SpinCore Technologies, Inc.

PulseBlasterDDS™

Model DDS-II-300 USB

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 try 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 PulseBlasterDDS-II™ series of Intelligent Pattern and Waveform Generation systems from SpinCore Technologies, Inc., couple SpinCore's unique Intelligent Pattern Generation processor core, dubbed PulseBlaster™, with two Direct Digital Synthesis (DDS) units for use in system control and pulse generation.

The PulseBlaster's state-of-the-art timing processor core provides all the necessary timing control signals required for overall system control and pulse synchronization with 13.3 ns resolution. The additional DDS features allow the PulseBlasterDDS-II-300-AWG to provide not only digital (TTL) but also analog output signals from two independent RF output channels, ranging in frequency from 5 kHz to 100 MHz. This allows the PulseBlasterDDS-II-300-AWG to meet high-performance and high-precision complex excitation/stimuli needs of demanding users. The PulseBlasterDDS-II-300-AWG provides users the ability to control their systems through the generation of fully synchronized (digital and analog) excitation pulses from a small form factor USB system. The PulseBlasterDDS-II-300-AWG also contains user-programmable Arbitrary Waveform Generation (AWG) capabilities for modulating the shape of the output waveforms for both analog outputs. The PulseBlasterDDS-II-300-AWG is available in two standard options: a) the board-only option b) the single-bay standalone enclosure. If you require a specific enclosure design, please contact SpinCore directly.
Key features of the PulseBlasterDDS-II-300-AWG include:

- Four vectored interrupts with programmable Interrupt Service Routine (ISR) registers (three hardware pins and one software interrupt)
- Interrupt enable registers to disable or enable interrupts on-the-fly
- User programmable registers to specify which "NORMAL" vs. "IMMEDIATE" interrupts:
    - Normal Interrupt: Waits for the current instruction to complete, then performs the ISR and returns to the start of the next instruction
    - Immediate Interrupt: Stops the current instruction immediately, performs the ISR and then returns to the beginning of the next instruction
- Hardware and software interrupt triggers
- 124 bit instruction memory words
- 4K instruction memory words
- Programmable Pulse Program start address allowing the user to program the device once with multiple pulse programs, then pick the program entry point with a single write to the device
- Modifiable Pulse Programs while the core is running
- User programmable default return-to state for core flags when the core is not running, allowing for full control of flag outputs by the user when the core is STOPPED
- 1024 Frequency, 128 Phase, and 1024 Amplitude registers for each DDS output[1]
- All DDS registers are programmable while the PulseBlaster Core is running

[1] For all available Firmware designs, please refer to Table 9.

Core Architecture

Figure 1 presents the general architecture of the PulseBlasterDDS-II-300-AWG system. The two major building blocks of the system are the two independent DDS Cores and the Pulse Programming and Timing Processor Core (PulseBlaster Core). The DDS Cores contain a numerically controlled oscillator and have 1024 programmable frequency registers[2] that are under the pulse program control. Prior to gating, the DDS signal can be phase offset by one of the 128 programmable phase registers[2]. The waveform envelope and amplitude can be programmed by the user via the AWG table and digital attenuator. The PulseBlaster Core controls the timing of the gating pulses and provides the necessary control signals for the frequency, phase and amplitude registers. The DDS and PulseBlaster cores have been integrated onto a single silicon chip. High performance DAC chips and broadband output transformers complement the design. User control to the system is provided through the USB Programming Interface over the USB Bus. The USB Programming Interface has a 32-bit address bus and 32-bit data bus with auto address incrementing for easier user programming.

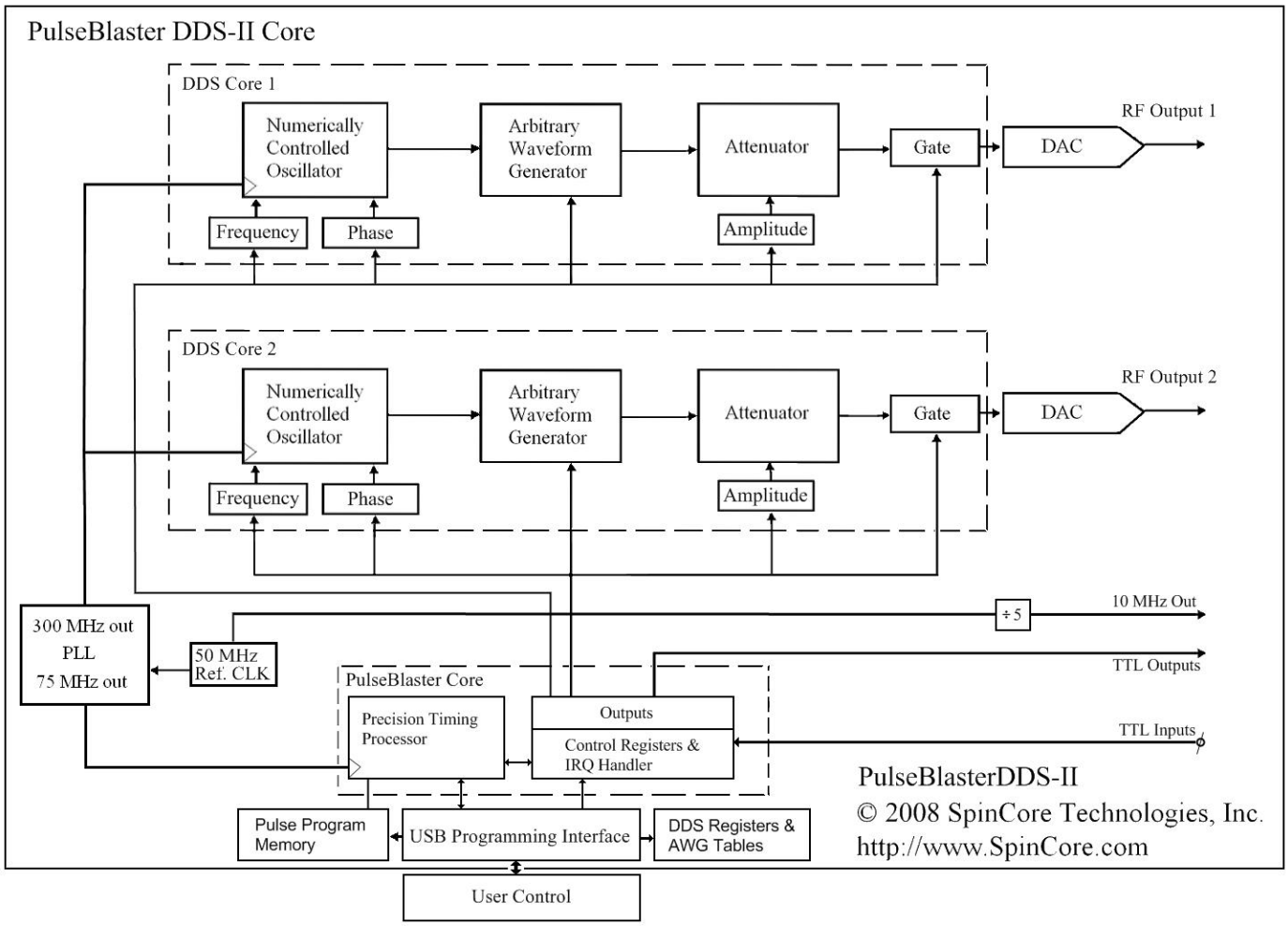

Figure 1: PulseBlasterDDS-II-300-AWG-300 System Core Architecture.

[2] Please refer to Table 9 for available Firmware designs.

Product Specifications

	Parameter	*Min*	*Typ*	*Max*	*Units*
Analog Output[1]	D/A sampling rate		300		MHz
	D/A sampling precision			14	bits
	Output voltage range (peak-peak)			1.2[2]	Vpp
	Phase resolution			0.09	deg.
	Frequency resolution		0.28[3]		Hz
Digital Output	Number of digital outputs[4]		4	12	
	Logical 1 output voltage		3.3[5]		V
	Logical 0 output voltage		0		V
	Output drive current			25	mA
	Rise/Fall time			< 1	ns
Digital Input (HW_Trig,HW_Reset, IRQ[2..0])	Logical 1 input voltage	0.7		3.3	V
	Logical 0 input voltage	0		0.7	V
Pulse Program	# of instruction words		4k[6]		words
	Pulse resolution		13.3		ns
	Instruction duration[7]	66.6 ns		693 days	

Table 1: PulseBlasterDDS-II-300-AWG Product Specifications.

Notes

1) A reconstructive filter should be used at the output of the DDS channels and the output must be properly terminated using a 50 Ohms terminating resistor. The output is AC coupled with a minimum frequency of 5 kHz. Both BNC and SMA versions of low-pass and band-pass filters are available for this purpose. More information on the significance of the reconstructive filtering can be found in Section VII.
2) Analog output voltage maximum is 1.2 Vpp on 50 Ohms.
3) Frequency resolution is .28 Hz when using a 50 MHz reference clock.
4) The number of digital outputs is firmware dependent. Please contact SpinCore for more details.
5) This is the value seen without using termination. When the line is terminated with 50 Ohms, the output voltage will be lower.
6) If you require more instruction words (up to 8k), please contact SpinCore directly.
7) To get instructions longer than 55 seconds, the “LONG_DELAY” instruction is needed. For more information see the “Pulse Programs” part of Section III.

II. System Installation

To install the PulseBlasterDDS-II-300-AWG system you must complete the following four steps:

1. Install the latest SpinAPI version and example programs found at: http://spincore.com/support/spinapi/. SpinAPI is a custom Application Programming Interface developed by SpinCore Technologies, Inc. for use with the PulseBlaster board. It can be utilized using C/C++ or graphically using the options in the next section below, Testing the PulseBlasterDDS-II-300-AWG. The API will also install the necessary drivers.
2. Connect the PulseBlasterDDS-II-300-AWG system using a USB 2.0 cable.
3. Apply power to the PulseBlasterDDS-II-300-AWG system.
4. 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 “PulseBlasterDDS-II” folder and run all .exe programs to test your PulseBlasterDDS-II.

Testing the PulseBlasterDDS-II-300-AWG

Once your PulseBlasterDDS-II-300-AWG system is installed properly, you are now ready to test the functionality of your device with the example programs included in the SpinAPI package. You can begin by running the compiled executable in the PBDDS-II folder to test the functionality of your system. Then you can modify the C source files as needed to start making your own programs. All of the sequences are easily verified using an oscilloscope which can be triggered using any of the TTL outputs.

The program `dds2_excite_test` tests the analog outputs. This program will produce a 10.0 MHz sine wave on both analog outputs which will turn on for 10 µs, off for 200 µs, and then repeat indefinitely.

The signal output can be stopped by using the `pb_stop.exe` program. The corresponding `pb_start.exe` will restart the PBDDS-II board. Reloading the board with new program will also stop the execution of any running program.

Next, `dds2_TTL_only` will test the TTL outputs. This program will produce a pulse pattern with all TTL Flag Bits high for 10 µs, low for 10 µs, and then repeat indefinitely.

To test the AWG feature of the board, use `dds2_awg`. This program will produce a 2.0 MHz signal that is modulated with a ramp envelope on the I_OUT1 output and produce a 10.0 MHz signal that is modulated with a three lobe 'sinc' waveform on the I_OUT2 output. Both channels are on for 10 µs, off for 2 ms and then the pattern is repeated.

III. Using the PulseBlasterDDS-II-300-AWG

The PulseBlasterDDS-II-300-AWG is a highly versatile excitation system, and as a result there are many possible approaches to program the device. However, most applications can be programmed following these basic steps:

1. Select the device (if using more than one device).
2. Initialize the device.
3. Load frequency, phase, and amplitude registers with the desired values.
4. Define the shape of the carrier for the Numerically Controlled Oscillator and define the shape of the modulator envelope for the Arbitrary Waveform Generator. Please refer to Figure 1 in section I. Introduction.
5. Load a pulse program which will control the timing of the experiment.
6. Enable any interrupts that will be utilized and program the ISR vector for the corresponding IRQ.
7. Trigger the pulse program. The experiment will then proceed autonomously.

These steps are described in detail below. For each of the steps, the relevant SpinAPI functions are listed which control the actions needed to perform that particular step. More details on the format for the API calls are given in the next section.

Selecting and Initializing the System

By default, the API uses the first PCI device (or the first USB device, if no PCI devices are connected). In order to select the PulseBlasterDDS-II-300-AWG device when other SpinCore Technologies, Inc. products are connected to your system, it is necessary to first call the `pb_select_board(..)` routine with the appropriate argument. The PCI devices are always enumerated first by the API, so any PCI devices connected to your system will be numbered first. To determine the number of your device when multiple SpinCore Technologies, Inc. USB devices are connected, please check the Device Manager for device numberings.

Before using any API functions to control your system, it is necessary to first initialize the device using the pb_init() routine. The pb_init() routine opens a file handle to the selected device allowing for the API to communicate with it.

Once the device has been initialized, it is necessary to set the clock frequency so that the proper counter values can be programmed. This can be accomplished using `pb_core_clock(..)`. The clock frequency is that of the PulseBlaster Core, not the DAC or external oscillator. For standard DDS-II-300 boards, this value is 75 MHz. See Code Example 1.

```
printf("Number of boards detected: %d\n", pb_count_boards());
pb_select_board(1); //Selects the second board detected (0 is the
first.)

if(pb_init() != 0) //Attempt to initialize the board.
{
        printf("There was an error initializing your board.\n");
        return -1;
}

pb_core_clock(75.0); //Program the clock frequency value.
```

Code Example 1: Selecting and initializing your device.

Frequency, Phase and Amplitude Registers

The PulseBlasterDDS-II-300-AWG contains two DDS channels. Each DDS channel drives a Digital-to-Analog Converter (DAC) that forms the Analog Output channel. The frequency, phase, and amplitude of each channel is

controlled by selecting values from a bank of on-board registers. Each DDS contains its own set of frequency, phase, and amplitude registers. These registers should be programmed with appropriate values after device initialization, but before triggering the device. In order to access the registers of a particular DDS, it is necessary to first select the register by calling the pb_select_dds(...) routine. Each pulse program instruction selects which register is used at any given time during an experiment. The number of available registers is given in the following table:

Register Bank	***Number of registers Firmware 14-2***	***Number of registers Firmware 14-3***
DDS0 Frequency	16	1024
DDS0 Phase	8	128
DDS0 Amplitude	4	1024
DDS1 Frequency	16	1024
DDS1 Phase	8	128
DDS1 Amplitude	4	1024

Table 2: DDS Register information.

```
pb_select_dds(0); //Select DDS0 (this is selected by default.)

pb_start_programming(FREQ_REGS); //Program DDS0's frequency
                                 //registers.
pb_set_freq(1.0*MHz); //Program Frequency Register 0.
pb_set_freq(2.0*MHz); //Program Frequency Register 1.
pb_stop_programming();

pb_start_programming(TX_PHASE_REGS); //Program DDS0's phase
registers.
        pb_set_phase(0.0);
pb_stop_programming();

pb_set_amp(1.0,0); //Program DDS0's amplitude registers.

pb_select_dds(1); //Select DDS1.

pb_start_programming(FREQ_REGS); //Program DDS1's frequency
registers.
pb_set_freq(1.0*MHz); //Program Frequency Register 0.
pb_set_freq(2.0*MHz); //Program Frequency Register 1.
pb_stop_programming();

pb_start_programming(TX_PHASE_REGS); //Program DDS1's phase
                                     //registers.
pb_set_phase(90.0);
pb_stop_programming();

pb_set_amp(1.0,0); //Program DDS1's amplitude registers.
```

Code Example 2: Programming the DDS Registers.

Sample Output

Figure 2, below, shows an example RF 70 MHz output pulse that was generated by the device. The data was captured using a Tektronix TDS224 oscilloscope. Notice the time base of 25 ns/division.

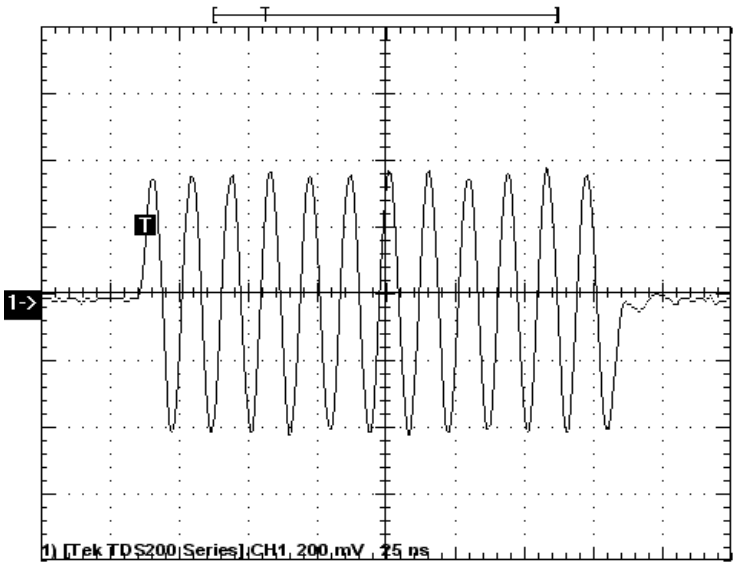

Figure 2: 185 ns 70 MHz RF/IF output pulse.

Figure 3, below, demonstrates the zero-latency phase-switching agility. In this figure, two short back-to-back pulses were recorded with a 180-degree phase offset and a 70 MHz carrier frequency (expanded view).

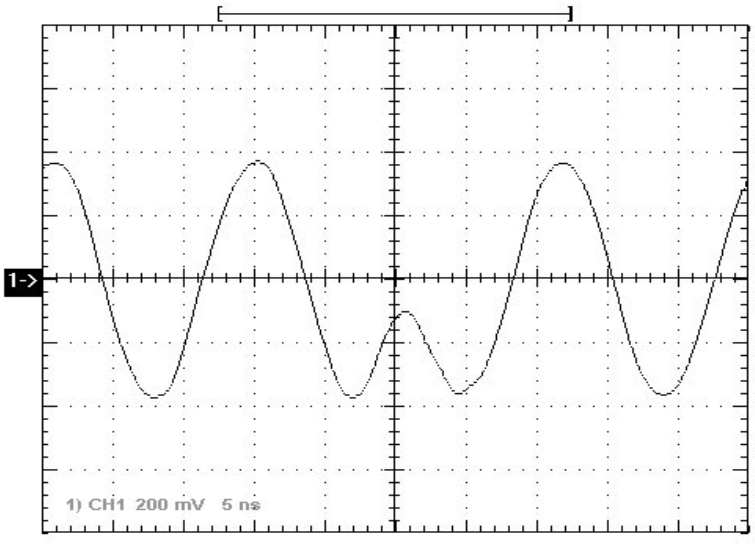

Figure 3: Two RF output pulses, back to back, with a 180 degree phase switch and 70 MHz RF frequency.

Figure 4, below, demonstrates the frequency-shift agility. In this figure, the frequency jumps from 20 MHz to 10 MHz with no latency.

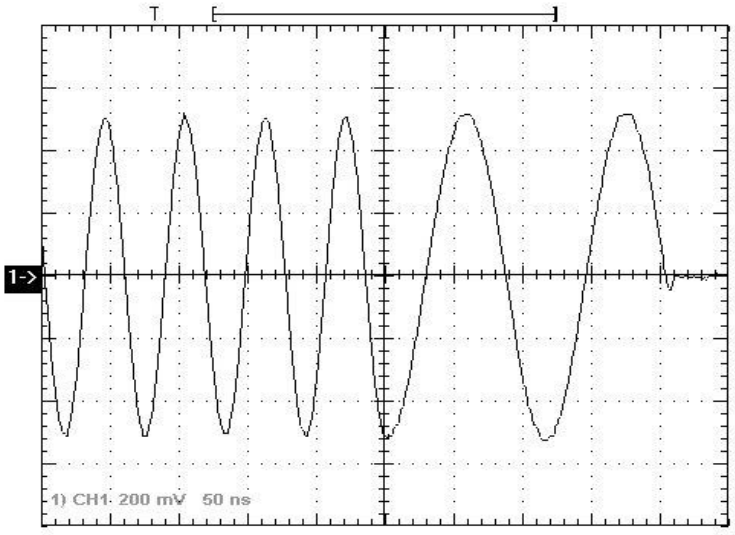

Figure 4: Frequency shift from 20 MHz to 10 MHz.

Shape and DDS Devices and Amplitude Registers

The AWG (Arbitrary Waveform Generator) system is flexible and can be programmed with a wide variety of parameters. The significant features of this system are:

- RF outputs can be shaped by an arbitrary waveform. (for example, a sinc waveform)
- RF outputs can be scaled by a constant value.
- The RF output itself can be set to waveforms other than a sine-wave (e.g., triangle wave, square wave, etc.).
- The shortest possible pulse is 66.6 ns, the longest is 693 days.

This allows, for example, the generation of soft pulses as shown in Figure 5.

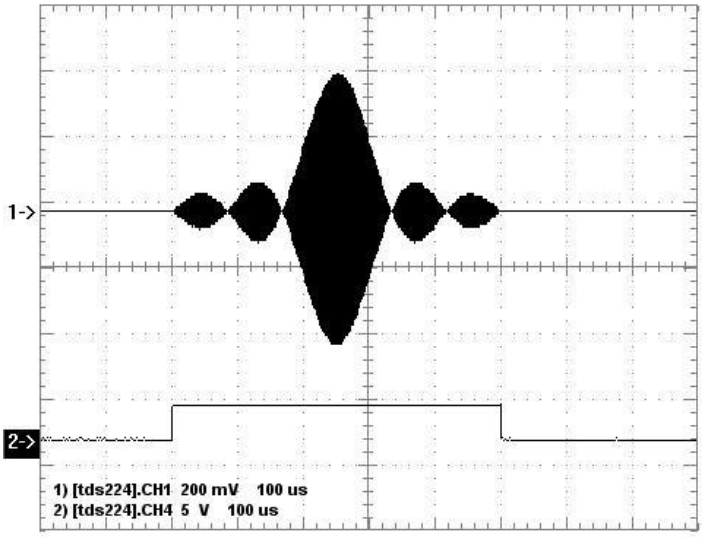

Figure 5: Sinc-shaped soft pulse. Pulse duration is 0.5 ms.

For example, as shown in Figure 6 and Figure 7 below, pulse sequences that have pulses with multiple different amplitudes can be generated. It is also possible to generate both hard and soft pulses in a single pulse program. The shape of the pulse is not limited to a sinc-shape; it is user-loadable with any arbitrary waveform.

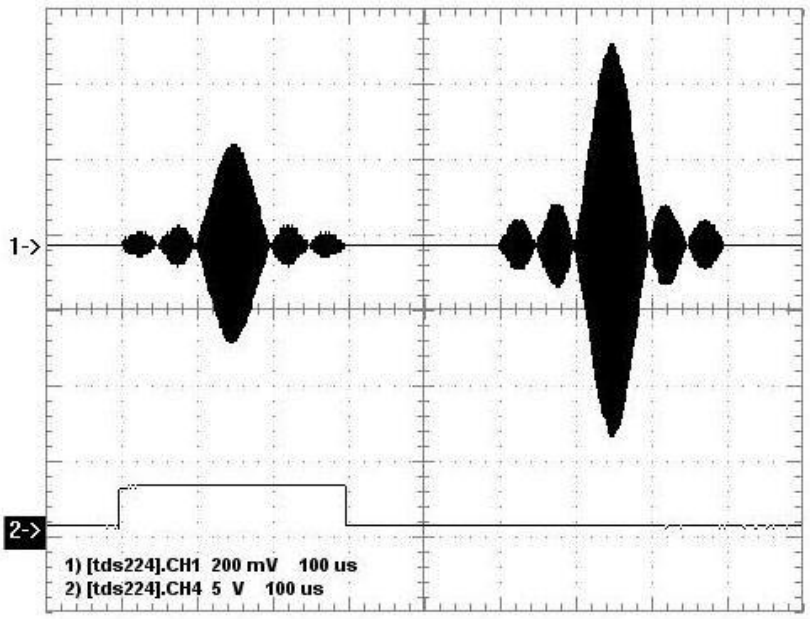

Figure 6: Combinations of RF pulses - variable amplitudes.

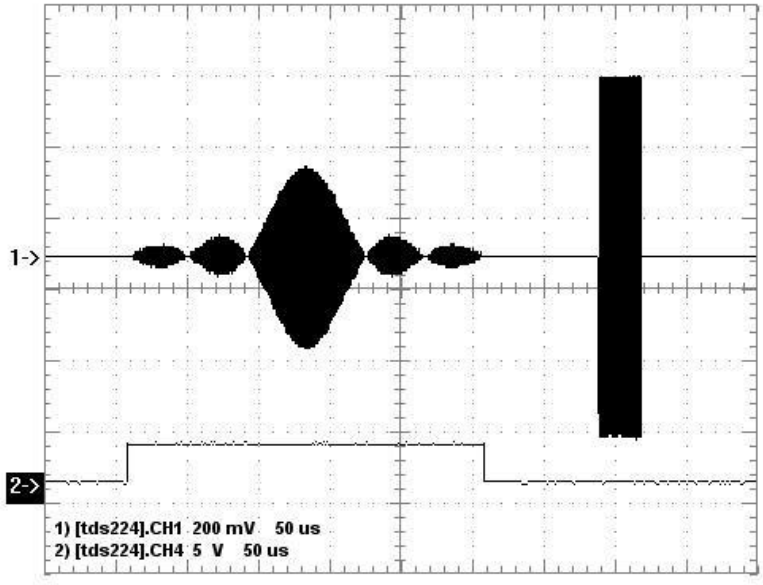

Figure 7: Combinations of RF pulses - soft- and hard-pulses in a sequence.

Example programs

An example program is included with SpinAPI in the PBDDS-II-300 directory to demonstrate how the AWG capabilities are used. The source code for this program is well documented, please review the program to gain a better understanding on how the AWG features of the device are controlled.

Pulse Programs

The PulseBlasterDDS-II-300-AWG contains an integrated PulseBlaster pulse generation timing core. This timing core controls all aspects of the systems functionality by setting internal control lines at user specified times. Four user programmable digital outputs are also available for control of external hardware. The internal control lines and user programmable outputs are collectively referred to as flags. The pulse program modifies these flags in a user-defined way to control all aspects of an experiment.

The PulseBlaster uses a robust instruction set to enable the creation of complex pulse programs with ease. Each instruction is defined by an OpCode which specifies the action of that instruction and an optional Instruction data (`inst_data`) field which elaborates on that action. In addition, each instruction specifies the desired value for the flags, as well as a time delay until the next instruction executes. The “next” instruction is not necessarily the next sequential instruction, as the instruction set contains branching and looping instructions which can cause the program to be executed out of sequential order. A list of the available instructions is given in the following table.

OpCode #	***Instruction***	***Inst_data field***	***Function***
0	CONTINUE	Unused	Program execution continues to next instruction.
1	STOP	Unused	Stop execution of program. Aborts the operation of the micro-controller and all digital and analog signals turn off.
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 begging of loop and decrements loop counter.
4	JSR	Address of first subroutine instruction	Program execution jumps to beginning of a subroutine
5	RTS	Unused	Program execution returns to instruction after JSR was called
6	BRANCH	Address of instruction to skip to	Program execution continues at specified instruction. This behaves like the goto statement found in many programming languages
7	LONG_DELAY	Number of desired loops. This value must be greater than or equal to 2.	Required for instructions more than 256 clock cycles. Delay field is multiplied by data field. Execution continues to next instruction
8	WAIT	Unused	Program execution pauses and waits for a software or hardware trigger to resume it. The latency between a trigger occurring and the program resuming is the time used as the delay for the wait instruction plus a fixed time of 6 clock cycles.
9	RTI	Unused	Marks the end of an interrupt service routine.

Table 3: PulseBlaster instructions.

```
//pb_inst_dds2(int freq0, int phase0, int amp0, int dds_en0, int phase_reset0,
//     int freq1, int phase1, int amp1, int dds_en1, int phase_reset1, int flags,
//     int inst, int inst_data, double length);

pb_start_programming(PULSE_PROGRAM);
int start = pb_inst_dds2(0, 0, 0, TX_ENABLE,  NO_PHASE_RESET, 0, 0, 0, TX_DISABLE,
PHASE_RESET, 0xf, CONTINUE, 0, 10.0*us);
pb_inst_dds2(0, 0, 0, TX_DISABLE, PHASE_RESET, 0, 0, 0, TX_ENABLE,  NO_PHASE_RESET,
0x0, BRANCH, start, 10.0*us);

//Program the first interrupt service routine.
int isr1 =  pb_inst_dds2(1, 0, 0, TX_ENABLE, NO_PHASE_RESET, 0, 0, 0, TX_ENABLE,
NO_PHASE_RESET, 0xf, LOOP, 10, 200.0*us);
pb_inst_dds2(1, 0, 0, TX_DISABLE, PHASE_RESET, 0, 0, 0, TX_DISABLE, PHASE_RESET,
0x0, END_LOOP, isr1, 200.0*us);
pb_inst_dds2(0, 0, 0, TX_DISABLE, PHASE_RESET, 0, 0, 0, TX_DISABLE, PHASE_RESET,
0x0, RTI, 0, 100.0*us);
pb_stop_programming();
```

Code Example 3: Programming the Pulse Program.

Control lines

To control the operation of the PulseBlasterDDS-II-300-AWG, each instruction in the pulse program specifies a flag word which sets both the internal control lines and user programmable digital outputs. The control lines stay in the given state for the duration of the instruction. The internal control lines are described below:

Control Line	*Function*
frequency select	Selects between the available frequency registers[5]
TX[6] channel phase select	Selects between the available phase registers[5]
tx_enable	Enables TX output on the Analog Out connector. If this control line is disabled, the Analog Out channel is turned off (zero output voltage).
phase_reset	When this control line is enabled, all DDS channels will be reset to their time=0 phase. For example, if a channel is set to use a phase register with 90deg, it will be reset to the midpoint output level and stay that way until the phase reset control line is disabled. This allows the the phase of pulses to be synchronized between scans.

Table 4: Internal control lines.

[5] Please refer to Table 9 for the number of available frequency, phase, and amplitude registers.
[6] TX refers to the RF output.

Triggering

The PulseBlasterDDS can be triggered in two ways, either by software trigger or hardware trigger. The software trigger is initiated by sending a command from the host PC. Because these devices are typically used with non real-time operating systems, the exact time between issuing a software trigger and the device acting on that trigger cannot be precisely specified. For precision control, the pulse program can also be triggered by the transition of the HW_Trigger pin from logical 1 to logical 0. This will cause the pulse program to be triggered within two clock cycles (starting a program), or a minimum of 8 clock cycles (resuming from WAIT instruction). NOTE: The PulseBlasterDDS-II-300-AWG supports the function, pb_read_status(), which can be used to determine whether the board is executing a program or not. For more information see the SpinAPI Reference Manual here (the “API Reference” link under the “Windows Drivers and Example Programs“ heading).

NOTE: *The PulseBlasterDDS-II-300-AWG requires a 3.3 V TTL input signal for HW_Reset.* ***A signal that is more than 3.3 V or less than 0 V will damage the device****.*

```
pb_start(); //Trigger the start of a pulse program or resume execution from a WAIT instruction.
```

Code Example 4: Triggering the start of a pulse program.

Triggering the pulse program has one of the following three effects:

1. Begin execution of a pulse program.
2. Restart execution of a pulse program after the device has been reset.
3. Resume execution of a pulse program which is currently paused by a WAIT instruction.

Interrupt Requests

The PulseBlasterDDS-II-300-AWG has four vectored, maskable interrupts (three hardware pins are available for interrupts as well as one software interrupt). To utilize an interrupt, it is necessary to first unmask the interrupt using `pb_set_irq_enable_mask(...)`. Next, the Interrupt Service Routine (ISR) address needs to be programmed to the ISR vector table using `pb_set_isr(..)`. The Interrupt Request (IRQ) can be triggered by either hardware (via the external interrupt pins) or software (using the `pb_generate_interrupt(...)` routine.)[7] The ISR address can be changed while the PulseBlaster core is executing.

Enabling Interrupts requires the following steps:

1. Unmask the interrupt using the routine: **`void`** `pb_set_irq_enable_mask(`**`char`**` mask)`. The lower four bits of the mask correspond to the IRQ[3..0] mask bits.
2. Program the corresponding ISR using the routine: **`void`** `pb_set_isr(`**`int`**` irq_num, `**`int`**` address)`.
   -*irq_num* is the IRQ[3..0] that the ISR will be set for.
   -*address* is the ISR start address.

The interrupts on the PulseBlasterDDS-II-300-AWG, by default, wait for the current instruction to finish execution before performing the ISR when an interrupt occurs. To immediately perform an ISR when an interrupt occurs, it is necessary to program the interrupt immediate mask using the routine:
`int` `pb_set_irq_immediate_mask(`**`char`**` mask)`. Where the lower four bits of the mask correspond to the enable bit for IRQ[3..0].

Interrupt Service Routine Pulse Program

The Pulse Program from an ISR is like any other pulse program. The only difference is that the last instruction in the ISR must be “RTI”(Return from Interrupt). Without the RTI instruction, the interrupt handler will not know when the ISR has completed and will never return to the interrupt return address.

```
pb_set_interrupt_enable_mask(0x1);//Enable IRQ0
pb_set_isr(0,isr1); //Program IRQ0 ISR address.
```

Code Example 5: Enabling Interrupts.

[7] The delay for the `pb_generate_interrupt(..)` routine is unknown and depends completely on the Windows operating system.

Available Core Registers

There are currently two available core registers that can be written to at any time using the SpinAPI function `pb_write_register(unsigned int address, unsigned int value);`. These registers contain the starting address location of the pulse program, and the 'return-to' state of the TTL Flags when the PulseBlaster Core is in a STOP state.

The registers are written to use a base + offset addressing scheme. The base address for the PulseBlaster Core is 0x040000. The offset for the pulse program starting address register is 0x07. The offset value for the 'return-to' state of the TTL Flags register is 0x08. **Note:** Once a core register is written to it will retain that value until it is changed or the device is powered off.

The following code examples show how this works:

```
int start1 = pb_inst_dds2(FREQ0, PHASE0, AMP0, TX_ENABLE,
NO_PHASE_RESET, FREQ0,PHASE0, AMP0, TX_DISABLE, PHASE_RESET, 0xf,
CONTINUE, 0, 10.0*us);

pb_inst_dds2(FREQ0, PHASE0, AMP0, TX_DISABLE, PHASE_RESET, FREQ0,
PHASE0, AMP3, TX_ENABLE,  NO_PHASE_RESET, 0x0, BRANCH, start1,
10.0*us);

int start2 = pb_inst_dds2(FREQ1, PHASE0, AMP1, TX_ENABLE,
NO_PHASE_RESET, FREQ0,PHASE0, AMP0, TX_DISABLE, PHASE_RESET, 0xf,
CONTINUE, 0, 20.0*us);

pb_inst_dds2(FREQ0, PHASE0, AMP0, TX_DISABLE, PHASE_RESET, FREQ0,
PHASE0,AMP2, TX_ENABLE,  NO_PHASE_RESET, 0x0, BRANCH, start2, 10.0*us);

int start3 = pb_inst_dds2(FREQ0, PHASE0, AMP1, TX_ENABLE,
NO_PHASE_RESET, FREQ0,PHASE0, AMP0, TX_DISABLE, PHASE_RESET, 0xf,
CONTINUE, 0, 30.0*us);

pb_inst_dds2(FREQ0, PHASE0, AMP0, TX_DISABLE, PHASE_RESET, FREQ1,
PHASE0,AMP1, TX_ENABLE,  NO_PHASE_RESET, 0x0, BRANCH, start3, 10.0*us);

#define START_LOCATION (0x40000+0x07)
pb_write_register(START_LOCATION,start4);  // You may choose from
                                           //start1, start2, start3
```

Code Example 6: Changing Pulse Program Start Address.

```
// This will return all the TTL Flags to high when the core stops

      #define FLAG_STATES (0x40000+0x08)

      pb_write_register( FLAG_STATES, 0xf );
```

Code Example 7: Changing 'Return-To' value of TTL Flags.

Pulse Delays

The following instruction shows the recommended way to program the core registers to stop the RF output, thereby creating a gap or delay between the other pulses in a program:

```
pb_inst_dds2_shape(FREQ0, PHASE0, AMP0, NO_SHAPE, TX_DISABLE, PHASE_RESET, FREQ0,
PHASE0, AMP0, NO_SHAPE, TX_DISABLE, PHASE_RESET, 0x000, CONTINUE, 0, 1.0*us);
```

Code Example 8: Adding a gap between pulses.

Other methods for controlling the PulseBlasterDDS-II-300-AWG

The functions provided with SpinAPI can be used to create more than just C executables. Custom software can be made using SpinAPI libraries. Examples of this are LabVIEW and MATLAB programs. LabVIEW examples are provided online at http://spincore.com/support/ and are further detailed in the following section. We currently do not have MATLAB example programs for download. Custom software development is available, please contact SpinCore Technologies, Inc. for details.

IV. LabVIEW Extensions for PBDDS-II-300-AWG

PulseBlaster LabVIEW for the PulseBlasterDDS-II-300-AWG, or the PBLV-DDS-II package, utilizes LabVIEW's intuitive graphical user interface to control the dual RF functionality of the PulseBlaster-DDS-II board.

The LabVIEW extensions manual for the PBDDS-II-300 is available here:

http://www.spincore.com/support/PBLV/PBLV_DDSII_Manual.pdf

To download the GUI interface: http://www.spincore.com/support/PBLV/PBLV_DDSII_Interface.exe

To download the interface examples: http://www.spincore.com/support/PBLV/PBLV_DDSII_Examples.exe

Note: The GUI and examples require LabVIEW 2010 Runtime Engine which is available here:

http://www.spincore.com/support/PBLV/LVRTE2010std.exe

Further LabVIEW information and customizable .VI files are available on our website:

http://www.spincore.com/support/PBLV/DDS.shtml.

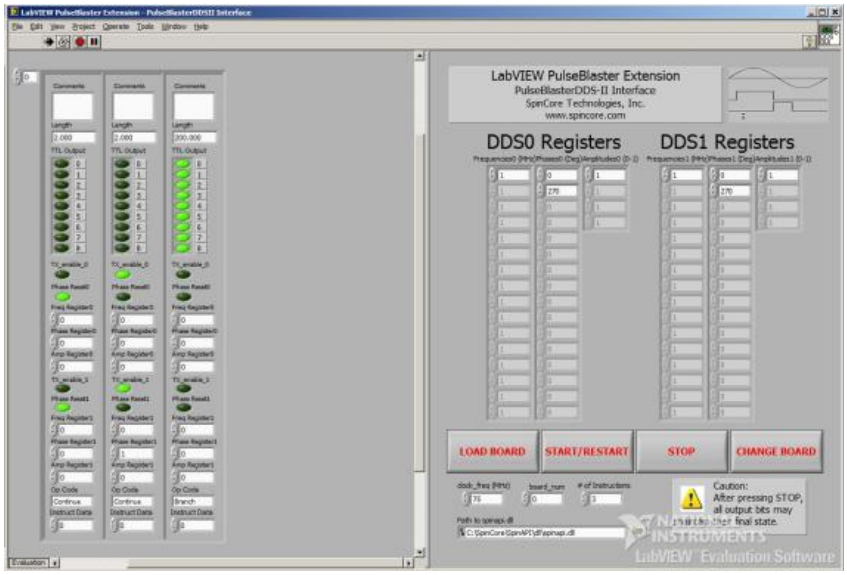

Figure 8: PBLV-DDS-II GUI screenshot.

V. Connecting to the PulseBlasterDDS-II-300-AWG

The PulseBlasterDDS-II-300-AWG is available in two standard options: a) the board-only option b) the single-bay standalone enclosure. The connector information for both options is described below.

Connector information for board-only option

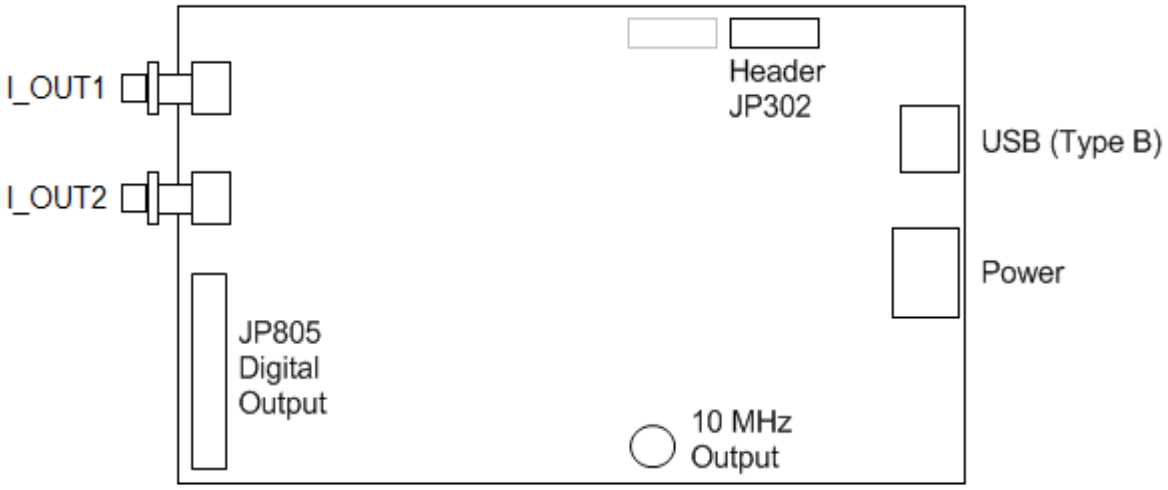

Figure 9: Board connector locations. The board measures 4 X 6 inches (10.2 X 15.2 cm).

Power Connector

The PulseBlasterDDS-II-300-AWG board has a 4-pin Molex-style connector for supplying power. The pin and signal arrangements for this connector is as follows:

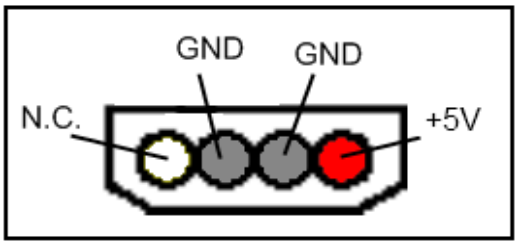

Figure 10: 4 Pin Molex-style Power Connector.

Please note that it is suggested that a PC power supply with the connector specified above is used. This is a standard connector type common on most PCs that will satisfy the power requirements and prevent damage (such as accidentally reversing polarity). The PulseBlasterDDS-II-300-AWG board does not contain over-voltage protection, reverse polarity protection, or undercurrent detection. If a PC power supply is not easily accessible it is suggested that a fixed-voltage +5 Volt power supply with a 2 Amp maximum current be used.

Digital Output Connector (JP805)

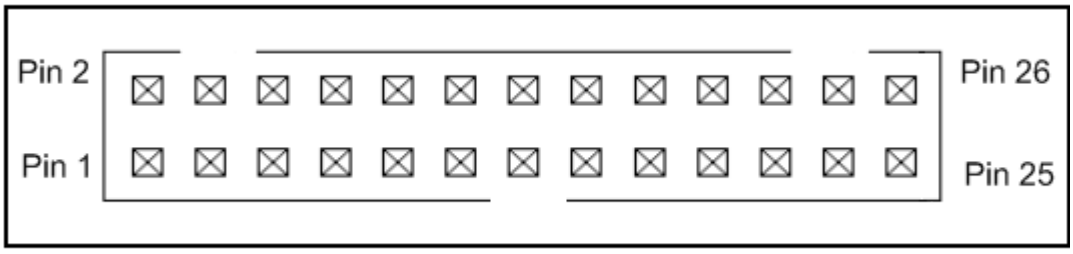

Figure 11: Output Header JP805. Pin 1 is closest to the JP805 label on the silkscreen.

Pin number	Function	Pin number	Function
1	FLAG 0	14	Ground
2	Ground	15	FLAG 7
3	FLAG 1	16	Ground
4	Ground	17	FLAG 8
5	FLAG 2	18	Ground
6	Ground	19	FLAG 9
7	FLAG 3	20	Ground
8	Ground	21	FLAG 10
9	FLAG 4	22	Ground
10	Ground	23	FLAG 11
11	FLAG 5	24	Ground
12	Ground	25	No Connect
13	FLAG 6	26	Ground

Table 5: Onboard Output Header Signal List.

JP805 is a 26-position shrouded male header (Digi-Key part#: A33167-ND). For an appropriate mating connector use a matching 26-position IDC socket connector such as Digi-Key part#: HKC26H-ND. To purchase a cable with the appropriate connector, see: http://spincore.com/products/InterfaceCable/.

Firmware 14-2 has twelve TTL outputs and Firmware 14-3 has four TTL outputs. If you require a specific number of TTL outputs, please contact SpinCore directly.

Alternatively, JP805 can be connected to an SP32 board (Figure 16) which allows the use of MMCX cables. This enables the individual bits of the PulseBlasterDDS-II to be more easily accessed. Pin 1 on the MMCX adapters can be identified with a square pin.

Digital Input Connector (JP302)

The male header on the PulseBlasterDDS-II-300-AWG board labeled JP302 contains the *IRQ, Hardware Trigger* and *Hardware Reset* lines. Please see Figure 14 on the next page.

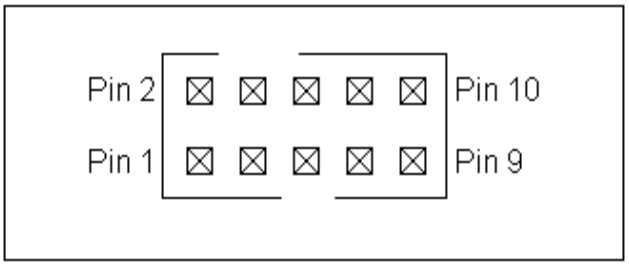

Figure 12: Input Header JP302.

Pin number	Function
1	Ground
2	IRQ0
3	Ground
4	IRQ1
5	Ground
6	IRQ2
7	Ground
8	Hardware Trigger
9	Ground
10	Hardware Reset

Table 6: Input Header JP302 signal list.

JP302 is a 10-position shrouded male header (Digi-Key part#: A33159-ND). For an appropriate mating connector use a matching 10-position IDC socket connector such as Digi-Key part#: HKC10H-ND.

10 MHz Reference Clock Output

PulseBlasterDDS-II-300-AWG board's 10 MHz SMA connector continuously outputs a 50% duty cycle square wave derived directly from the on-board 50 MHz clock oscillator, and is intended for synchronization purposes.

Clock Input Signal Standard

The PulseBlasterDDS-II-300-AWG is a digital system built in CMOS technology and powered off a 3.3 V DC source. It will accept external clock signals that conform to the low-voltage 3.3 V TTL standard only. Negative voltage below 0 Volts would damage the processor chip, and thus any external sinusoidal signal would need to be converted to the positive-only TTL signal prior to using with the PulseBlasterDDS. The PulseBlasterDDS-II-300-AWG board is only designed to work with a 50 MHz clock. Note that that this refers to the on-board clock oscilllator, not the PulseBlaster Core frequency(75MHz standard). A sub-ppm stable oven controlled oscillator is available. If your application calls for a different clock rate, please contact SpinCore Technologies.

Analog Output Connectors

The PulseBlasterDDS-II-300-AWG has two BNC analog outputs designated I_OUT1 and I_OUT2. These RF outputs should be connected to reconstructive filters (either low-pass or band-pass) and then connected to the load (typically 50 Ohms). Both BNC and SMA versions of these filters are available. More information on the significance of the reconstructive filtering can be found in Section VII.

Connector information for single-bay enclosure

For the single-bay enclosure, the diagram below shows the location of all the connectors.

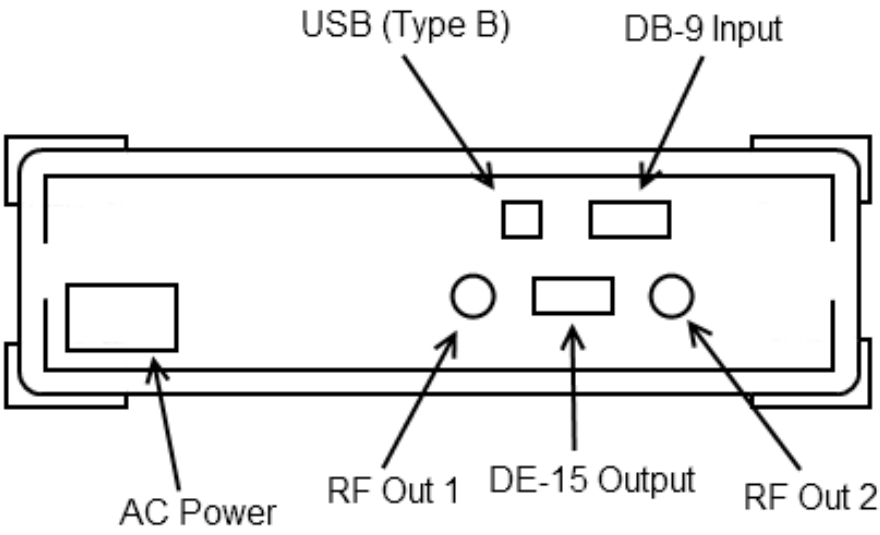

Figure 13: Connector locations at the rear of the enclosure.

AC Power Connector

The PulseBlasterDDS-II-300-AWG is provided with an internal auto-sensing universal AC power supply. An IEC C14 male connector is exposed at the rear of the enclosure which may be connected to 120/240V, 50/60Hz mains power.

Analog Output Connectors

The PulseBlasterDDS-II-300-AWG has two BNC analog outputs designated RF OUT 1 and RF OUT 2. The RF outputs should be connected to reconstructive filters (either low-pass or band-pass) and then connected to the load (typically 50 Ohms). Both BNC and SMA versions of these filters are available. More information on the significance of the reconstructive filtering can be found in Section VII.

DE-15 Output Connector (TTL)

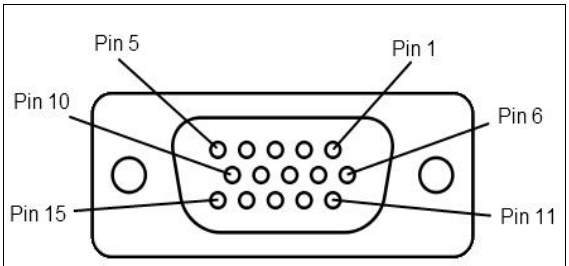

Figure 14: DE-15 Output Connector.

Pin number	Function
1	FLAG 1
2	FLAG 3
3	FLAG 5
4	FLAG 7
5	FLAG 9
6	Ground
7	Ground
8	Ground
9	Ground
10	Ground
11	FLAG 0
12	FLAG 2
13	FLAG 4
14	FLAG 6
15	FLAG 8

Table 7: Output DE-15 (TTL) signal list.

The TTL Output connector is a 15-pin D-sub socket connector (Digi-Key part#: 17HD015SAA000-ND). For an appropriate mating connector use a matching 15-pin D-sub pin connector such as Digi-Key part#: 17EHD015PAA000-ND. To purchase a cable with the appropriate connector, see: http://spincore.com/products/InterfaceCable/.

Firmware 14-2 and Firmware 14-3 have four TTL outputs. If you require a specific number of TTL outputs, please contact SpinCore directly.

DB-9 Input Connector (IRQ/Trig/Res)

Digital inputs are provided through a female DB9 connector at the rear of the enclosure. These connections provide interrupt, trigger and reset support.

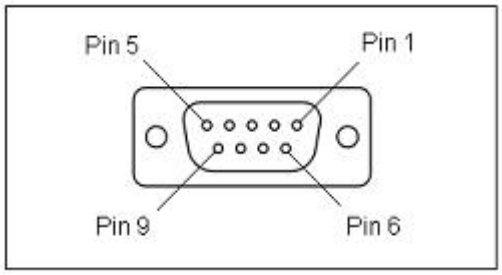

Figure 15: DB9 Input Connector.

Pin number	Function
1	IRQ0
2	IRQ1
3	IRQ2
4	Hardware Trigger
5	Hardware Reset
6	Ground
7	Ground
8	Ground
9	Ground

Table 8: Input DB9 (IRQ/Trig/Rst) signal list.

IRQ[2..0] (pins 1-3) When this input detects a logical true voltage level, an interrupt is produced if the interrupt has been enabled.

Hardware Trigger (pin 4) is pulled high by default using a 10kΩ resistor. When this input detects a low signal (for example by shorting it with pin 6), a hardware trigger is produced. This has the same effects as issuing a trigger through software, though the hardware trigger is more precise, since there are no software latencies involved.

NOTE: *The PulseBlasterDDS-II-300-AWG requires a 3.3 V TTL input signal for HW_Trigger.* ***A signal that is more than 3.3 V or less than 0 V will damage the device****.*

Hardware Reset (pin 5) is pulled high by default using a 10 kΩ resistor. When this input detects a falling edge (for example by shorting it with pin 7), the pulse program is reset.

NOTE: *The PulseBlasterDDS-II-300-AWG requires a 3.3 V TTL input signal for HW_Reset.* ***A signal that is more than 3.3 V or less than 0 V will damage the device****.*

VI. PBDDS-II-300-AWG Firmware Designs

The chart below shows specifications for different PBDDS-II-300-AWG firmware designs. The PBDDS-II-300-AWG firmware design can be read from the board using the read_firmware.exe example (located in: ...\SpinCore\SpinAPI\general). (Note: the firmware ID can also be physically read from the label placed on the design EEPROM of each board).

Firmware	*Clock Speed (on board INPUT-pb core-DAC)*	*Frequency registers*	*Amplitude registers*	*Phase registers*	*Instruction words*	*TTL digital output flags*
14-2	50-75-300	16	4	8	8k	12
14-3	50-75-300	1024	1024	128	4k	4

Table 9: Available Firmware designs for PBDDS-II-300-AWG.

VII. Significance of reconstructive filters

The PBDDS-II digitally generates analog output signals. The RF outputs should be connected to reconstructive filters (either low-pass or band-pass) and then connected to the load (typically 50 Ohms).

Below is an oscilloscope image of two traces that show an 80 MHz signal generated with the PBDDS-II. At this frequency, there are only 3.75 samples per period (300MHz/80MHz). The bottom trace is the unfiltered output signal and the top trace is the same signal after being passed through a bandpass-filter. Instead of a passive filter, a selective amplifier could also be used.

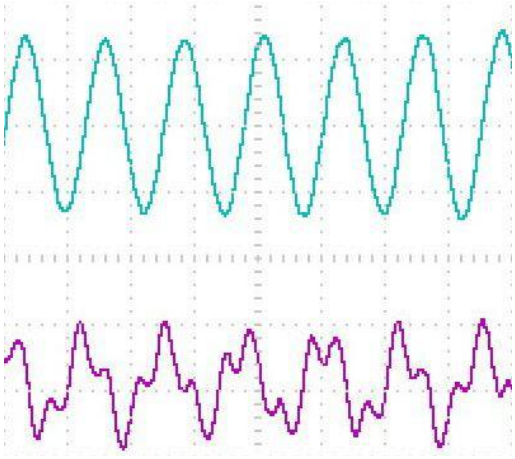

Figure 16: An 80 MHz signal before (bottom) and after (top) being bandpass filtered.

Related Products and Accessories

1. RadioProcessor – Complete single-card solution for RF pulse generation and acquisition. For more information, please visit http://spincore.com/products/RadioProcessor/.

2. If you require a specific number of amplitude registers, or memory words, or an Oven Controlled Clock Oscillator (sub-ppm stability), please inquire with SpinCore Technologies through our contact form, which is available at http://spincore.com/contact.shtml

3. SpinCore MMCX Adapter Board Figure 16 – This adapter board allows easy access to the individual bits of the PulseBlasterDDS-II-300-AWG. This adapter board can be part of a package that includes 12 MMCX to BNC cables and three SMA to BNC adapters. This package can be changed to include any number of cables and any number of adapter boards. For ordering information contact SpinCore at http://www.spincore.com/contact.shtml.

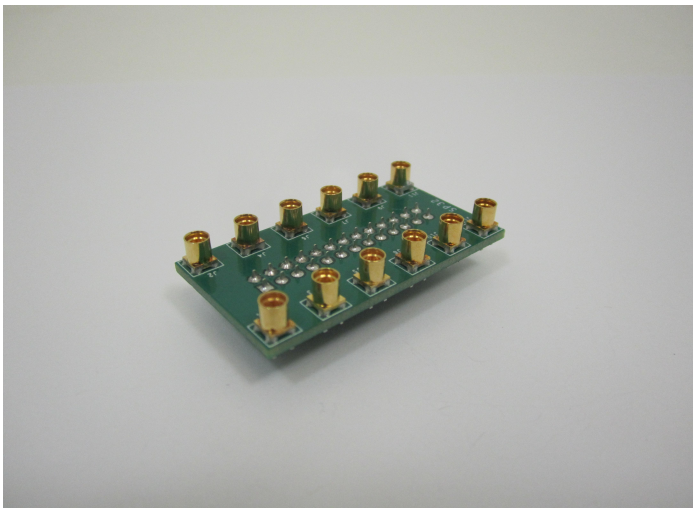

***Figure 17**: MMCX Adapter Board allows easy access to the Digital Output*

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://www.spincore.com/contact.shtml

Document Information

Revision history available at SpinCore.