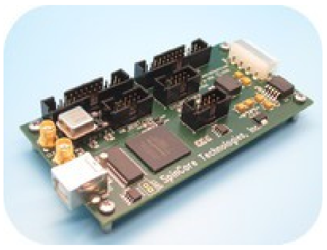
PulseBlasterUSB™ (v2)

(USB Board SP50A, SP59)

(USB Rackmount System, USB Rackmount2 System, Rackmount Front Panel SP55)

Owner's Manual



SpinCore Technologies, Inc.
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

Product Overview

The PulseBlasterUSB™ is a programmable multichannel pulse/delay generator that is capable of generating pulses and delays ranging from 60 ns to 4.5x10^7 seconds (~521 days) in length when operating with a 100 MHz clock frequency. The PulseBlasterUSB has up to 24 output channels, features highly flexible program flow control, and can accommodate pulse programs containing up to 8192 instructions (see Appendix II: Available Firmware Designs). The PulseBlasterUSB is capable of generating complex output pulse patterns featuring very short and very long pulses in the same sequence with a resolution of one clock period (i.e., 10 ns at 100 MHz).

The intelligence of the PulseBlasterUSB comes from its proprietary microprocessor core architecture. Unlike general-purpose processors, the PulseBlaster processor features a highly optimized instruction set that has been specifically designed for timing applications. A unique and distinguishing feature of the PulseBlaster processor is that the execution time of instructions is user programmable.

User interaction with the PulseBlasterUSB can be accomplished in several different ways. The *PulseBlaster Interpreter* and *PulseBlaster.NET*, SpinCore's graphical programming environments for Windows, allow users to easily create, edit, save and run pulse programs. Graphical User Interfaces (GUI) for MATLAB and LabVIEW have also been developed and are available for download from SpinCore Technologies, Inc. at no charge. PulseBlasterUSB can be controlled through user input of either software or hardware triggering.

Development environments like LabVIEW, MATLAB, and C/C++ can be used to interface with the PulseBlasterUSB through the dedicated Application Programming Interface (API) package called SpinAPI. The SpinAPI package can be used with virtually all Windows programming environments and can also be used with other operating systems, including Linux.

The PulseBlasterUSB is a USB 2.0 device, and can be easily integrated into a wide range of computer systems.

The PulseBlasterUSB is available in four standard options: a) the board-only option with no enclosure (+5 V DC required), b) the 2U BNC 10” depth Rackmount enclosure (full width), and c) the 2U BNC 2” depth Rackmount enclosure (full width) d) the 2U Rackmount front panel. The 2U BNC 10” depth Rackmount enclosure enclosed option includes the internal AC/DC power supply.

The 2U BNC Rackmount enclosure (full width) is standard.

Board Architecture

Block Diagram

Figure 1 presents the internal architecture of the PulseBlasterUSB system. The major building blocks are the SRAM memory, the microcontroller (uPC), the integrated bus controller (IBC), the counter, and the output buffers. The entire logic design, including output buffers, is contained on a single silicon chip, making it a System-on-Chip design. User control of the system is accomplished via the universal serial bus (USB).

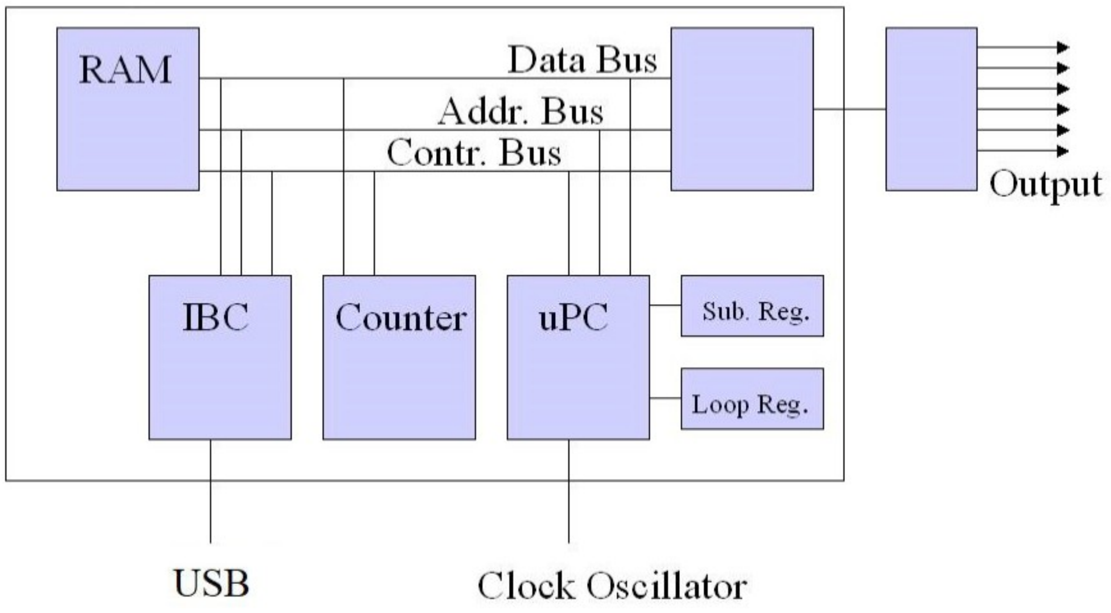


Figure 1: PulseBlasterUSB system architecture. The clock oscillator signal originates from an on-chip PLL circuit with a typical base frequency of 50 MHz.

The next page describes the *general* characteristics of the PulseBlasterUSB. For more detailed information on the *internal* operation of the PulseBlaster processor, please see **Appendix I. Instruction Set Architecture**.

Output Signals

The PulseBlasterUSB has up to 24 digital output signal lines. All output signal lines are impedance matched to 50 ohm. The individually controlled digital output bits are 3.3V LVTTL standard unterminated, and are capable of delivering in excess of 25 mA per bit/channel.

Timing Characteristics

The PulseBlasterUSB's timing controller accepts an on-board crystal oscillator (or externally applied 3.3V LVTTL clock signal) of 50 MHz. The innovative architecture of the timing controller allows the processing of either simple timing instructions (delays of up to 2^{32} –1 clock cycles, e.g., approximately 43 seconds at 100 MHz), or double-length timing instructions (up to 2^{52} –1 clock cycles, equivalent to 4.5×10^7 seconds, or approximately 521 days, at 100 MHz). Regardless of the type of timing instruction, the timing resolution remains constant for any delay – just one clock period (e.g., 10 ns at 100 MHz).

The core timing controller has a very short minimum delay cycle – only six clock periods. This translates to a 60 ns shortest pulse/delay at 100 MHz.

Instruction Set

The PulseBlasterUSB's instruction set allows for highly flexible program flow control. The core microcontroller allows for programs to include branches, subroutines, and loops at up to 8 nested levels – all of this to assist the user in creating dense pulse programs that can cycle through repetitious events, such as those required for multidimensional spectroscopy and imaging applications.

External Triggering

The PulseBlasterUSB can be triggered and/or reset externally via dedicated hardware lines. These two hardware lines, Hardware Trigger and Hardware Reset, are pulled high via 10 kOhm resistors and the required control signals are "low-true." This means the control signals will become true on the falling edge and remain true while held at ground. Trigger occurs when the trigger pin is shorted to ground. To protect hardware, do not exceed the recommended voltage levels.

Summary

The PulseBlasterUSB is a versatile, high-performance, programmable LVTTL signal generator which operates at 100 MHz and is capable of generating pulses/delays/intervals ranging from 60 ns to 4.5×10^7 s in duration. The PulseBlasterUSB has up to 24 output channels, features highly flexible program flow control, and can accommodate pulse programs containing as many as 8192 instructions (see Appendix II: Available Firmware Designs).

Each of the PulseBlasterUSB's outputs is individually controlled, has a standard LVTTL 3.3 V output voltage[1], and can provide in excess of 25 mA of current. User interaction and control can be accomplished in several ways: through SpinCore's *PulseBlaster Interpreter* and *PulesBlaster.NET*, which are free graphical programming environments for Windows, or through the SpinAPI control library which is also free of charge and can be integrated into C/C++, MATLAB, LabVIEW and other third-party design environments.

[1] No-load voltage level

Specifications

Output Channels

- Up to 24 individually controlled digital output channels (LVTTL levels, 3.3 V unterminated high state)
- Variable pulses/delays for every output channel
- 25 mA output current per output channel

Pulse Timing

- 60 ns shortest pulse/interval (at 100 MHz)
- 4.5×10^7 s longest pulse/interval (at 100 MHz)
- 10 ns pulse/interval resolution (at 100 MHz)
- LVTTL-level (3.3 V max.) external trigger and reset inputs

Pulse Program Flow Control

- Subroutines, nested up to 8 levels deep
- Loops, nested up to 8 levels deep
- 20-bit loop counters (max. 1,048,575 repetitions per loop)
- Wait for trigger with 8 clock cycle latency (80 ns at 100 MHz), can wait up to 4.5×10^7 seconds (~521 days)
- Status polling via hardware and software
- Up to 8192 instructions (see Appendix II: Available Firmware Designs)

II. Installation

Installing the SpinAPI Driver and Control Library

SpinAPI is a custom Application Programming Interface developed by SpinCore Technologies, Inc. for use with the PulseBlaster board. SpinAPI can be utilized using C/C++ or graphically by using other methods detailed in Section III.

NOTE: To ensure proper installation of the board, you must uninstall any previous versions of SpinAPI and complete the following:

- Please do not connect the system/ board until SpinAPI has been installed.
- Install the latest version of SpinAPI found at: http://spincore.com/support/spinapi/.
- After installation, shut down the computer, plug the PulseBlasterUSB's USB cable into the computer and then restart.

Now you are ready to run the test C programs that came in the SpinAPI Package. These example files can be found in the "examples" folder that came with the SpinAPI Package. Open the "PulseBlaster24" sub-folder to access the pre-compiled examples.

Testing the PulseBlaster

The simplest way to test whether the PulseBlasterUSB has been installed properly and can be controlled as intended is to run a simple test program. The pb24_ex1.exe program, for example, will cause the outputs to turn on and off with a period of 400ms. To test the board, run pb24_ex1.exe and observe each digital output with an oscilloscope or with LEDs.

If using a high input impedance oscilloscope to monitor the PulseBlaster's output, 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 Figure 2 and Figure 3 on the next page).

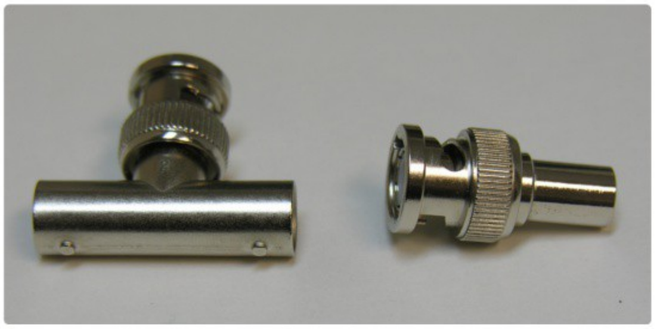


Figure 2: Left: BNC T-Adapter and Right: BNC 50 Ohm resistor.

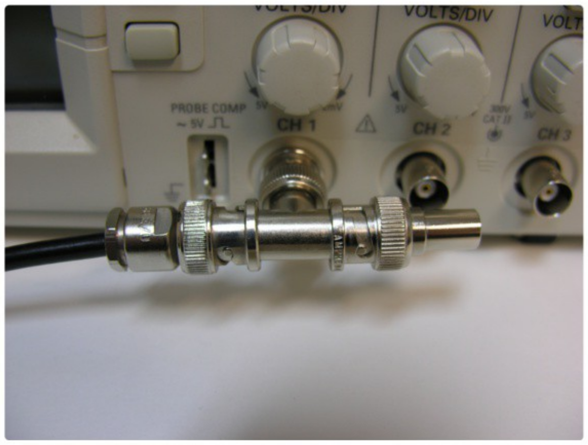


Figure 3: BNC T-Adapter on the oscilloscope with coaxial transmission line connected on the left and BNC 50 Ohm resistor connected on the right, to terminate the line.

Figure 4 shows what you should expect to see on your oscilloscope when running pb24_ex1.exe and using a 50 Ohm resistor to terminate the line. Once this behavior has been verified, the user can be confident the board is installed properly.

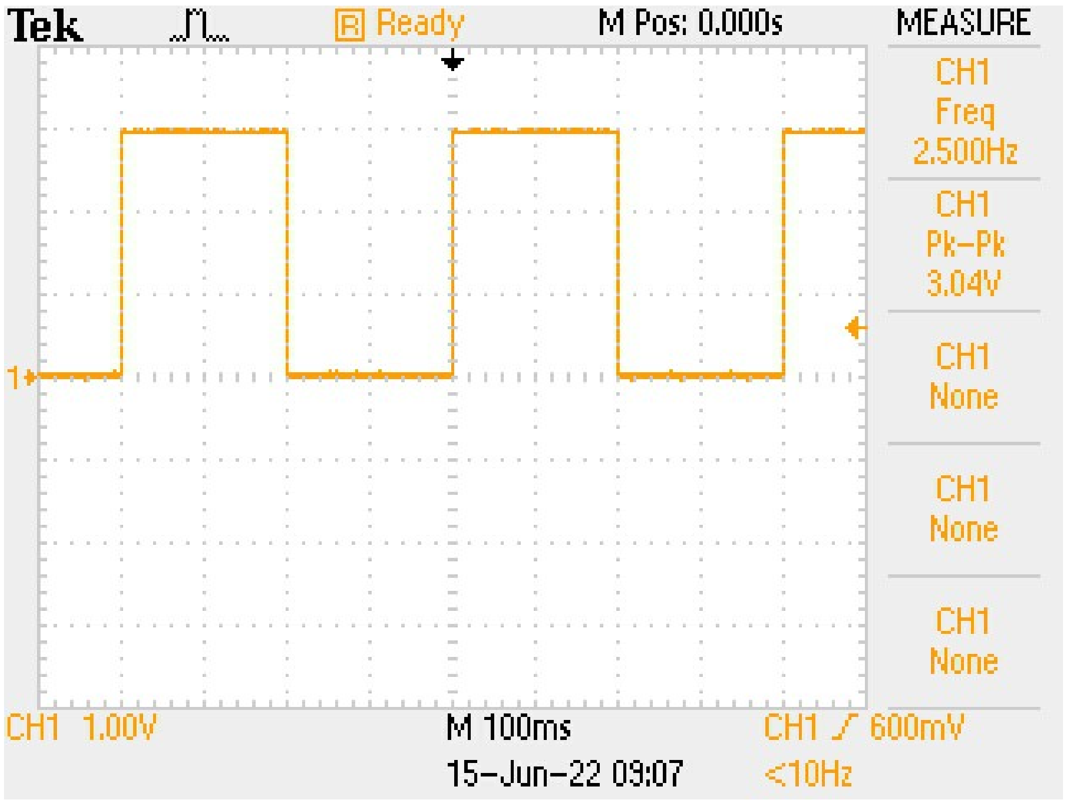


Figure 4: The expected signal from a PulseBlasterUSB (SP50A) running example program pb24_ex1.exe. Notice the amplitude of 3.04 V is due to a 50 Ohm resistor which is terminating the line.

III. Programming the PulseBlasterUSB

Introduction

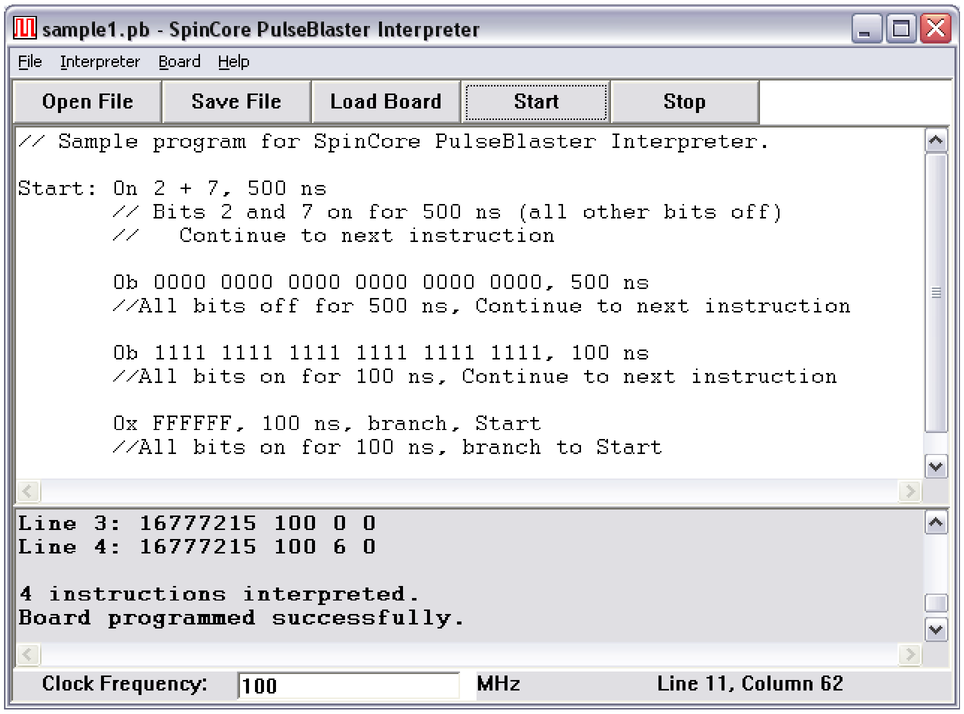
There are five major methods of programming the PulseBlasterUSB. The simplest and most intuitive methods of control are the graphical interfaces provided by *PulseBlaster Interpreter* and *PulseBlaster.NET.* More advanced control can be exercised by writing custom C/C++ programs which will interact with the SpinAPI control package.

Programming can also be accomplished through the use of third-party GUI packages such as LabVIEW and MATLAB which have the ability to access Dynamic Link Libraries (DLLs) like SpinAPI. All five programming methods are described in detail below.

Programming with the PulseBlaster Interpreter

The PulseBlasterUSB can be programmed and controlled via the *PulseBlaster Interpreter*, which is an easy to-use editor that allows you create, edit, save and run your pulse sequence.

With the PulseBlaster Interpreter, pulse patterns can be generated using the simple notation of natural numbers (see the first line in Figure 5, below). For more complex bit patterns, the PulseBlaster Interpreter also accepts binary notation and hexadecimal notation (see lines 2, 3, and 4 in Figure 5, below).



```
// Sample program for SpinCore PulseBlaster Interpreter.

Start: On 2 + 7, 500 ns
       // Bits 2 and 7 on for 500 ns (all other bits off)
       //   Continue to next instruction

       0b 0000 0000 0000 0000 0000 0000, 500 ns
       //All bits off for 500 ns, Continue to next instruction

       0b 1111 1111 1111 1111 1111 1111, 100 ns
       //All bits on for 100 ns, Continue to next instruction

       0x FFFFFF, 100 ns, branch, Start
       //All bits on for 100 ns, branch to Start
```

```
Line 3: 16777215 100 0 0
Line 4: 16777215 100 6 0

4 instructions interpreted.
Board programmed successfully.
```

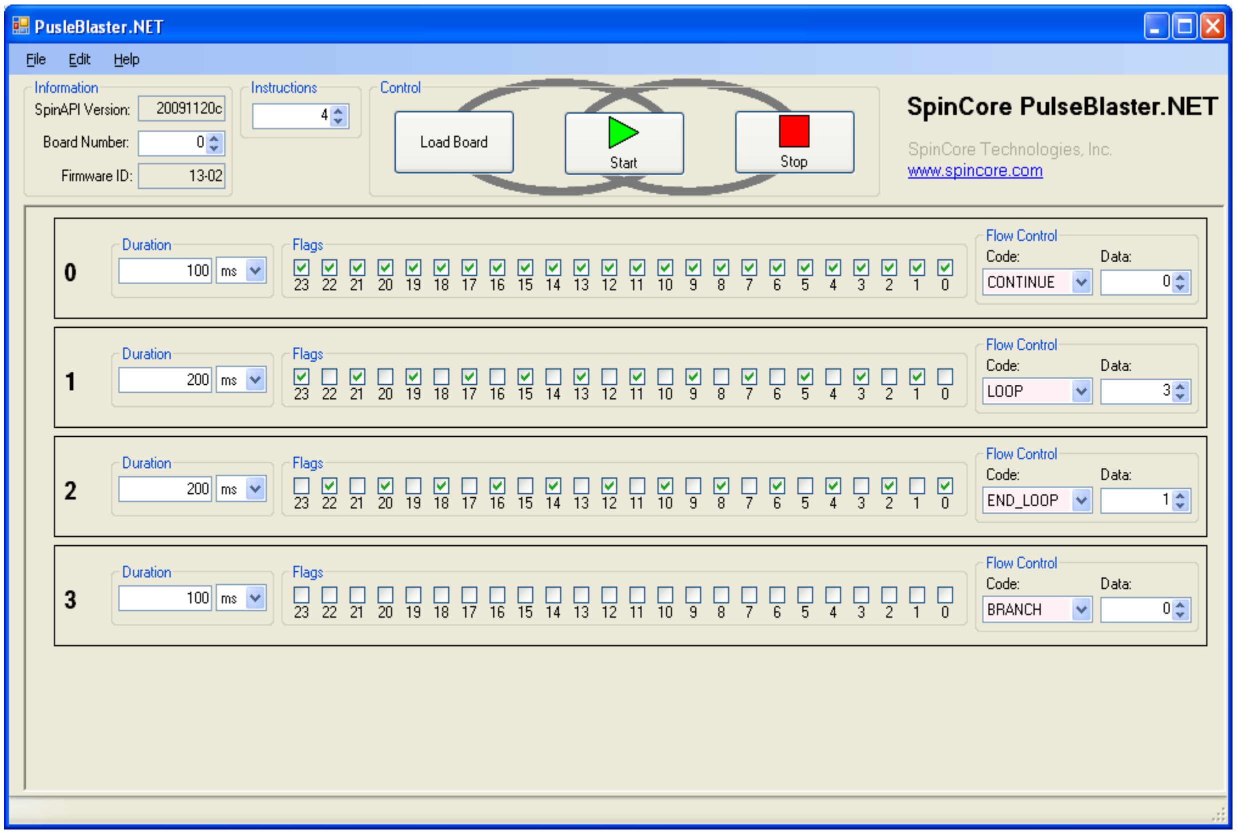
Clock Frequency: 100 MHz   Line 11, Column 62

Figure 5: The PulseBlaster Interpreter.

Please follow the link http://www.spincore.com/support/SPBI/ for the most recent version of the PulseBlaster Interpreter software, installation instructions, documentation, and sample programs.

Programming with PulseBlaster.NET

The PulseBlasterUSB can be programmed and controlled using an intuitive .NET GUI designed in C#.



SpinAPI Version: 20091120c
Board Number: 0
Firmware ID: 13-02
Instructions: 4

	Duration	Unit	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Code	Data
0	100	ms	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	CONTINUE	0
1	200	ms	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	LOOP	3
2	200	ms	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	☐	✓	END_LOOP	1
3	100	ms	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	BRANCH	0

Figure 6: The PulseBlaster.NET Interface.

The PulseBlaster.NET interface provides a quick and intuitive method for designing pulse programs for your device. Programming is no longer done in text, but instead presented as a simple visual metaphor. Instructions are presented as a series of checkboxes to represent the LVTTL values produced.

PulseBlaster.NET is capable of saving and loading files that may be transferred between computers. Best of all, any programming errors (e.g. invalid opcode) are indicated to the user immediately, eliminating much debugging work.

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

Programming in C/C++ using SpinAPI

The SpinAPI control library allows users to write pulse programs for controlling the PulseBlasterUSB with C/C++ functions. The SpinAPI package also contains several precompiled example pulse programs, along with their source code. These example programs provide an easy way for the user to verify the proper operation of the device and will also provide the user with a jump-start in programming with C/C++ and SpinAPI.

Programming the PulseBlasterUSB in C/C++ uses a simple paradigm in which each generated interval corresponds to a single instruction. Every instruction is programmed with the state of the output channels for that interval, the duration (time) and the program flow information (Op Code and data). An example is shown below.

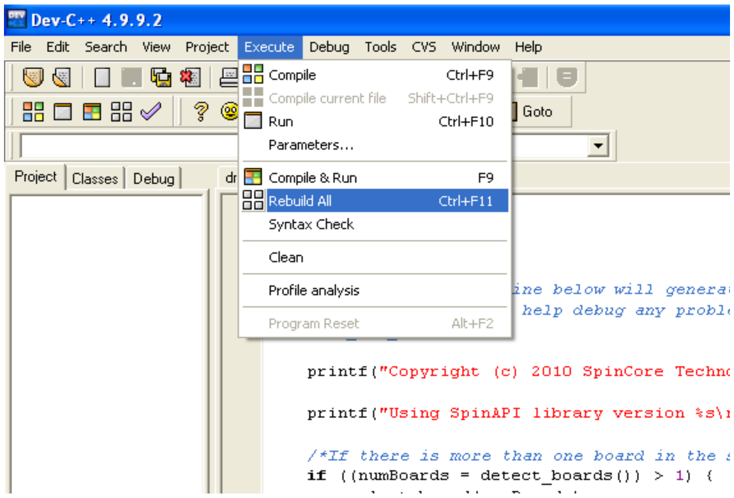
```
int start = pb_inst(0xFFFFFF, CONTINUE, 0, 500*ms);
            pb_inst(0x000000, BRANCH, start, 500*ms);
```

Example 1: Code segment to produce a 1Hz square wave on all outputs. The first parameter of pb_inst corresponds to the state of the outputs; the second and third, the Op Code and data; and the fourth to the duration of the interval.

Use of this programming paradigm is explained in further detail in the document titled “Using SpinAPI in C/C++ for PulseBlaster Programming” which is available from the SpinCore website at http://spincore.com/support/spinapi/using_spin_api_pb.shtml. The SpinAPI reference document can be found at http://www.spincore.com/CD/spinapi/spinapi_reference/.

The SpinAPI package is available for download free of charge. Detailed explanations of every available API function are available on-line, and the source code of the entire SpinAPI package is also freely available. This may be of value to those users wishing to port the package to other operating systems or platforms. For more information about the SpinAPI package, see http://spincore.com/support/spinapi/.

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 PulseBlasterUSB board is as easy as clicking “Rebuild All” (see Figure 7 below).

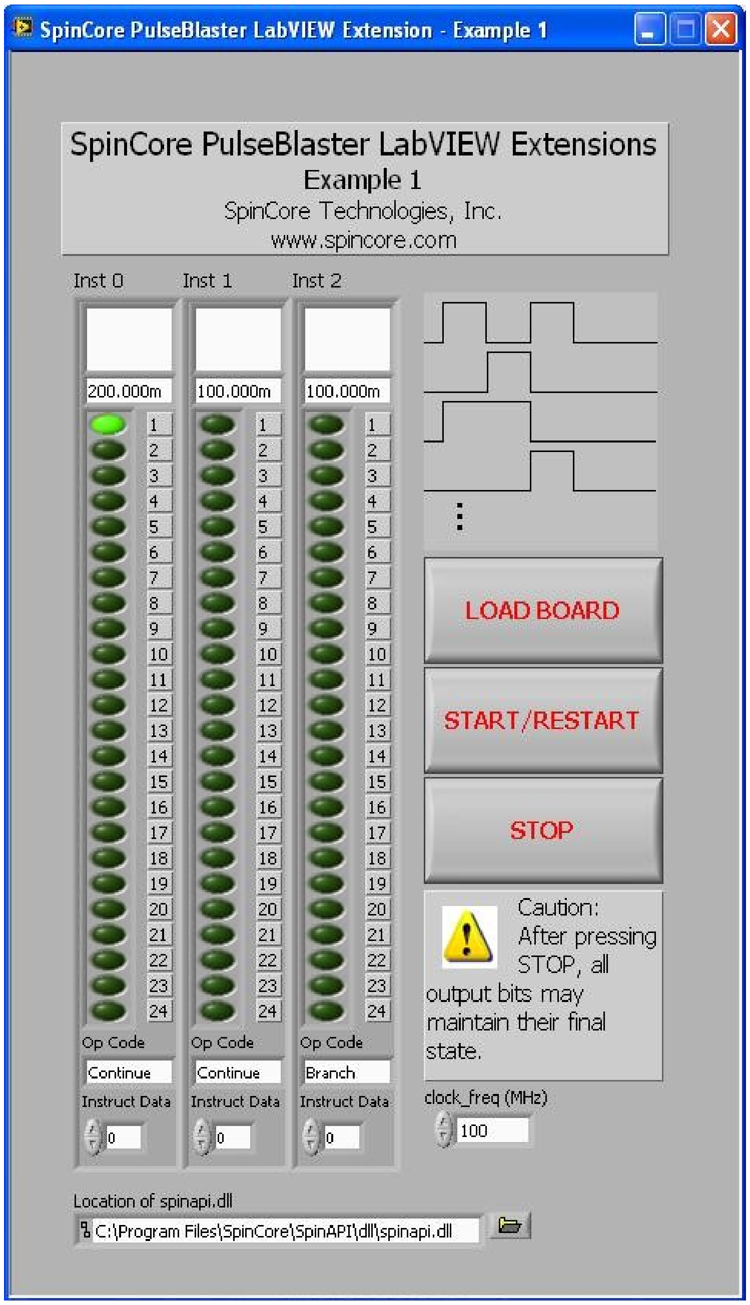


```
ine below will genera
 help debug any probl

printf("Copyright (c) 2010 SpinCore Techn

printf("Using SpinAPI library version %s\

/*If there is more than one board in the 
if ((numBoards = detect_boards()) > 1) {
```

Figure 7: Compiling a C example program.

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.



SpinCore PulseBlaster LabVIEW Extension - Example 1

SpinCore PulseBlaster LabVIEW Extensions
Example 1
SpinCore Technologies, Inc.
www.spincore.com

	Inst 0	Inst 1	Inst 2
Duration	200.000m	100.000m	100.000m
Op Code	Continue	Continue	Branch
Instruct Data	0	0	0

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

clock_freq (MHz) 100

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

Figure 8: Example of PulseBlaster LabVIEW Extensions User Interface. The example shown has three instructions that toggle LVTTL bit 1 on for 200 ms and off for 200 ms.

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 8 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 PulseBlaster board. For more information and downloads please visit http://www.spincore.com/support/PBLV/.

Programming with Third-Party GUI Environments

The SpinAPI control library is a standard Windows Dynamic Link Library (DLL), and virtually all programming languages and software environments (including LabVIEW and MATLAB) provide mechanisms for accessing the functionality of libraries like SpinAPI.

IV. Connecting to the PulseBlasterUSB

The PulseBlasterUSB is available in four standard options: a) the board-only option with no enclosure (+5 V DC required), b) the 2U BNC 10” depth Rackmount enclosure (full width), and c) the 2U BNC 2” depth Rackmount enclosure (full width) d) the 2U Rackmount front panel.

Connector Information for the SP50 and SP59 PulseBlasterUSB Boards

The power connectors are different between the SP50 and SP59 PulseBlasterUSB boards. All of the other connectors are identical.

SP50 Power Connector

The SP50 PulseBlasterUSB board-only option has a 4-pin Molex-style connector for supplying power. The pin and signal arrangements for this connector is as follows:

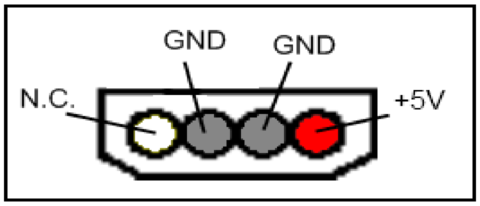


Figure 9: 4 Pin input connector (Molex part 0531090410).

This power connector mates with standard PC power supply connector or Molex part 0015244048.

Note that it is suggested that a PC power supply with the connector specified below 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 PulseBlasterUSB board 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.

SP59 Power Connector

The SP59 PulseBlasterUSB is powered through the USB-C connector labeled PWR. The power source must be capable of providing 5V/3A, otherwise the board does not turn on. This connector does not have any data capabilities.

Shrouded IDC Connector FLAG[0..11]_OUT Pin Assignments

The shrouded IDC connector labeled Flag[0..11]_OUT outputs LVTTL signals generated by the user's program. Please consult the table below for pin assignments.

Pin Assignments			
Pin#		Pin#	
1	Bit 0	14	GND
2	GND	15	Bit 7
3	Bit 1	16	GND
4	GND	17	Bit 8
5	Bit 2	18	GND
6	GND	19	Bit 9
7	Bit 3	20	GND
8	GND	21	Bit 10
9	Bit 4	22	GND
10	GND	23	Bit 11
11	Bit 5	24	GND
12	GND	25	NC
13	Bit 6	26	GND

Table 1: Lower 12 output bits and 12 ground lines on the 26 pin IDC connector.

The shrouded IDC connector labeled FLAG[0..11]_OUT can also be accessed using an SP53 board (Figure 16 on page 31) which allows the use of MMCX cables. This enables the individual bits of the PulseBlaster to be more easily accessed. Pin 1 on the MMCX adapter board can be identified with a square pin.

Shrouded IDC Connector FLAG[12..23]_OUT Pin Assignments

The shrouded IDC connector labeled Flag[12..23]_OUT outputs LVTTL signals generated by the user's program. Please consult the table below for pin assignments.

Pin Assignments			
Pin#		Pin#	
1	Bit 12	14	GND
2	GND	15	Bit 19
3	Bit 13	16	GND
4	GND	17	Bit 20
5	Bit 14	18	GND
6	GND	19	Bit 21
7	Bit 15	20	GND
8	GND	21	Bit 22
9	Bit 16	22	GND
10	GND	23	Bit 23
11	Bit 17	24	GND
12	GND	25	NC
13	Bit 18	26	GND

Table 2: Higher 12 output bits and 12 ground lines on the 26 pin IDC connector.

The shrouded IDC connector labeled FLAG[12..23]_OUT can also be accessed using an SP53 board (Figure 16 on page 31) which allows the use of MMCX cables. This enables the individual bits of the PulseBlaster to be more easily accessed. Pin 1 on the MMCX adapter board can be identified with a square pin.

HW_RESET/TRIG IDC Header (Trig/Res/Stat) PulseBlasterUSB Board

For PulseBlasterUSB boards, the trigger, reset and status pins are available on the IDC Header labeled HW_RESET/TRIG. The pinout for the IDC header is shown in Figure 10 and Table 3 below. Please refer to the **Status and Hardware Pins** section on page 24 for more details about the status signals and hardware pins.

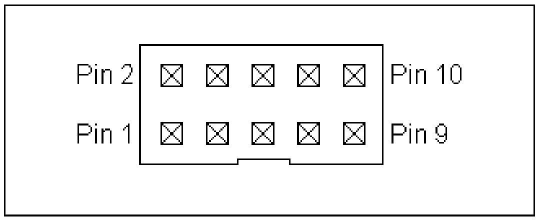


Figure 10: Shrouded IDC Header HW_RESET/TRIG.

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

Table 3: IDC Header HW_RESET/TRIG Pinout.

SMA Connector CLK_OUT

This SMA connector outputs the reference clock as a 3.3 V LVTTL signal, i.e., it generates positive- only voltage. Note that the PulseBlasterUSB boards use 50 MHz as the reference clock frequency and that clock is internally multiplied to provide that actual PulseBlaster Core frequency[2]. 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 LVTTL, 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 the R001 pad. 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.

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

Connector Information for 2U BNC Rackmount Enclosure, Rackmount2 Enclosure, and Rackmount Front Panel

Connectors for 2U BNC Rackmount Enclosure, Rackmount2 Enclosure, and Rackmount Front Panel

The front panel of the all three rackmount models provides 24 output bits through BNC connectors. They are arranged in a format which is 3 rows of 8 bits.

The front panel contains the USB connector, and the Trig/Res/Stat DB9 connector. The Rackmount Enclosure and Rackmount Front Panel models have an ON/OFF toggle switch. The Rackmount2 Enclosure model has a USB-C connector for power.

For the Rackmount Enclosure model, the ON/OFF toggle switch toggles the internal power supply of the PulseBlasterUSB. On the reverse side of this switch for this model is the AC Input.

For the Rackmount Front Panel model, the ON/OFF toggle switch is not connected to anything and can be used for custom user applications.

USB-B Data Connector

The USB-B connector is for communication with the PC. A USB 2.0 port or a USB port compatible with USB 2.0 is required for data transfer.

DB9 (DE9) Connector (Trig/Res/Stat) for 2U BNC Rackmount Enclosure and Rackmount Front Panel

The Trig/Res/Stat DB9 connector information is shown in Figure 11, below and Table 4, on the next page. The numbering of the pins may be different from the text on the physical DB9 connector. 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**, for additional information about each pins functionality.

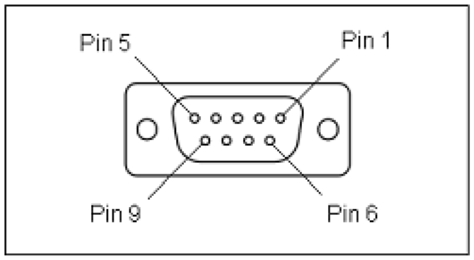


Figure 11: Trig/Res/Stat Male/Female DB9 connector drawing. This pin numbering is for both male and female DB9 connectors. When making a custom cable, starting with the mating DB9 connector may be helpful in recognizing where the pins are on the mating connector. This image is drawn with a view looking 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 positions are located in Figure 11. 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 PulseBlasterUSB Rackmount Front Panel option has a serial advanced technology attachment (SATA) connector for supplying power. The pin and signal arrangements for the power portion of the connector is as follows:

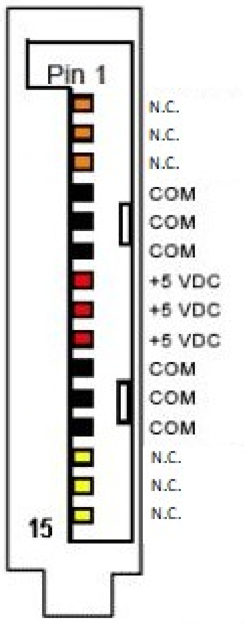


Figure 12: SATA pinout of the power connector. The data part of the SATA connector is not used.

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 PulseBlasterUSB Rackmount Front Panel does not contain overvoltage protection, reverse polarity protection, or undercurrent detection. It is suggested to use a PC power supply that can supply at least 3 A on the +5 V SATA rail.

USB-C Power Connector for the Rackmount2 Enclosure

The PulseBlasterUSB Rackmount2 is powered through the USB-C connector at the front panel labeled POWER. The power source must be capable of providing 5V/3A, otherwise the board does not turn on. This connector does not have any data capabilities.

Status and Hardware Pins

Status Pins Description

Stopped – Driven high when the PulseBlasterUSB device has encountered a STOP Op Code during program execution and has therefore entered a stopped state. The device must be reset before it can start running again.

Reset – This signal is driven high when the device is in a RESET state.

Running – Driven high while the PulseBlasterUSB device is executing a program and low when the device enters a reset state or idle state.

Waiting – Driven high when the PulseBlasterUSB device has encountered a WAIT Op Code and is waiting for the next trigger in order to resume operation (trigger may be hardware or software generated).

Hardware Reset

The hardware reset signal is pulled to high voltage (3.3V) on the board and can be triggered by a low pulse (or shorting to ground). When a falling edge is detected (e.g., by shorting the pin to ground) the program execution will halt and the controller will reset itself back to the beginning of the program. Program execution can be resumed by either a software start command (pb_start()) or by a hardware trigger.

NOTE: *The PulseBlasterUSB 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 PulseBlasterUSB.***

NOTE: *Output flags hold the value in the state the board is reset in.*

Hardware Trigger

The hardware trigger is pulled to high voltage (3.3V) on the board and can be triggered by a low pulse (or shorting to ground). When a falling edge is detected (e.g., shorting the pin to ground) and the program is idle, code execution is triggered. If the program is idle due to a WAIT Op Code, then the HW_Trigger will cause the program to continue to the next instruction. If the program is idle due to a STOP Op Code or a HW_Reset signal, then the HW_Trigger starts execution from the beginning of the program. When using the STOP Op Code, a HW_Reset or software reset (pb_reset()) needs to be applied prior to the HW_Trigger.

NOTE: *The PulseBlasterUSB 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 PulseBlasterUSB.***

Figure 13, below, shows an example of the HW_Trigger signal with a latency of 80 ns. 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 10 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. 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.

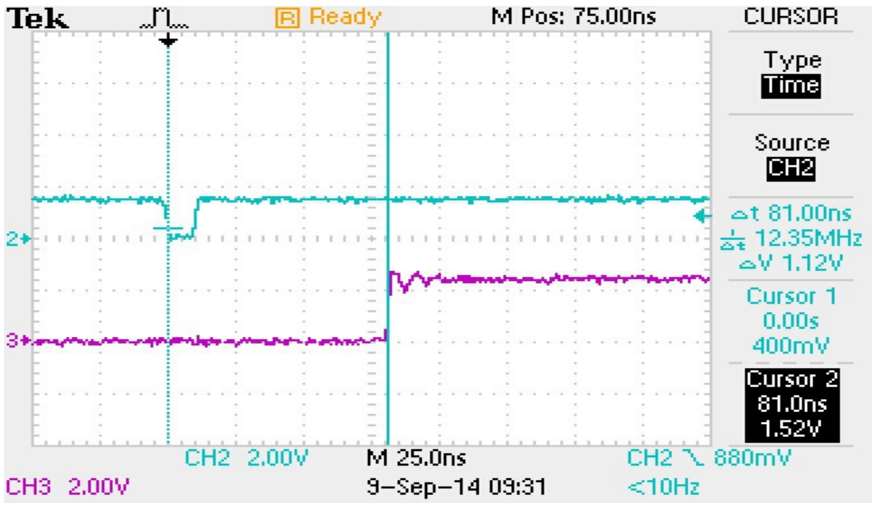
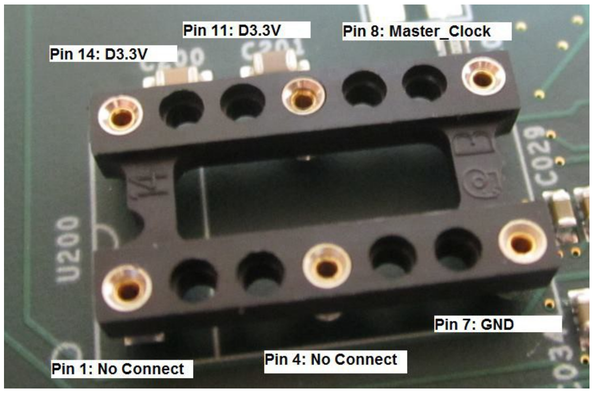


Figure 13: Demonstration of HW_Trigger signal with a latency of 80 ns. The blue shows the HW_Trigger signal, the pink shows one of the output flags. **Caution: applying voltages to the input pins that are greater than 3.3V or less than 0V will damage the board.**

Clock Oscillator Header

The PulseBlasterUSB 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 PulseBlasterUSB 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 LVTTL 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 LVTTL signal must be a positive-only signal, any negative voltage will damage the programmable-logic chip.



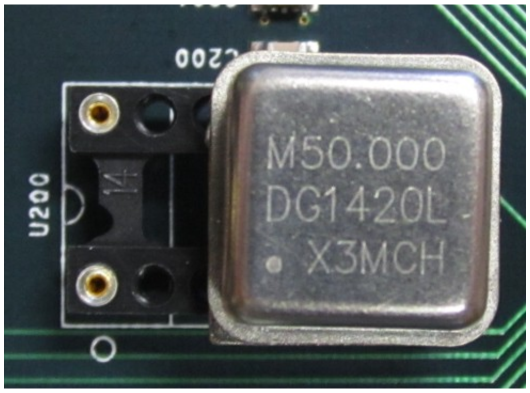


Figure 14: Both the bare header socket and the installed clock module are shown above. Please note the proper orientation of the 50 MHz clock.

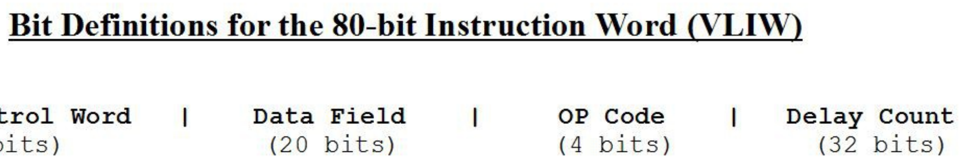
Appendix I. Instruction Set Architecture

Instruction Set Architecture of the PulseBlaster Processor

Machine-Word Definition

The PulseBlaster pulse timing and control processor implements an 80-bit-wide Very Long Instruction Word (VLIW) architecture. The VLIW memory words have specific bits/fields dedicated to specific purposes, and every word should be viewed as a single instruction of the micro-controller. The maximum number of instructions that can be loaded into on-board memory varies by model.

The execution time of instructions can be varied and is controlled by one of the fields of the instruction word – the shortest execution time is six clock cycles of the reference clock oscillator and the longest is 2^32 - 1 clock cycles (for standard instructions). All instructions have the same format and bit length, and all bit fields must be filled. Figure 15 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   |   OP Code   |   Delay Count
    (24 bits)              (20 bits)        (4 bits)       (32 bits)
```

Figure 15: Bit definitions of the 80-bit instruction/memory word.

Breakdown of 80-bit Instruction Word

The 80-bit instruction word is broken up into 4 sections:

1. Output Pattern and Control Word - 24 bits.
2. Data Field - 20 bits.
3. Op Code - 4 bits.
4. Delay Count - 32 bits.

Output Pattern and Control Word

The first 24 bits of the 80-bit machine word of the PulseBlaster processor are the output bits. The 24 output bits retain their value throughout the entire duration of the instruction represented by the machine word.

Data Field and Op Code

Table 5 (below) describes the available operational codes (referred to as Op Codes) and their associated data fields. Note that the data field's function is dependent on the Op Codes.

Op Code #	Instruction	Data Field	Function
0	CONTINUE	Ignored	Program execution continues to next instruction
1	STOP	Ignored	Stop execution of program. Aborts the operation of the micro-controller with no control of output states.
2	LOOP	Number of desired loops. This value must be greater than or equal to 1.	Specifies the beginning point of a loop. The Data Field is used to specify number of loops. Execution will continue to next instruction.
3	END_LOOP	Address of beginning of loop	Specifies the 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 the beginning of the desired subroutine
5	RTS	Ignored	Return from subroutine. Execution continues with the instruction following the previous JSR.
6	BRANCH	Address of next instruction	Program execution continues with the instruction at the specified address
7	LONG_DELAY	Desired multiplication factor for the “Delay Count” field. Must be greater than or equal to 2.	Used to execute very long instructions/ intervals. Data Field specifies a multiplier of the Delay Field. Execution continues to next instruction.
8	WAIT	Ignored	Program execution stops and waits for hardware trigger. Execution continues to next instruction after receipt of trigger. The exit latency is equal to the delay value entered in the WAIT instruction line plus a fixed delay of 6 clock cycles. The WAIT Op Code may not be used by the first instruction in memory.

Table 5: Op Code and Data Field Description.

The execution time of instructions is determined by the content of the Delay field (and the Data field, when applicable), and the output state remains constant during the entire instruction.

NOTE : The following two exceptions apply:

(1) The processor will ignore the delay field of instructions containing the STOP command. Upon encountering an instruction containing the STOP command, the processor will halt execution immediately rather than after the delay specified in the Delay Count field.

(2) The WAIT Op Code may not be used by the first instruction in memory. If you wish to have the processor wait for an external trigger at the beginning of your pulse routine, we suggest that you use two instructions: one with the CONTINUE Op Code (and with a very short delay) followed by an instruction which uses the WAIT Op Code.

NOTE : The behavior of the output flags after a STOP command is firmware dependent. If a different behavior is desired, please contact SpinCore Technologies, Inc. for alternate firmware options.

Delay Count

The value of the Delay Count field (a 32-bit value) determines the length of time that the current instruction should be executed. The number in this field represents the number of periods of the master clock oscillator that will occur before the execution of the next instruction begins.

Standard time units should be entered into the Delay Field when using the SpinAPI package or the PulseBlaster Interpreter. The allowed units are *ns* (nanoseconds), *us* (microseconds), and *ms* (milliseconds). SpinAPI and the PulseBlaster Interpreter will automatically convert these values into the appropriate number of clock cycles when the PulseBlasterUSB is programmed.

About SpinAPI

SpinAPI is a control library which allows programs to be written that can communicate with the PulseBlasterUSB 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 SpinAPI Driver and Control Library” 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 for the PulseBlasterUSB series of boards mentioned in this Manual.

Firmware Design	Board	Clock Speed (MHz)	Number of Output Bits	Memory Depth (words)	Minimum Instruction Duration (Clock Cycle)
29-1	SP50, SP59	100	24	4k	6
29-2	SP50, SP59	100	12	4k	6
29-3	SP59	100	24	8k	6
32-1	SP55	100	24	4k	6
32-3	SP55	100	24	8k	6

Table 6: Firmware designs available for the PulseBlasterUSB series mentioned in this Manual.

Related Products and Accessories

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

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

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

4. SP53, SpinCore MMCX Adapter Board (Figure 16) – This adapter board allows easy access to the individual bits of the SP50A and SP59 PulseBlaster boards. For ordering information, please visit http://spincore.com/products/Adapters/ or contact SpinCore at http://www.spincore.com/contact.shtml.

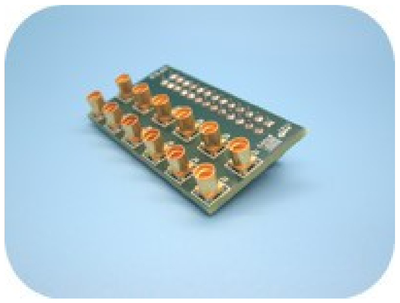


Figure 16: SP53, MMCX Adapter Board allows easy access to individual bits.

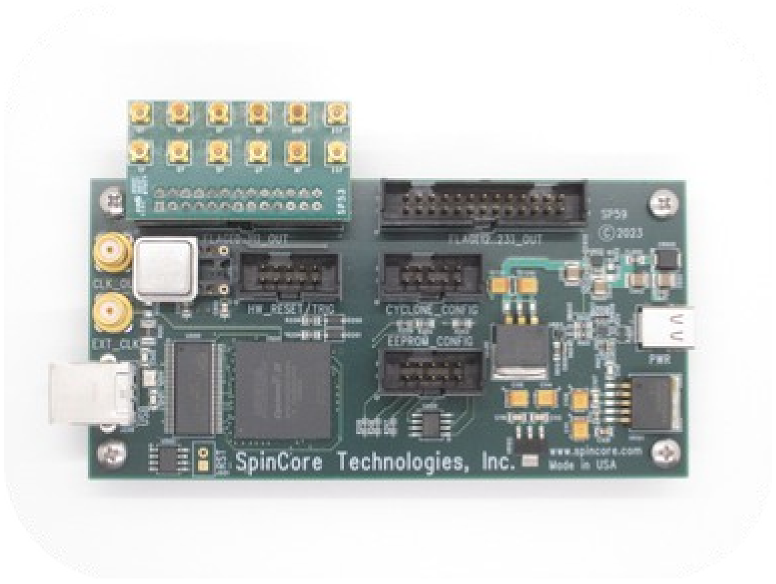


Figure 17: SP53, MMCX Adapter Board orientation and configuration with PulseBlasterUSB SP59 Board.

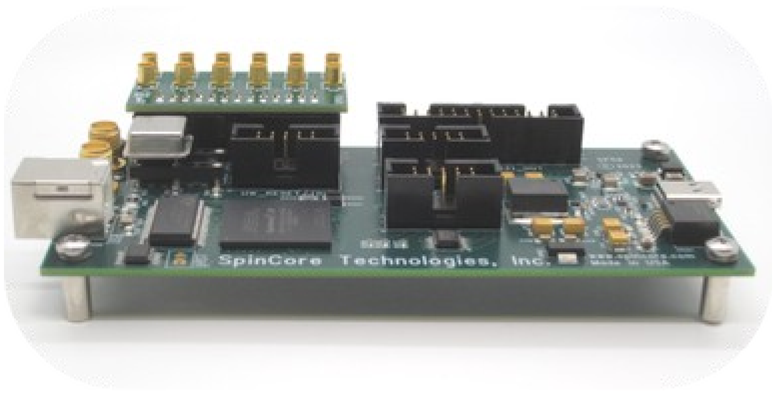


Figure 18: SP53, MMCX Adapter Board orientation and configuration with PulseBlasterUSB SP59 Board from a different angle.

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 Form:	http://spincore.com/contact.shtml

Document Information

Revision history of this document is maintained at SpinCore.