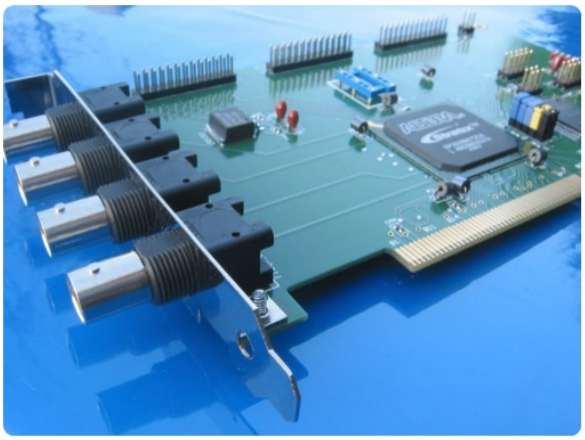
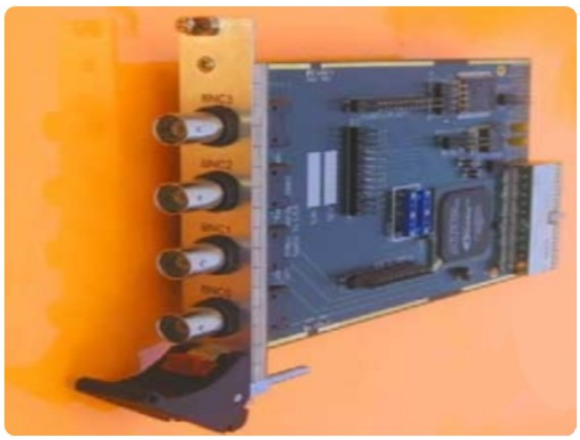
SpinCore Technologies, Inc.

PulseBlasterESR-PRO-II™

PCI and CompactPCI Boards

Owner's Manual





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

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

We appreciate your business!

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

Table of Contents

I. Introduction

Product Overview

The PulseBlasterESR-PRO-II device is a high speed, multichannel pulse/pattern generator. The PulseBlasterESR-PRO-II features a new control unit that allows for functionality not found in the PulseBlasterESR-PRO. This device can generate a unique output for each channel every clock period.

There is currently one PulseBlasterESR-PRO-II model offered by SpinCore Technologies, Inc, the 250 MHz model. The PulseBlasterESR-PRO-II 250 MHz model features 24 output channels with 8k words.

Pulse resolution is one clock period. This allows the device to create pulse sequences with durations or gaps as short as one clock period. Pulse sequences attributes can be configured in any combination of the one clock period resolution (see Figure 1 for example pulse sequences).

Figure 1 below shows an example timing diagram of arbitrary pulse sequences utilizing four output channels. This example only shows four channels, but up to 24 channels are supported. Pulse sequences can have duration as short as one clock period (see beginning of the pulse sequence in Channel 3), or any length supported by the resolution (e.g. 2 clock periods, 3 clock periods, etc.). Using the internal 250 MHz clock signal, pulse resolution is 4 ns. If additional performance is needed, please contact SpinCore Technologies, Inc.

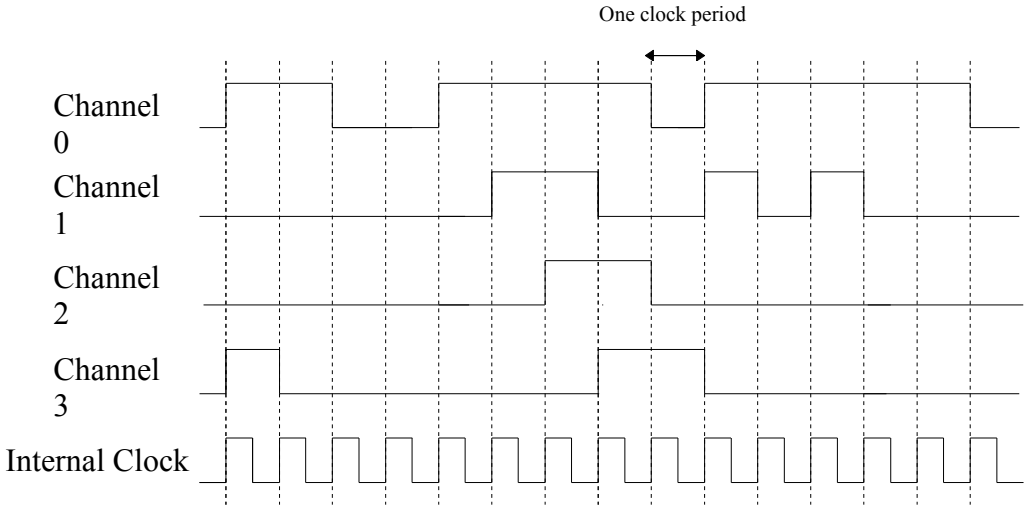


Figure 1: Sample timing diagram for PulseBlasterESR-PRO-II.

II. Product Description and Specifications

Device Architecture

Figure 2 presents the general architecture of the PulseBlasterESR-PRO-II system. The major building blocks are the SRAM memory, the Control Unit, and the Phase-locked Loop (PLL). The entire logic design, including the SRAM memory, is contained on a single FPGA chip (System-on-a-Chip design).

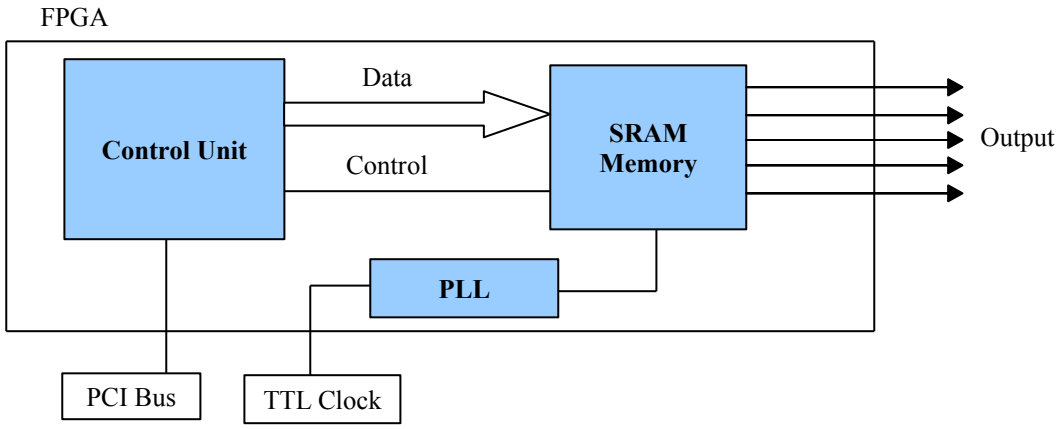


Figure 2: General PulseBlasterESR-PRO-II board architecture. This is all implemented on a single FPGA, making this device a System-on-a-Chip design. The clock oscillator signal is derived from an on-chip PLL circuit typically using a 50 MHz on-board reference clock.

Operation

The PulseBlasterESR-PRO-II uses a control unit and memory to send pulses to the output lines. The SRAM memory for the 250MHz model can contain up to 8k output words for the 24-channel model. Each channel can be programmed individually (see the example C program in Section V). Not all memory locations need to be be programmed. The pulse sequence can be varied as desired as long as it fits inside the device's capabilities.

When the PulseBlasterESR-PRO-II is programmed and triggered, it will step through each memory location and output the desired pulse sequence. Once the last programmed memory location has been reached, the device will loop back to the start of the pulse sequence.

Output signals

The PulseBlasterESR-PRO-II allows up to 24 output channels to be configured. All output signals are routed to the pins of the IDC on-board connector. Additionally, the first 4 output signals are routed to four bracket mounted BNC connectors (see Section III for more information). The BNC connector outputs have impedance matched to 50 ohms.

The output channels comply with the 3.3V TTL-level standard, and are capable of delivering ±25 mA per channel. If more output current is necessary, please contact SpinCore Technologies, Inc.

Timing characteristics

The PulseBlasterESR-PRO-II contains an internal (on-board) 50 MHz crystal oscillator. This input frequency is then internally multiplied by five yielding an internal (on-chip) frequency of 250 MHz. The internal memory allows for pulse attributes to last up to 32.768 us in duration. The new control unit allows pulse sequence attributes to be as small as one clock period, so as short as 4.0 ns. If shorter pulses are needed, please contact SpinCore Technologies, Inc.

External triggering

PulseBlasterESR-PRO-II can be triggered externally via a dedicated hardware line. The latency of the external trigger is one clock period. The hardware trigger is “active low” (triggered when external trigger pin is grounded). The hardware trigger and hardware reset pins are pulled to 3.3 V via a 10 kΩ resistor.

Summary

PulseBlasterESR-PRO-II is a versatile, multichannel, high-performance pulse/pattern TTL signal generator. It can operate at speeds of up to 250 MHz, and is capable of generating pulse sequences with attributes as short as 4.0 ns, with durations lasting up to 32.768 us. The device can output up to 8k unique outputs. Its high-current output logic bits are independently controlled with an output voltage of 3.3 V (unterminated).

Specifications

TTL Specifications

- 24 individually controlled digital output lines (TTL levels, 3.3 V logical “one” when unterminated).
- 4 bracket mounted BNC connectors, impedance matched to 50 ohm.
- variable pulses/delays for every TTL line.
- 25 mA output current per TTL line.

Pulse Parameters

- 4.0 ns resolution.
- 4.0 ns pulse sequence attributes.
- Shortest pulse duration/gap: 4.0 ns.
- Longest pulse duration/gap: 32.768 us 24-channel model.
- Up to 64k different output words with 8-channel model, or up to 10k different words with 24-channel model.
- Software and hardware trigger (TTL levels) and software reset.

III. Installation

Installing the PulseBlasterESR-PRO-II

When installing or uninstalling the PulseBlasterESR-PRO-II, always have it disconnected from the computer. 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-II 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, insert the PulseBlasterESR-PRO-II card into an available PCI slot and fasten the PC bracket securely with a screw.

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

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

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

IV. Connecting to the PulseBlasterESR-PRO-II

Connector Information

There are three main connector banks on the PulseBlasterESR-PRO-II board: the BNC headers, the IDC headers, and the Trigger/Reset header.

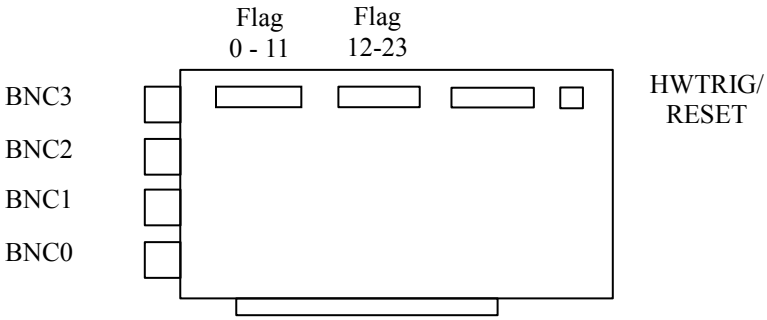


Figure 3: Connector Locations.

BNC Connectors

The four BNC connectors provide access to the least significant four bits of the flag word. On PCI boards, Bit 3 is connected to the output farthest from the PCI connector, and Bit 0 is connected to the connector closest to the PCI.

If using a high input impedance oscilloscope to monitor the PulseBlasterESR-PRO-II'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 4 and 5 below). When using an oscilloscope with an adjustable bandwidth, set the bandwidth to as large as possible. Failure to do so may yield inaccurate readouts on the oscilloscope.

Figure 4: Left: BNC T-Adapter and
Right: BNC (M) 50 Ohm resistor.

Figure 5: BNC T-Adapter on oscilloscope with coaxial transmission line connected on the left and BNC 50 Ohm resistor connected on the right, to terminate the line.

IDC Headers

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

Figure 6: IDC header pin-out

There are three IDC headers on the PulseBlasterESR-PRO-II, which provide access to the digital outputs.

This design has 24 output bits, and only the first two IDC headers are used. The signal is carried on pins 1-13 of each IDC header, although pin 13 is unused. The two headers are labeled Flag0..11_Out and Flag12..23_Out. On each IDC header, the top row of pins (14-26) are grounds, and signals are carried on bottom pins 1-13 of each header.

Each pin on an IDC header corresponds to a bit in the flag field of an instruction. The association between bits and pins is shown in the table on the next page.

Pin Assignments			
Pin#	Flag0..11	Flag12..23	Flag24..35
1	Bit 0	(Bit 12)	Unused
2	Bit 1	(Bit 13)	Unused
3	Bit 2	(Bit 14)	Unused
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 1: IDC connector pin-outs.

HWTRIG/RESET Header

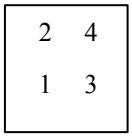


Figure 7: HWTRIG/RESET Header pin out.

This is an input connector for hardware triggering (HW_Trigger) and resetting (HW_Reset). Pins 1 and 2 are the reset and trigger inputs, respectively, and pins 3 and 4 are grounds. Both inputs are pulled high by an on board 10 kΩ pull-up resistor.

HW_Trigger(pin 2) When this input is set to logical 0 (for example by shorting it with pin 4), a hardware trigger is produced. This has the same effects as issuing a trigger through software, although the hardware trigger is faster, since there are no software latencies involved.

HW_Reset (pin 1) HW_Reset is not utilized in this design, please contact SpinCore Technologies, Inc. if you would like to have this functionality enabled.

V. Programming the PulseBlasterESR-PRO-II

The PulseBlasterESR-PRO-II board is easily programmed using C/C++ with the SpinAPI software. The PulseBlasterESR-PRO-II is also programmable using any interface that lets you utilize a C API package such as SpinAPI.

C/C++ Programming

The most dynamic and flexible way to program the PulseBlasterESR-PRO-II board is with C/C++ using the SpinAPI package. Coding in C/C++ allows you to better utilize the interrupt aspects of the board, and provides a dynamic and flexible method of programming. With the pre-configured compiler, changing one of our example programs and recompiling the executable file for use with your PulseBlasterESR-PRO-II board is as easy as clicking “Rebuild All” (see Figure 8 below). You can get this compiler on our website at

http://www.spincore.com/CD/Setup/SpinCore_SpinAPI_Tools_2007_07_11.exe

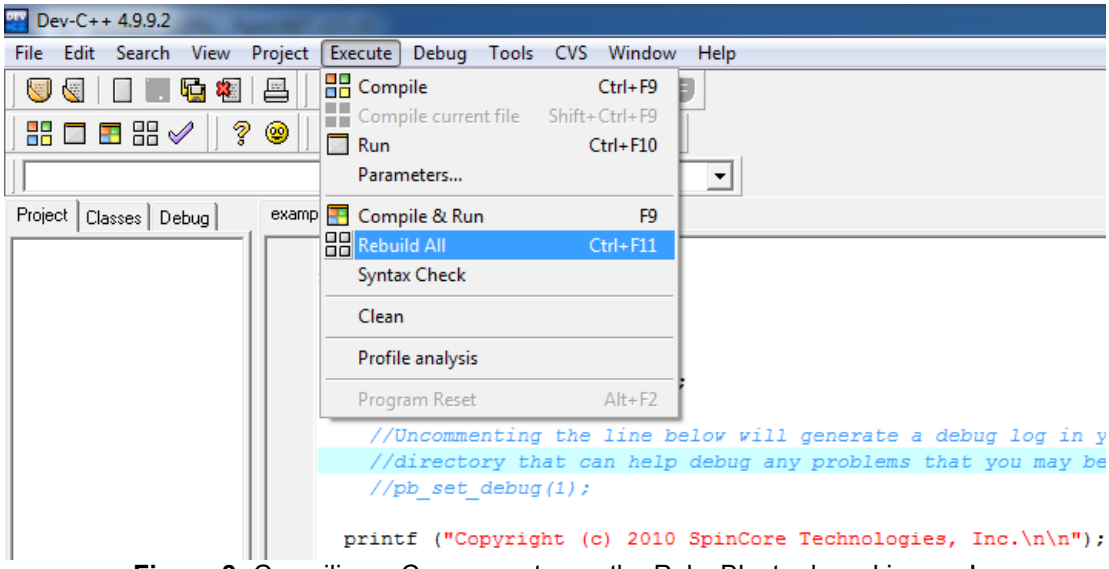


Figure 8: Compiling a C program to run the PulseBlaster board is easy!

Making changes to an example program requires understanding of only a few lines of code. The most important is the following line from example1_8bit.c or the line after that from example3_24bit.c found in your SpinCore directory:

```
pb_inst_hs8("11111111",20.0*us);
```

or

```
pb_inst_hs24("111111111111111111111111",20.0*us);
```

Each of these lines of code provide a high pulse on all output bits that will move on to the next instruction after 20 us. The first line is for the version of the board with 8 output channels, and the second line is for the version with 24 output channels. For more information about the example programs and how to modify them to fit your needs, please consult the appendix at the end of this manual.

VI. Appendix: C Programming with SpinAPI

About SpinAPI

SpinAPI is a control library which allows programs to be written to communicate with the PulseBlaster 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. A reference document for the API is available online at:

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

Using C Functions to Program the PulseBlasterESR-PRO-II

A series of functions have been written to control the board and facilitate the construction of pulses. 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), the header file (spinapi.h), must be appropriately linked. SpinCore offers a pre-configured compiler package (SpinAPI Tools) that is already set up for easy compilation of new pulse programs. To download SpinCore's pre-configured compiler package see the link below. If you wish to use your own compiler, you may look at our SpinAPI compilation instructions document also available at the link below:

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

Listed below are the necessary functions that you will need to use to program your PulseBlasterESR-PRO-II. Be sure to include spinapi.h in your source code in order to use these functions. Note: these functions are also documented in the SpinAPI Reference Documents available at the previously mentioned URL.

```
int pb_init();
```

Initializes PulseBlaster board. Needs to be called before calling any functions using the PulseBlaster. Returns a negative number on an error or 0 on success.

```
int pb_close();
```

Releases PulseBlaster board. Needs to be called as last command in pulse program. Returns a negative number on an error or 0 on success.

```
int pb_outp(unsigned int address, char data)
```

This function is used to control the board. The **char data** is set to zero for all instructions except for the last instruction in which **char data** is the byte to be sent. The following chart explains the values for **unsigned int address**:

unsigned int address	Function
0	Resets the board
1	Triggers execution of the board
4	Resets the memory counter
6	Output one byte to PCI bus

Table 2: Addresses for pb_outp() function.

```
int pb_inst_hs8(char* Flags, double length)
```

This function is intended for 8 channel PulseBlasterESR-PRO-II boards only. The first argument, **char* Flags** is an 8 bit string of ones and zeros (with the highest numbered channel on the left and decreasing to channel 2, channel 1, and channel 0). The second argument, **double length**, is the desired instruction length in nanoseconds. This function returns a negative number upon error and sets spinerr to a description of the error. The number of clock periods used for the instruction is returned on success.

```
int pb_inst_hs24(char* Flags, double length)
```

This function is intended for 24 channel PulseBlasterESR-PRO-II boards only. Similar to **pb_inst_hs8**, but uses a 24-bit string instead of an 8-bit. Other properties of the function are the same as **pb_inst_hs8.**

Included SpinAPI Programs

Included in the PBESR-PRO-II directory of the SpinAPI package (default installation location: C:\Program Files\SpinCore\Examples\PulseBlasterESR-PRO-II) are several example programs to test your board, as well as to create a template for users to customize their own pulse programs. Also included for convenience are two executable programs called “reset.exe” and “trigger.exe.” These programs can be used for software triggering and resetting of your pulse program. The source code for these programs are included, so you may add the code to your custom programs as necessary. Examples 1-3 are for earlier versions of the PulseBlasterESR-PRO-II board with 8 channels.

example1_8bit

This example program tests the maximum pulse sequence length. This program will test the entire memory.

NOTE: It is important to terminate all signals properly (i.e. with a 50 Ohm terminating resistor at the end of your cable).

example2_8bit

This example program demonstrates the ability to change outputs every clock period.

NOTE: It is important to terminate all signals properly (i.e. with a 50 Ohm terminating resistor at the end of your cable).

example3_8bit

An example program further demonstrating the device.

NOTE: It is important to terminate all signals properly (i.e. with a 50 Ohm terminating resistor at the end of your cable).

example4_24bit

This example program tests the maximum pulse sequence length. This program will test the entire memory.

NOTE: It is important to terminate all signals properly (i.e. with a 50 Ohm terminating resistor at the end of your cable).

example5_24bit

This example program demonstrates the ability to change outputs every clock period.

NOTE: It is important to terminate all signals properly (i.e. with a 50 Ohm terminating resistor at the end of your cable).

reset

This program will generate a software reset. The device will cease outputting data, but the device will keep the program. The board can be restarted using “trigger”

trigger

This program will generate a software trigger. The device will start outputting data stored in its memory counter. The device can be stopped using reset.c.

Example Use of C Functions

This example code will generate the pulses shown in the timing diagram on page 5 using a 250MHz model. An equivalent program is included with the SpinAPI package and is named “example4_24bit.”

```
/* PulseBlasterESR-PRO-II high speed memory output Example program
 * © 2009 SpinCore Technologies Inc.
 *        http://www.spincore.com
 *
 * This sample code is available to SpinCore customers to use and modify for any
 *   purpose, including commercial use.
 *
 * For more information visit:
 *     http://www.spincore.com/products/PulseBlasterESR-PRO-II/
 *
 * This program loads a custom pulse sequence onto the PulseBlasterESR-PRO-II
 *   board and then triggers it.  This is a high-speed output function that will
 *   not work for any other PulseBlaster or RadioProcessor board.
 */

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

#include "spinapi.h"

#define CLOCK 250.0

int detect_boards();
int select_board(int numBoards);

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

      printf("Using SpinAPI library version %s\n", pb_get_version());

      if((numBoards = detect_boards()) > 1)
      {
         /*If there is more than one board in the system, have the user
         specify.*/
           select_board(numBoards); /*Request the board number to use from the
         user*/
      }

      if (pb_init () != 0)
      {
         printf ("Error initializing board: %s\n", pb_get_error ());
         system("pause");
         return -1;
      }

      //setup the clock frequency
      pb_core_clock(CLOCK);

      //The following two lines are used for PBESR-Pro-II designs in place of
```

```
        //   pb_start_programming(...) and pb_stop_programming().
        pb_outp(0,0); //Reset the board
        pb_outp(4,0); //Reset the memory counter

        //*** Pulse program loading begins here ***
        /* Max length of pulse sequence is 32.768 us
        *  The leftmost bit is Channel 23, followed by Channel 22,
        *    Channel 21, ... , Channel 1, and the rightmost bit is Channel 0.
                                                */
        pb_inst_hs24("00000000000000000000001",4.768*us);
        pb_inst_hs24("00000000000000000000010",2.0*us);
        pb_inst_hs24("00000000000000000000100",2.0*us);
        pb_inst_hs24("00000000000000000000010",2.0*us);
        pb_inst_hs24("00000000000000000000100",2.0*us);
        pb_inst_hs24("00000000000000000000010",2.0*us);
        pb_inst_hs24("00000000000000000000100",2.0*us);
        pb_inst_hs24("00000000000000000000010",2.0*us);
        pb_inst_hs24("00000000000000000000100",2.0*us);

        //Nothing for 12 us
        pb_inst_hs24("00000000",12.0*us);

        //Trigger the board
        printf("Beginning pulse generation\n");
        pb_outp(1,0);

        //Signal the end of communication
        pb_close();
        system("PAUSE");

        return 0;
}
```

Related Products and Accessories

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

2. PulseBlasterESR MultiCore 8M Series – A 500 MHz version of the PulseBlaster containing multiple PulseBlaster processor cores. Designed to run independent programs on each core, up to 8388608 (8M) instructions in total, in parallel, while maintaining precise timing synchronization between the cores. For more information, please visit http://www.spincore.com/products/PulseBlasterESR_MultiCore/PulseBlasterESR_MultiCore.shtml

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

Contact Information

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

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

Document Information

Revision history available at SpinCore.