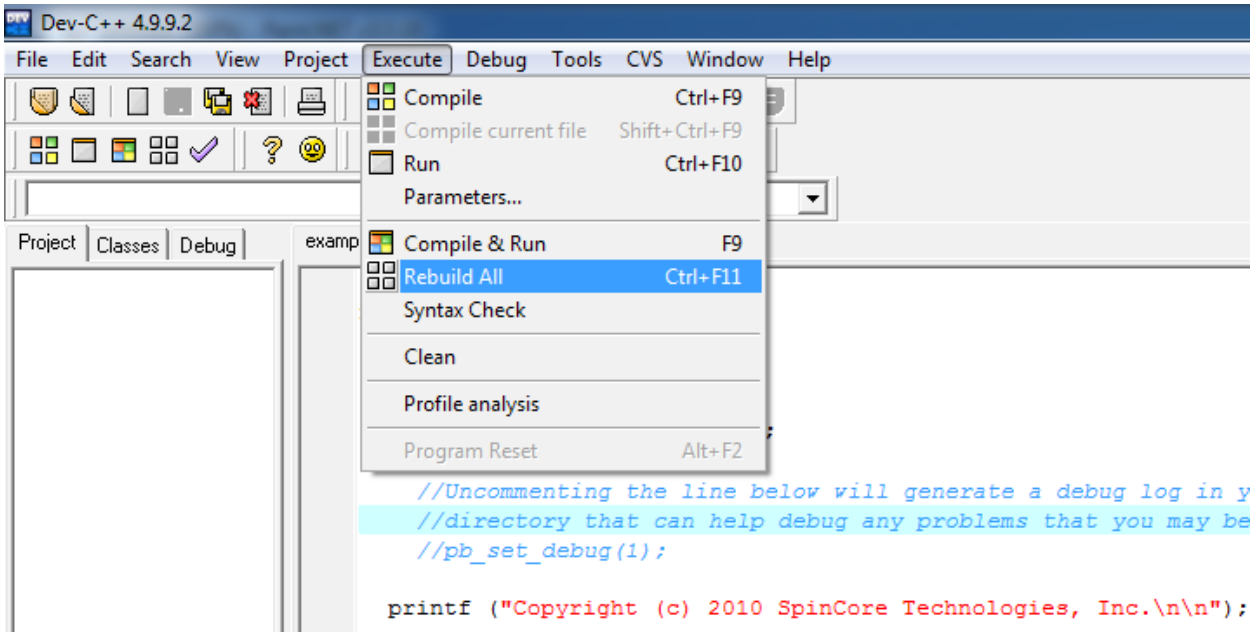


Figure 4: Compiling a C program to run the PulseBlasterESR-PRO board is easy!

Making changes to an example program requires understanding of only a few lines of code. The following C code example generates a 50% duty cycle square wave with a 400.0 ms period.

```
1:  pb_init(); /*Initialize communication with the board*/
2:  pb_core_clock (CLOCK); /*Set the internal clock frequency value – this
3:                          will be either 250, 300, 400, or 500 MHz
4:                          depending on your product */
5:
6:   /*Start programming the Pulse Program*/
7:  pb_start_programming (PULSE_PROGRAM);
8:  start=pb_inst(ON|0x01, CONTINUE, 0, 200.0*ms); /*Bit 0 on, 200ms*/
9:        pb_inst(0x00, BRANCH, start, 200.0*ms);/*All bits off, 200ms*/
10:  pb_stop_programming();
11:
12: pb_start(); /*Start the board executing*/
13: pb_close(); /*Close the communication with the board*/
```

A breakdown of the previous C code segment is as follows:

- *Line 1:* Initialize communication with the selected board. This must be called before any other functions that communicate with the board.
- *Line 2:* Set the internal block clock frequency (in MHz). This must be called to ensure proper timings in the pulse program.
- *Lines 7-10:* Programs the board's pulse program memory.
  - *Line 7*: pb_start_programming (PULSE_PROGRAM) must be called before using the pb_inst(..) function.
  - *Line 8*, instruction 1: Turn on bit 0 for 200.0 ms then continue to the next instruction. The address of this instruction is stored in the “start” variable.
    - *Note: If the output is high for more than 5 clock cycles, it is necessary to turn off the Short Pulse feature by settings bits 21-23 of the flag bits to “111.” This can be easily accomplished by using the C-macro “ON” defined in “spinapi.h.”*
  - *Line 9*, instruction 2: All bits off for 200.0 ms, then branch to “start.”
  - *Line 10:* pb_stop_programming() must be called before calling any other SpinAPI functions.
- *Line 12:* Start the board executing the Pulse Program.
- *Line 13:* Close communication with the board (Pulse Program execution will continue).

Using C Functions to Program the PulseBlasterESR-PRO

A series of functions have been written to control the board and facilitate the construction of pulse program instructions. It should be noted that the pb_inst C function accepts any delay value greater than five clock cycles. Since the clock period is 2.5 ns for a 400 MHz clock, values which are not integer multiples of the clock will be rounded to the closest integer multiple.

In order to use these functions, the DLL (spinapi.dll), the library file (libspinapi.a for MinGW, spinapilibgcc for Borland, and spinapi.lib for MSVC), and the header file (spinapi.h), must be in the working directory of your C compiler[4].

```
int pb_init();
```

Initializes the PulseBlasterESR-PRO board. Needs to be called before calling any functions using the device. It returns a 0 on success or a negative number on an error.

```
int pb_close();
```

Releases the PulseBlasterESR-PRO board. Needs to be called as last command in pulse program. It returns a 0 on success or a negative number on an error.

```
void pb_core_clock(double clock_freq);
```

Used to set the clock frequency of the board. The variable **clock_frequency** is specified in MHz when no units are entered. Valid units are MHz, kHz, and Hz.

```
int pb_start_programming(int device);
```

Used to initialize the system to receive programming information. It accepts a parameter referencing the target for the instructions. The only valid value for **device** is PULSE_PROGRAM. It returns a 0 on success or a negative number on an error.

[4] These functions and library files have been generated and tested with MinGW, Borland 5.5, MS Visual Studio 2003 compilers.

```
int pb_inst(int flags, int inst, int inst_data, double length);
```

Used to send one instruction of the pulse program. Should only be called after pb_start_programming(PULSE_PROGRAM) has been called. It returns a negative number on an error, or the instruction number upon success. If the function returns –99, an invalid parameter was passed to the function. Instructions are numbered starting at 0.

int flags – determines state of each TTL output bit. Valid values are 0x000000 to 0xFFFFFF. For example, 0x000010 would correspond to bit 4 being on, and all other bits being off.

int inst – determines which type of instruction is to be executed. Please see Table 8 for details.

int inst_data – data to be used with the previous inst field. Please see Table 8 for details.

double length – duration of this pulse program instruction, specified in nanoseconds (ns), microseconds (us) or milliseconds (ms).

```
The largest value for the delay field of the pb_inst is 8589 ms (using a 500 MHz clock).
For longer delays, use the LONG_DELAY instruction (see Table 8).  The maximum value for
the data field of the LONG_DELAY is 1048576.  Even longer delays can be achieved using
the LONG_DELAY instruction inside of a loop.
```

```
int pb_stop_programming();
```

Used to tell that programming the board is complete. Board execution cannot start until this command is received. It returns a 0 on success or a negative number on an error.

```
int pb_start();
```

Once board has been programmed, this instruction will start execution of pulse program. It returns a 0 on success or a negative number on an error.

```
int pb_stop();
```

Stop the Pulse Program execution. TTL outputs will either remain in their last state or return to zero, depending on the firmware version of the board. It returns a 0 on success or a negative number on an error.

There are precompiled C example programs available with the SpinAPI package in the PulseBlasterESR-PRO directory. In addition, the PulseBlaster24 examples will work for PulseBlasterESR-PRO products.

Example Use of C Functions

```
#include <stdio.h>
#include <stdlib.h>

#define PBESRPRO
#include "spinapi.h"

#define CLOCK 400.0    // PulseBlaster core clock rate

int main (int argc, char **argv)
{
    int start;

    printf ("Copyright (c) 2010 SpinCore Technologies, Inc.\n\n");
    printf("Using SpinAPI library version %s\n", pb_get_version());

    if (pb_init () != 0) {
        printf ("Error initializing board: %s\n", pb_get_error());
        system("pause");
    }

    // Tell driver what clock frequency the board uses
    pb_core_clock(CLOCK);

    // Prepare the board to receive pulse program instructions
    pb_start_programming(PULSE_PROGRAM);

    // Instruction 0 - Continue to instruction 1 in 20ns. The lower 4 bits
    // (all BNC connectors) will be driving high. For PBESR-PRO boards,
    // or-ing THREE_PERIOD with the flags causes a 3 period short
    // pulse to be used.
    start = pb_inst(THREE_PERIOD | 0xF, CONTINUE, 0, 20.0 * ns);

    // Instruction 1 - Continue to instruction 2 in 40ns
    // The BNC1-3 will be driving high the entire 40ns.
    pb_inst(ON | 0xE, CONTINUE, 0, 40.0 * ns);

    // Instruction 2 - Branch to "start" (Instruction 0) in 40ns
    // Outputs are off
    pb_inst(0, BRANCH, start, 40.0 * ns);

    pb_stop_programming();    // Finished sending instructions

    pb_reset();
    pb_start();    // Trigger the pulse program

    // End communication with PulseBlasterESR-PRO board. The pulse program
    // will continue to run even after this is called.
    pb_close();

    return 0;
}
```

IV. Connecting to the PulseBlasterESR-PRO

The connectors for the PulseBlasterESR-PRO (SP18A, SP49, SP56) boards and (SP47, SP51) USB rackmounts are explained below in their respective sections.

Connector Information for PulseBlasterESR-PRO Boards

On the PCI (SP18A) and PCIe boards (SP49, SP56), there are four main connector banks: the BNC headers, the IDC headers, the SMA headers, and the Trigger/Reset header. Please refer to the sections below for detailed connector information.

BNC Headers

The four BNC headers provide access to the least significant four bits of the flag word. Bit 3 is connected to the output farthest from the PCI or PCIe connector, and Bit 0 is connected to the connector closest to the PCI or PCIe connector. Please note that the BNC connectors have the same configuration for all board models.

If using a high input impedance oscilloscope to monitor the PulseBlasterESR-PRO's output via the BNC connectors, place a resistor that matches the characteristic impedance of the transmission line in parallel with the coaxial transmission line at the oscilloscope input (e.g., a 50 Ω resistor with a 50 Ω transmission line, see Figures 5, below, and 6 on the next page). 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 5: Left: BNC T-Adapter and Right: BNC 50 Ohm resistor.

Figure 6: BNC T-Adapter on the oscilloscope with coaxial transmission line connected on the left and BNC 50 Ohm resistor connected on the right, to terminate the line.

IDC Headers for SP18A, SP49, and SP56

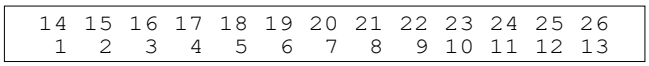


Figure 7: IDC header Pin-Out.

There are three IDC headers on SP18A, SP49, and SP56 boards, which provide access to all digital 21 outputs as well as the status bits. These are labeled on the SP18A board as **Flag0..11_Out**, **Flag12..23_Out** and **Flag24..35_Out**. On the SP49 and SP56 boards, these are labeled as **Flag0..11**, **Flag12..23**, **Flag24..35**. On each IDC header, the top row of pins (14-26) are grounds, and the signals are carried on pins 1-13. The recommended mating connector for IDC header can be obtained through Digi-Key Part Number: CKR26G-ND.

Each pin on an IDC header corresponds to a bit in the flag field of an instruction. The association between bits and pins for the PCI and PCIe boards can be found in Table 1 and 2, respectively.

Alternatively, the IDC headers can be connected to IDC-MMCX adapter boards (Figures 20 and 21) which allow the use of MMCX cables. This enables the individual bits of the PulseBlasterESR-PRO to be more easily accessed. Pin 1 on the MMCX adapters can be identified with a square pin.

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 connector pin outs for SP18A boards.

Pin Assignments			
Pin#	Flag0..11	Flag12..23	Flag24..35
1	Bit 0	Bit 12	Reset
2	Bit 1	Bit 13	Running
3	Bit 2	Bit 14	Waiting
4	Bit 3	Bit 15	Unused
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 2: IDC connector pin outs for SP49 and SP56 boards.

The status pins are set or cleared based on the state of the PulseBlasterESR-PRO. For the SP18A boards, the status pins are located on IDC header Flag24...35, from pin 1 to pin 4. The status pins of the SP49 and SP56 boards are located on IDC header Flag24...35, from pin 1 to pin 3 and does not have a pin for the Stopped status. Please refer to the Status Pins Description for additional information on the status pins.

Connector Locations for the SP18A board

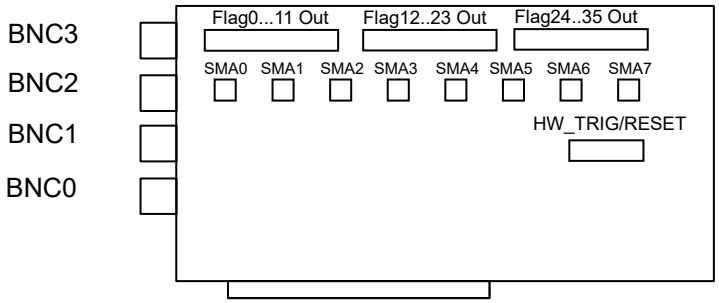


Figure 8: Connector Locations (SP18A).

SMA Headers for SP18A

The eight SMA headers provide access to the flag bits 4 through 11. SMA0, the header closest to the BNC headers, corresponds to flag bit 4. SMA7, the header furthest from the BNC header, corresponds to flag bit 11.

HW_Trig/Reset Header for SP18A

9	7	5	3	1
10	8	6	4	2

Figure 9: HW_Trig/Reset Header Pin-Out (SP18A).

This is an input connector for hardware triggering (HW_Trigger) and resetting (HW_Reset). Pins 8 and 10 are the HW_Reset and HW_Trigger pins, respectively. Pins 2, 4, and 6 are reserved. Pins 1, 3, 5, 7, and 9 are all grounds. The part number for this header is 5103308-1. Please refer to the Hardware Reset and Hardware Trigger sections for more information on these hardware pins.

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

Connector Locations for the SP49 and SP56 boards

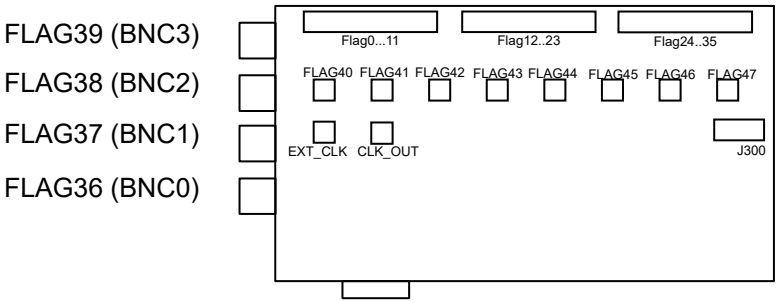


Figure 10: Connector Locations (SP49 and SP56).

SMA Headers for SP49 and SP56

The eight SMA headers provide access to the flag bits 4 through 11. FLAG40, the header closest to the BNC headers, corresponds to flag bit 4. FLAG47, the header furthest from the BNC header, corresponds to flag bit 11.

J300 Header for SP49 and SP56

2	4	6	8	10
1	3	5	7	9

Figure 11: J300 Header Pin-Out (SP49 and SP56).

Pin Assignments			
Pin#		Pin#	
1	GND	2	HW_Trigger_H
3	GND	4	HW_Trigger_H
5	GND	6	HW_Reset_H
7	GND	8	HW_Reset
9	GND	10	HW_Trigger

Table 3: SP49 and SP56 J300 Pin Assignments.

This is an input connector for hardware triggering (HW_Trigger, HW_Trigger_H) and resetting (HW_Reset, HW_Reset_H). The SP49 and SP56 boards come with three hardware trigger pins and two hardware reset pins. The part number for this header is 5103308-1. Please refer to the Hardware Reset and Hardware Trigger sections for more information on these hardware pins.

SMA Connector CLK_OUT

This SMA connector outputs the reference clock as a 3 V positive-only signal. Note that the PulseBlasterESR-PRO PCIe board uses 50 MHz as the reference clock frequency and that clock is internally multiplied to provide that actual PulseBlaster Core frequency[5]. The output resembles a square wave if properly terminated. This signal can be measured with an oscilloscope using either a high impedance probe at the SMA connector or a 50 ohm coaxial line that is terminated.

SMA Connector EXT_CLK

This SMA connector can be used to input an external clock signal. Extreme care should be exercised, and certain conditions have to be met prior to using this connector. First, before attaching any external clock source, the internal clock oscillator must be removed from its socket. The internal clock oscillator's orientation should be noted - if the internal clock is reconnected, it must be inserted in the same orientation or board damage may occur. Second, the external clock signal must be 3.3 V TTL, i.e., a positive-only voltage - any negative voltage at the EXT_CLK connector will damage the programmable-logic processor chip. Third, the EXT_CLK connector for certain boards is not terminated on the printed circuit board, and a 50 ohm terminating resistor should be used externally via a T connector placed directly at the SMA EXT_CLK connector. If the EXT_CLK is terminated, there will be a 50 ohm resistor on pad R103 for SP49 and SP56 PCIe boards. Soldering a 50 ohm resistor to this pad, if not already populated, is an alternative to using a T connector with a 50 ohm resistor.

Connector Information for 2U BNC Rackmount Enclosure and Rackmount Front Panel

Connectors for 2U BNC Rackmount Enclosure and Rackmount Front Panel

The front panel of the rackmount provides 21 individually controlled output channels through BNC connectors. The BNC connectors are arranged in a format which is 3 rows of 8 bits. The front panel also contains the USB connector, the Trig/Res/Stat DB9 connector, and the ON/OFF toggle switch.

The ON/OFF toggle switch toggles the internal power supply of the PulseBlasterESR-PRO-USB-RM. On the reverse side of the rackmount enclosure is the AC Input.

[5] Custom firmware may use a different speed reference clock and may not be internally multiplied.

DB9 Connector (Trig/Res/Stat) for 2U BNC Rackmount Enclosure and Rackmount Front Panel

The Trig/Res/Stat DB9 connector information is shown in Figure 12 and Table 4, below. The Hardware Trigger and Hardware Reset are both low-true, so each of these pins would need to be shorted to ground to cause a trigger or reset, respectively. Please refer to the **Status and Hardware Pins** section for additional information about each pins functionality.

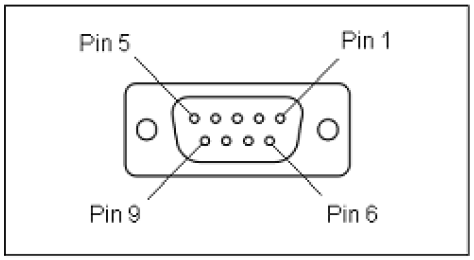


Figure 12: Trig/Res/Stat Male/Female DB9 connector drawing. This pin numbering is for both male and female DB9 connectors. When making a custom cable, starting with the mating DB9 connector may be helpful in recognizing where the pins are on the mating connector. This image is drawn with a view looking at the front of the front panel.

Pin Number	Function
1	Hardware Trigger
2	Hardware Reset
3	WAITING
4	RUNNING
5	RESET
6	Ground
7	Ground
8	Ground
9	STOPPED

Table 4: Trig/Res/Stat Male/Female DB9 connector pin functions. The DB9 connector pin postions are located in Figure 12. When making a custom cable, starting with the mating DB9 connector may be helpful in recognizing where the pins are on the mating connector.

Power Connector for the Rackmount Front Panel

The PulseBlasterESR-PRO Rackmount Front Panel option has a 4-pin Molex-style connector for supplying power. The pin and signal arrangements for this connector is as follows:

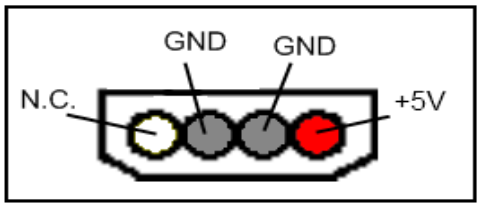


Figure 13: 4 Pin input connector (TE Connectivity AMP part 174804-1).

This power connector mates with standard PC power supply connector or TE Connectivity AMP part 1-480424-0.

Note that it is suggested that a PC power supply with the connector specified is used. This is a standard connector type common on most PCs that will satisfy the power requirements and prevent any damage (such as accidentally reversing polarity). The PBESR-PRO-USB-RM-FP does not contain overvoltage protection, reverse polarity protection, or undercurrent detection. If a PC power supply is not easily accessible it is suggested that a fixed-voltage +5 V power supply with a 3 A maximum current be used.

Status and Hardware Pins

Status Pins Description

Stopped – Driven high when the PulseBlasterESR-PRO has encountered a STOP OpCode during program execution and has entered a stopped state.

Reset – Driven high when the PulseBlasterESR-PRO is in a RESET state.

Running – Driven high when the PulseBlasterESR-PRO is executing a program. The pin is driven low when the PulseBlasterESR-PRO enters either a reset or idle state.

Waiting – Driven high when the PulseBlasterESR-PRO has encountered a WAIT OpCode, and is waiting for the next trigger (either hardware or software) to resume operation. Note that the Running bit will also be high during a WAIT state.

Note that it is also possible to read the status bits via software by using the *pb_read_status()* function. Please see http://www.spincore.com/CD/spinapi/spinapi_reference/ for details.

Hardware Reset

The SP18A, SP47, and SP51 have the HW_Reset hardware reset pin. The SP49 and SP56 PCIe boards have the HW_Reset and HW_Reset_H hardware reset pins. **HW_Reset** is pulled to high voltage (3.3V) on the board and can be activated by a low voltage pulse (or shorting to GND). **HW_Reset_H** is pulled to low voltage on the board (ground) and can be activated by a high voltage pulse (to 3.3V). When the signal is activated during the execution of a program, the controller resets itself back to the beginning of the program. Program execution can be started from the beginning by either a software start command (pb_start()) or by a hardware trigger.

NOTE: *The PulseBlaster requires a 3.3V input signal for HW_Reset.* ***Applying voltages to the input pins that are greater than 3.3V or less than 0V will damage the PulseBlasterESR-PRO****.*

Hardware Trigger

The SP18A, SP47, and SP51 have the HW_Trigger hardware trigger pin. The SP49 and SP56 boards come with three hardware trigger pins: one HW_Trigger and two HW_Trigger_H. **HW_Trigger** is pulled to high voltage (3.3V) on the board and can be triggered by a low pulse (or shorting to GND). The two **HW_Trigger_H** pins are pulled to low voltage (ground) on the board and can be triggered with a high voltage pulse (to 3.3V). When the falling edge is detected (or rising edge on HW_Trigger_H), and the program is idle, code execution is triggered. If the program is idle due to a WAIT instruction, the HW_Trigger will cause the program to continue to the next instruction. If the program is idle due to a STOP instruction or a HW_Reset signal, the HW_Trigger will start execution from the beginning of the program. If the STOP instruction was used, a HW_Reset or software reset (pb_reset() or pb_stop()) needs to be applied prior to the HW_Trigger.

NOTE: *The PulseBlaster requires a 3.3V input signal for HW_Trigger.* ***Applying voltages to the input pins that are greater than 3.3V or less than 0V will damage the PulseBlasterESR-PRO****.*

Figure 14, on the next page, shows an example of the HW_Trigger signal with a latency of 80 ns. Please refer to Instruction Set Architecture in Appendix I for more details on programming the duration of the WAIT latency. To trigger once, the trigger signal must begin at logical-high voltage (between 2V and 3.3V), then must be pulled low (to ground) and stay low for at least 10 ns before returning to logical-high voltage. The PulseBlaster will continue to trigger or reset for as long as the HW_Trigger or HW_Reset signals stay at ground. If using a long TTL cable, make sure it is terminated and a buffer is used. If necessary, use an inverter or program the triggering device to match the high-low-high HW_Trigger signal. The input impedance of the HW_Trigger pin is 10 kOhms.

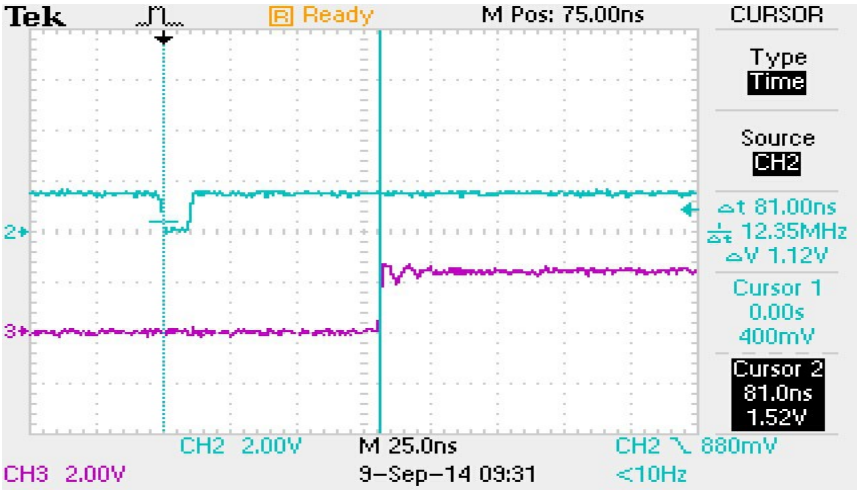
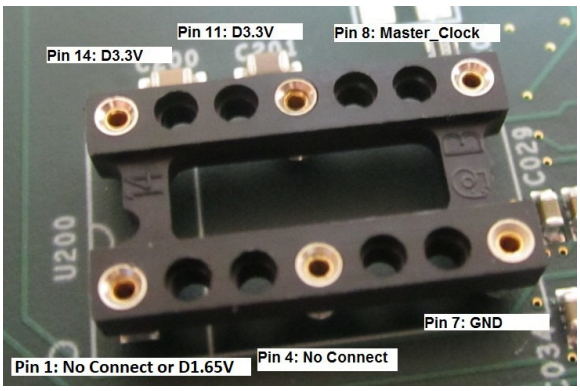


Figure 14: Demonstration of HW_Trigger high-low-high signal. The blue shows the HW_Trigger signal, the pink shows one of the output flags.
Caution: applying voltages to the input pins that are greater than 3.3V or less than 0V will damage the PulseBlasterESR-PRO.

Clock Oscillator Header

The PulseBlasterESR-PRO comes with a crystal oscillator mounted on the oscillator socket 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-PRO 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.



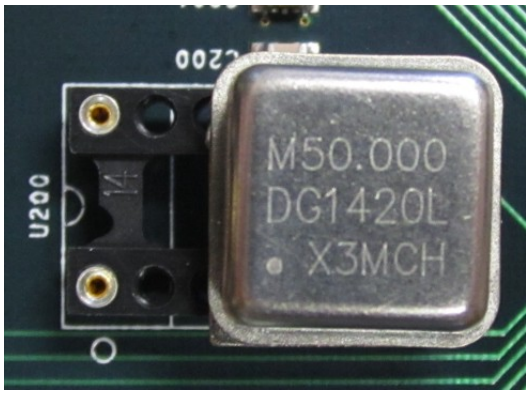


Figure 15: Both the bare header socket and the installed clock module are shown above. Please note the proper orientation of the 50 MHz clock. Pin 1 of the oscillator socket on the SP49, SP47, SP51, and SP56 outputs 1.65 V. Pin 1 of the oscillator socket on the SP18A is no connect.

Please take caution to provide a controlled signal at the correct frequency. The PulseBlasterESR-PRO 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.

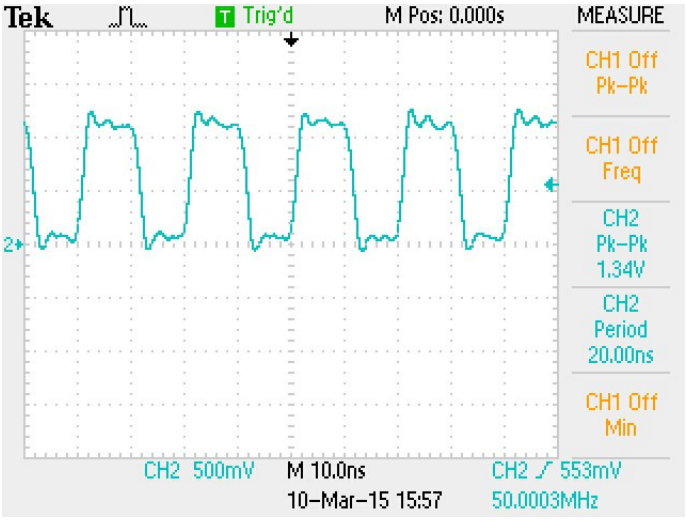


Figure 16: Example clock signal. Note that a small degree of voltage ripple is acceptable, so long as the voltage always remains above threshold for logical-high signals and below for logical-low signals.

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

Appendix I: Controlling the PulseBlasterESR-PRO with SpinAPI

Instruction Set Architecture

Machine-Word Definition

The PulseBlasterESR-PRO pulse timing and control processor implements an 80-bit wide Very Long Instruction Word (VLIW) architecture. The VLIW is partitioned into fields dedicated to specific purposes, and every VLIW is viewed as a single instruction by the microcontroller. The maximum number of instructions that can be loaded onto the PulseBlasterESR-PRO is firmware-dependent (4096 for 500 MHz designs, 32768 for designs 34-10, 34-15, and 31-3), see Firmware Designs. The execution time of instructions can be varied and is under (self) control by one of the fields of the instruction word – the shortest being five clock cycles and the longest being 2^{32} clock cycles.

Breakdown of 80-bit Instruction Word

All instructions have the same format and bit length, and all bit fields need to be filled. Table 5 shows the fields and bit definitions of the 80-bit instruction word.

Bit Definitions for the 80-bit Instruction Word (VLIW)

Output/Control Word	Data Field	OpCode	Delay Count
24 bits	20 bits	4 bits	32 bits

Table 5: Partitioning of the 80-bit Instruction Word (VLIW).

The 80-bit VLIW is broken up into 4 sections:

1. Output Pattern and Control Word: 24 bits.
2. Data Field: 20 bits.
3. OpCode: 4 bits.
4. Delay Count: 32 bits.

Output Pattern and Control Word

Table 6 shows the output pattern and control bit assignments of the 24-bit output/control word.

Bit #	Function	Bit #	Function
23	Controls Pulse Length for BNC connectors	11	Output Connector labeled Flag0..11, Pin 12
22	Controls Pulse Length for BNC connectors	10	Output Connector labeled Flag0..11, Pin 11
21	Controls Pulse Length for BNC connectors	9	Output Connector labeled Flag0..11, Pin 10
20	Output Connector labeled Flag12..23, Pin 9	8	Output Connector labeled Flag0..11, Pin 9
19	Output Connector labeled Flag12..23, Pin 8	7	Output Connector labeled Flag0..11, Pin 8
18	Output Connector labeled Flag12..23, Pin 7	6	Output Connector labeled Flag0..11, Pin 7
17	Output Connector labeled Flag12..23, Pin 6	5	Output Connector labeled Flag0..11, Pin 6
16	Output Connector labeled Flag12..23, Pin 5	4	Output Connector labeled Flag0..11, Pin 5
15	Output Connector labeled Flag12..23, Pin 4	3	Output Connector labeled Flag0..11, Pin 4
14	Output Connector labeled Flag12..23, Pin 3	2	Output Connector labeled Flag0..11, Pin 3
13	Output Connector labeled Flag12..23, Pin 2	1	Output Connector labeled Flag0..11, Pin 2
12	Output Connector labeled Flag12..23, Pin 1	0	Output Connector labeled Flag0..11, Pin 1

Table 6: Output Pattern and Control Word Bits.

When the bit corresponding to an IDC output connector is one, the voltage will be high for the duration of the instruction. If the bit is zero, the voltage will be low for the duration of the instruction.

Short Pulse Feature

The Short Pulse feature utilizes the upper three bits of the instruction flag output bits (bits 21 to 23) to control the number of clock cycles output flags are enabled. This allows for short pulses down to a single clock cycle during the instruction period. Note that **all flags** are synchronized to this instruction period. It is possible to create pulses longer than this length by setting a channel on for multiple instructions, so the Short Pulse feature does not limit the maximum length of pulses. The following table provides information on using the Short Pulse feature.

SpinAPI Define	Bits 21-23	Clock Periods	Pulse Length at 500 MHz (ns)
-	000	-	No Short Pulse. *See note (1).*
ONE_PERIOD	001	1	2
TWO_PERIOD	010	2	4
THREE_PERIOD	011	3	6
FOUR_PERIOD	100	4	8
FIVE_PERIOD	101	5	10
ON	111	-	No Short Pulse

Table 7: Short Pulse Feature Characteristics.

When bits 23-21 are from “000” to “101,” the programmed flag values will be outputted for the specified number of clock cycles. To disable the Short Pulse feature, bits 21 to 23 must be set to “111” or “000”.

Figure 17 gives an example of the Short Pulse feature. The example uses a 3 period duration. This example only shows 4 flags, but all flags will be affected.

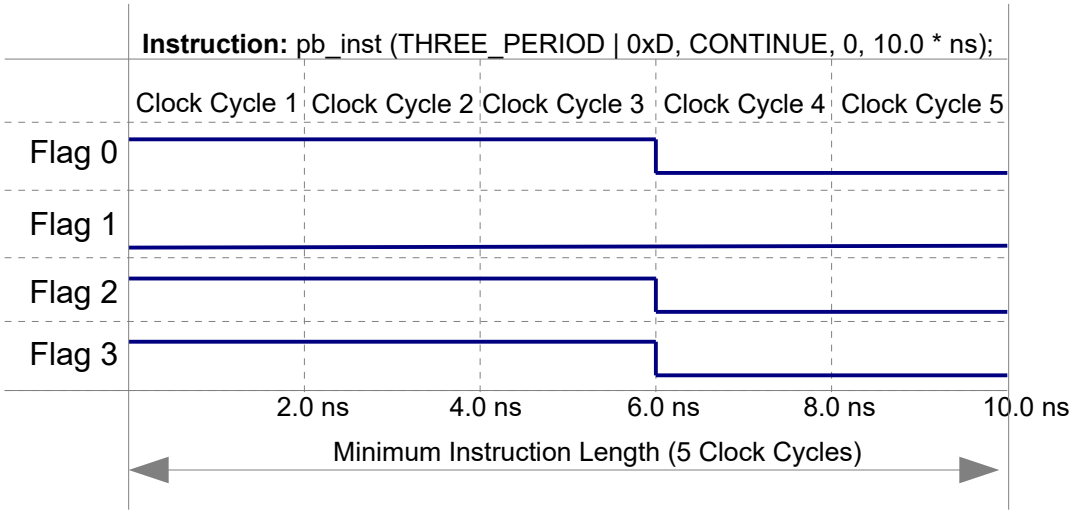


Figure 17: Example of the Short Pulse Feature. This example uses a 3 period duration. This example displays the output of 4 flags, but all flags are affected by the Short Pulse feature. Timing is done using a 500.0 MHz Clock.

NOTE: The Short Pulse functionality is firmware-dependent. Please inquire with SpinCore Technologies for details.

Data Field and OpCode

Please refer to the following table for information on the available instructions and their associated data field argument.

OpCode #	Inst	Inst_data	Function
0	CONTINUE	Ignored	Program execution continues to next instruction. *See note (1) following this table.*
1	STOP	Ignored	Stop execution of program. Aborts the operation of the micro-controller with no control of output states (all TTL values may remain from previous instruction). *See note (2).* Recommended that prior to the STOP OpCode a short interval be added to set the output states as desired.
2	LOOP	Number of desired loops. This value must be greater than or equal to 1.	Specify beginning of a loop. Execution continues to next instruction. Data used to specify number of loops
3	END_LOOP	Address of beginning of loop	Specify end of a loop. Execution returns to beginning of loop and decrements loop counter.
4	JSR	Address of first subroutine instruction	Program execution jumps to beginning of a subroutine.
5	RTS	Ignored	Program execution returns to instruction after JSR was called.
6	BRANCH	Address of next instruction	Program execution continues at specified instruction
7	LONG_DELAY	Desired multiplier of the delay. This value must be greater than or equal to 2.	For long interval instructions. Data field specifies a multiplier of the delay field. Execution continues to next instruction.
8	WAIT	Ignored	Program execution stops and waits for software or hardware trigger. Execution continues to next instruction after receipt of trigger. The latency is equal to the delay value entered in the WAIT instruction line plus a fixed delay of 6 clock cycles. The WAIT OpCode may not be used by the first instruction in memory.

(1) For instructions longer than 2^{32}-1 clock cycles (8589 ms for 500 MHz, 10737 ms for 400 MHz), please use a LONG_DELAY instruction.

(2) PulseBlasterESR-PRO-500-PCI (designs 17-16, and 17-11), PBESR-PRO-500-USB-RM (designs 27-8, and 33-1), PBESR-PRO-500-USB-RM-FP (design 33-1), PBESR-PRO-500-PCIe (designs 31-1, 31-3, 34-1, and 34-12), and PBESR-PRO-400-32k-PCIe (designs 34-10, 34-15, and 31-3) hold TTL values of current instruction.

Table 8: OpCode and Data Field Description.

Delay Count

The value of the Delay Count field (a 32-bit value) determines how long the current instruction should be executed. The allowed minimum value of this field is 0x00000002 and the allowed maximum is 0xFFFFFFFF. The timing controller has a fixed delay of three clock cycles and the value that one enters into the Delay Count field should account for this inherent delay. (NOTE: the pb_inst() family of functions in SpinAPI and the PulseBlaster Interpreter automatically account for this delay.)

About SpinAPI

SpinAPI is a control library which allows programs to be written that can communicate with the PulseBlasterESR-PRO board. The most straightforward way to interface with this library is with a C/C++ program, and the API definitions are described in this context. However, virtually all programming languages and software environments (including software such as LabVIEW and MATLAB) provide mechanisms for accessing the functionality of SpinAPI.

Please see the example programs for examples of how to use SpinAPI. If the programs have not been installed, then information to installing and finding them can be found in the “Installing the PulseBlasterESR-PRO” section. Reference documents for the API are available online at:

http://www.spincore.com/CD/spinapi/spinapi_reference/

http://www.spincore.com/support/spinapi/

Appendix II: Available Firmware Designs

The following table contains information about the various firmware designs available on the PulseBlasterESR-PRO series of boards mentioned in this Manual.

Firmware Revision	Board	Clock Speed (MHz)	Number of Output Bits	Memory Depth (words)	Minimum Instruction Duration (Clock Cycle)	Communication Interface
17-11	SP18A	500	21	4k	7	PCI
17-16	SP18A	500	21	4k	6	PCI
27-8	SP47	500	21	4k	6	USB
31-1	SP49	500	21	4k	6	PCIe
31-3	SP49	400	21	32k	7	PCIe
31-4	SP49	500	21	4k	6	PCIe
33-1	SP51	500	21	4k	6	USB
34-1	SP56	500	21	4k	6	PCIe
34-10	SP56	400	21	32k	7	PCIe
34-12	SP56	500	21	4k	6	PCIe
34-15	SP56	400	21	32k	7	PCIe

Table 9: Firmware designs available for the PulseBlasterESR-PRO series of boards mentioned in this Manual.

Related Products and Accessories

1. Oven Controlled Clock Oscillator (sub-ppm stability) shown in Figure 18. For ordering information, please visit http://spincore.com/products/OCXO/ or contact SpinCore at http://www.spincore.com/contact.shtml.

Figure 18: An Oven Controlled Clock Oscillator (or OCXO) with sub-ppm frequency stability is available for the PulseBlasterESR-PRO upon request.

2. Ribbon Cable with 2x13 IDC plug and DB-25 (Parallel port style*) connector on PC bracket. Please note this uses a standard computer ribbon cable. For ordering information, please visit http://spincore.com/products/InterfaceCable/ or contact SpinCore at http://www.spincore.com/contact.shtml.

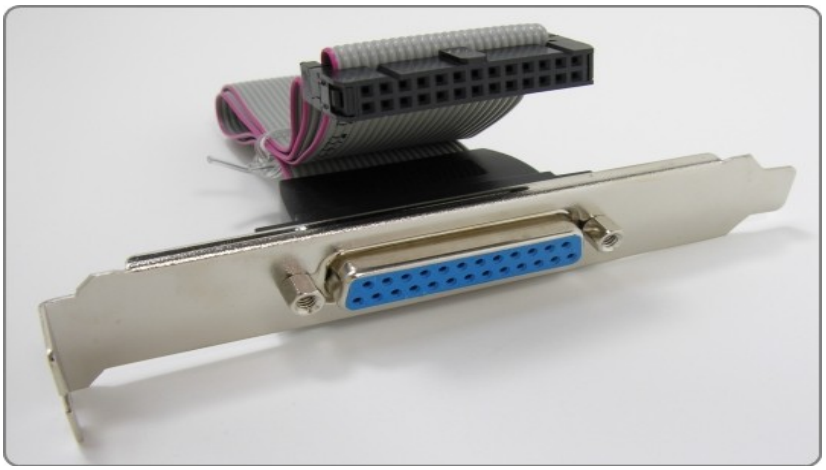


Figure 19: PulseBlasterESR-PRO Parallel Port Interface Cable.

***Note: This is NOT a parallel port and will not work with a PC printer or other such peripheral devices! This cable uses the parallel type DB-25 connector to easily access the TTL bits of the PulseBlasterESR-PRO Board.**

3. SpinCore MMCX Adapter Board (Figures 20 and 21) – These adapter boards allow easy access to the individual bits of the PulseBlasterESR-PRO boards. These adapter board can be part of a package that includes 12 MMCX to BNC cables and three SMA to BNC adapters. However, this package can be changed to include any number of cables and any number of adapter boards. For ordering information, please visit http://spincore.com/products/Adapters/ or contact SpinCore at http://www.spincore.com/contact.shtml.

Figure 20: SP32 MMCX Adapter Board allows easy access to individual bits.

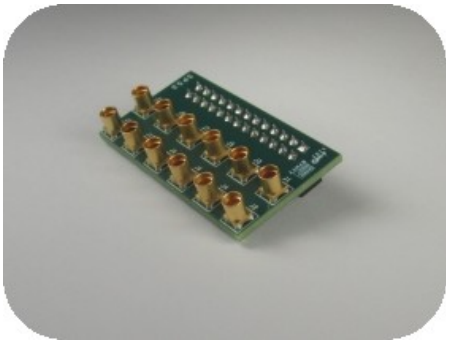


Figure 21: SP53 MMCX Adapter Board allows easy access to individual bits.

Figure 22: SP32, MMCX Adapter Board orientation and configuration with PulseBlaster SP18A Board.

Figure 23: SP53, MMCX Adapter Board orientation and configuration with PulseBlaster SP18A Board.

4. The SMA-BNC Adapter Board, shown in Figure 24, provides easy access to four additional output signals from the back panel of your computer. SMA-SMA cables are available from SpinCore upon request. For ordering information, please visit http://spincore.com/products/Adapters/ or contact SpinCore at http://www.spincore.com/contact.shtml.

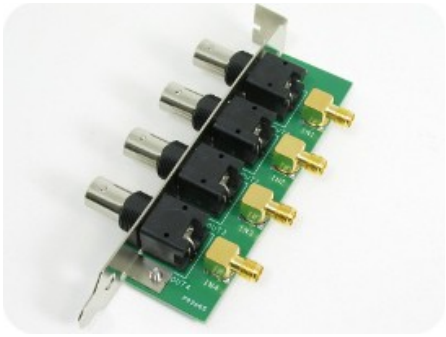


Figure 24: SMA-BNC Adapter Board is available for the SP18A board to access additional flag bits.

5. IDC to BNC Adapter Set-Up on a PulseBlasterESR-PRO Board Figure 25 – Additional BNC output signals can be accessed using a set-up consisting of an IDC-MMCX adapter board (SP32), MMCX-SMA cables, and an SMA-BNC adapter board (SP29P).

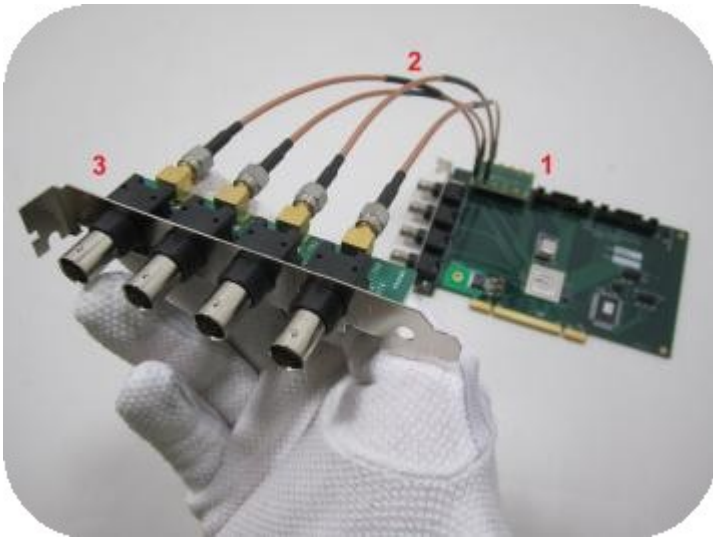


Figure 25: IDC to BNC Adapter Set-Up with a PulseBlasterESR-PRO Board.

6. SpinCore TTL Line Driver Figure 26 - A USB-powered device with four input channels and 8 output lines. Each output line is equipped with current driving capabilities to insure TTL voltage level over 50 Ohm loads. The SpinCore TTL Line Driver is the perfect tool to accompany any TTL device. Additional specifications, ordering information, and the manual for the TTL Line Driver are available at http://www.spincore.com/products/SpinCoreTTLLineDriver/SpinCoreTTLLineDriver.shtml.

Figure 26: TTL Line Driver assures TTL levels over 50 Ohm loads.

7. If you require a custom design, custom interface cables, or other custom features, please inquire with SpinCore Technologies through our contact form, which is available at http://www.spincore.com/contact.shtml.

Contact Information

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

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

Document Information

Revision history available at SpinCore.