



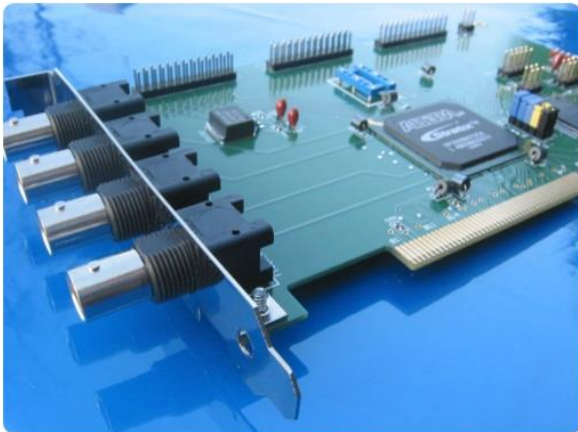
---

# **PulseBlasterESR-PRO-I™**

## **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.**

---

© 2000-2012 SpinCore Technologies, Inc. All rights reserved.

SpinCore Technologies, Inc. reserves the right to make changes to the product(s) or information herein without notice.

PulseBlasterESR-PRO-II™, PulseBlasterESR™, PulseBlaster™, SpinCore, and the SpinCore Technologies, Inc. logos are trademarks of SpinCore Technologies, Inc. All other trademarks are the property of their respective owners.

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

<b>I. Introduction.....</b>	<b>5</b>
<b>Product Overview.....</b>	<b>5</b>
<b>II. Product Description and Specifications.....</b>	<b>6</b>
<b>Device Architecture .....</b>	<b>6</b>
<b>Operation.....</b>	<b>6</b>
<i>Output signals.....</i>	<i>6</i>
<i>Timing characteristics.....</i>	<i>7</i>
<i>External triggering.....</i>	<i>7</i>
<i>Summary.....</i>	<i>7</i>
<b>Specifications.....</b>	<b>7</b>
<i>TTL Specifications.....</i>	<i>7</i>
<i>Pulse Parameters .....</i>	<i>7</i>
<b>III. Installation.....</b>	<b>8</b>
<b>Installing the PulseBlasterESR-PRO-II.....</b>	<b>8</b>
<b>IV. Connecting to the PulseBlasterESR-PRO-II.....</b>	<b>9</b>
<b>Connector Information.....</b>	<b>9</b>
<i>BNC Connectors.....</i>	<i>9</i>
<i>IDC Headers.....</i>	<i>10</i>
<i>HWTRIG/RESET Header.....</i>	<i>11</i>
<b>V. Programming the PulseBlasterESR-PRO-II.....</b>	<b>12</b>
<b>C/C++ Programming.....</b>	<b>12</b>
<b>VI. Appendix: C Programming with SpinAPI.....</b>	<b>13</b>
<b>About SpinAPI.....</b>	<b>13</b>
<b>Using C Functions to Program the PulseBlasterESR-PRO-II.....</b>	<b>13</b>
<b>Included SpinAPI Programs.....</b>	<b>14</b>
<i>example1_8bit.....</i>	<i>14</i>
<i>example2_8bit.....</i>	<i>14</i>

# *PulseBlasterESR-PRO-II*

<a href="#"><u>example3_8bit.....</u></a>	<a href="#"><u>14</u></a>
<a href="#"><u>example4_24bit.....</u></a>	<a href="#"><u>15</u></a>
<a href="#"><u>example5_24bit.....</u></a>	<a href="#"><u>15</u></a>
<a href="#"><u>reset.....</u></a>	<a href="#"><u>15</u></a>
<a href="#"><u>trigger.....</u></a>	<a href="#"><u>15</u></a>
<a href="#"><u>Example Use of C Functions.....</u></a>	<a href="#"><u>16</u></a>

<a href="#"><u>Related Products and Accessories.....</u></a>	<a href="#"><u>18</u></a>
--	---------------------------

<a href="#"><u>Contact Information.....</u></a>	<a href="#"><u>18</u></a>
---	---------------------------

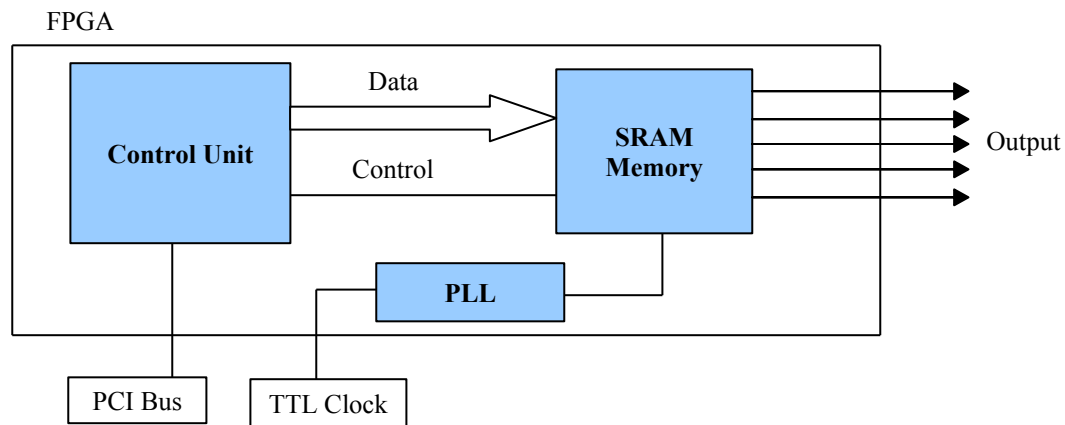
<a href="#"><u>Document Information.....</u></a>	<a href="#"><u>18</u></a>
--	---------------------------



## 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, making it a 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 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 are impedance matched to 50 ohms.

The output channels comply with the 3.3V TTL-level standard, and are capable of delivering  $\pm 25$  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 new control unit allows pulse sequence attributes to be as small as one clock period, which means pulse sequence attributes can be as small as 4.0 ns. If shorter pulses are needed, please contact SpinCore Technologies, Inc.

The internal memory allows for pulse attributes to last up to 32.768 us in duration.

## **External triggering**

PulseBlasterESR-PRO-II can be triggered externally via a dedicated hardware line. The latency of the external trigger is only 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

---

To install the board you must complete the following three steps:

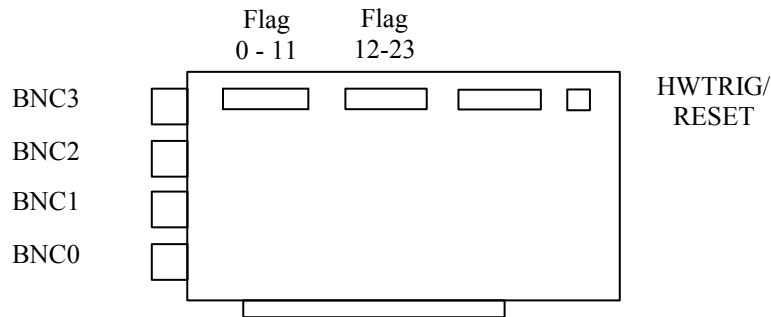
1. Download and install the latest SpinAPI software package on your computer. SpinAPI is available at: <http://www.spincore.com/support/spinapi/>
  - SpinAPI is a custom Application Programming Interface package developed by SpinCore Technologies, Inc. SpinAPI is designed to be used only with SpinCore Technologies, Inc. products. SpinAPI can be utilized using C/C++, or graphically (described in Section V).
  - SpinAPI contains device drivers, example programs, and the PulseBlaster Interpreter program (described in Section V). A shortcut to the location of these files is created automatically in the Windows Start Menu at “**Start>Programs>SpinCore**”. See the “readme.txt” file in this location for more information.
2. Shut down the computer, insert the PulseBlasterESR-PRO-II card into an available PCI slot and fasten the PC bracket securely with a screw.
3. Turn on the computer and follow the installation prompts.

Now you are ready to run the test programs provided in the SpinAPI package. We recommend running the example programs when you first receive a PulseBlaster device to verify that your PulseBlaster device is functional. Test programs for the PulseBlasterESR-PRO-II can be found in the Windows Start Menu at “Start>Programs>SpinCore>Examples>PBESR-PRO-II”.

## 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  $\Omega$  resistor with a 50  $\Omega$  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.

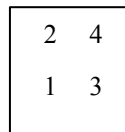
Since this design has 24 output bits, only the first two IDC headers are used. These headers are labeled Flag0..11\_Out and Flag12..23\_Out. On each IDC header, the top row of pins (14-26) are grounds, and the signals are carried on pins 1-13.

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 below.

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 MATLAB or 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](http://www.spincore.com/CD/Setup/SpinCore_SpinAPI_Tools_2007_07_11.exe)

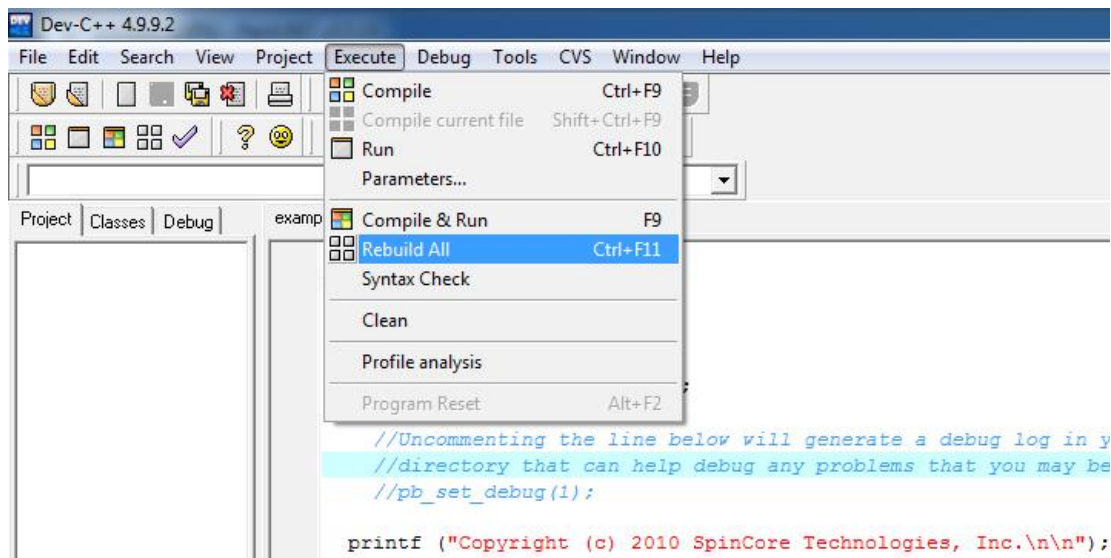


Figure 8: Compiling a C program to run the PulseBlaster board is easy!

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. However, virtually all programming languages and software environments (including software such as LabVIEW and MATLAB) provide mechanisms for accessing the functionality of standard libraries such as SpinAPI. A reference document for the API is available online at:

[http://www.spincore.com/CD/spinapi/spinapi\\_reference/](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**:

# *PulseBlasterESR-PRO-II*

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 executables 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.

## ***PulseBlasterESR-PRO-II***

**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
```

## *PulseBlasterESR-PRO-II*

```
// 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("00000000000000000000000000000001",4.768*us);
pb_inst_hs24("00000000000000000000000000000010",2.0*us);
pb_inst_hs24("000000000000000000000000000000100",2.0*us);
pb_inst_hs24("00000000000000000000000000000010",2.0*us);
pb_inst_hs24("000000000000000000000000000000100",2.0*us);
pb_inst_hs24("00000000000000000000000000000010",2.0*us);
pb_inst_hs24("000000000000000000000000000000100",2.0*us);
pb_inst_hs24("00000000000000000000000000000010",2.0*us);
pb_inst_hs24("000000000000000000000000000000100",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](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.