The PicoBlaze firmware in the Bio Sensor will communicate with a host computer via an RS-232 serial link, and provide for control of the device. Control logic resident in the FPGA along with the PicoBlaze will provide the low-level interface to the device hardware.

A block diagram of the system is shown in Illustration 1. There are four hardware devices on the PC board connected to the FPGA:

1. A Four-channel DAC to set DC bias voltages (DAC7564)
   This device is controlled directly by the PicoBlaze through a simple serial interface

2. A Single-channel DAC to generate the AC waveform (DAC7551)
   This device receives a sinusoidal waveform from a RAM, and is controlled by the FPGA logic.

3. A Two-channel DAC to cancel DC offsets in the measured signal.
   This device will initially be controlled by a simple serial interface, but there may be control logic implemented to zero the offset automatically.

4. A Four-channel ADC to measure the device current (AD7682)
   This device will be controlled by the FPGA logic. An option to capture a 256-sample raw waveform which may be read by the PicoBlaze will be provided.

At power-up, the PicoBlaze should delay for 1 second, then write a short start-up message, issue a prompt 'S' and wait for input from the host.

Each host command consists of a string of printable ASCII characters terminated with an ASCII CR. The PicoBlaze should echo each received character back to the host. When the host sends a CR, the PicoBlaze should echo the CR and also send a LF. Each line of text sent by the PicoBlaze should be terminated with the sequence CR, LF.

After each command is processed a new 'S' prompt should be sent.

Valid commands are as follows:

| W nn dddd | Write hexadecimal value dddd to register nn |
| R nn      | Read value of register nn                   |

These commands read and write external logic registers.

Register numbers are two hexadecimal digits. Data values are always four hexadecimal digits. These commands will read and write to external devices using address and data ports. See my demo.psm for an example of how to do this.

<table>
<thead>
<tr>
<th>B &lt;Vd1&gt; &lt;Vd2&gt; &lt;Vg1&gt; &lt;Vg2&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vd1 Drain bias for active device</td>
</tr>
<tr>
<td>Vd2 Drain bias for reference device</td>
</tr>
<tr>
<td>Vg1 Gate bias for active device</td>
</tr>
</tbody>
</table>
Vg2  Gate bias for reference device
This command writes 4 16-bit binary values to 4 registers of the bias DAC (which Bill calls “SAMP_DRAIN”, “REF_DRAIN”, “SAMP_SOLUTION”, “REF_SOLUTION”).

Z <Vz1> <Vz2>

Vz1  Zero-offset DAC setting for active device
Vz2  Zero-offset DAC setting for reference device
This command sets the zero-offset DAC registers (which Bill calls “DAC_SAMPLE” and “DAC_REFERENCE”).

S <smode> <sgain> <rmode> <rgain>

smode  Sample mode select
“1”, “2”, “3”, “4” to select a sample input
“T” to select 1M test resistor
some other not-yet-defined modes possible
sgain  Sample gain select
“1” or “5” to select G=1 or G=1/5
rmode  Reference mode select (same choices)
rgain  Reference gain select
This command sets several bits in Bill's registers “SAMPLE_SEL” and “REFERENCE_SEL”.

AZ

aaaa bbbb
This command performs an “auto-zero” operation by writing the corresponding command code to the command register and waiting for the operation to complete.

After the operation completes, two 16-bit values are sent which correspond to the zero-cancel DAC settings (from Bill's registers “DAC_SAMPLE” and “DAC_REFERENCE”). If the operation does not complete after some reasonable time (one second?) the message “FAIL” should be sent.

AC <mag> <cycles> <average>

aaaa bbbb cccc dddd ...

mag  One of 4 possible AC magnitudes (“0”, “5”, “10”, “20”).
cycles  Number of cycles to acquire
average  “Y”/”N” flag to enable averaging
This will acquire data for the number of cycles specified.
The magnitude is written as a two-bit code to Bill's register “AC_DRIVE”.
The output consists of four 16-bit hex numbers representing the average value calculated for each of the 4 measured quantities (reference sin, reference cos, sample sin, sample cos). If averaging is disabled, a list of <cycles> raw values will be output.

This command reads 256 raw samples for active and reference data from an external block RAM and sends them to the host.

Illustration 1: System Block Diagram
Appendix A – PicoBlaze I/O

The PicoBlaze I/O ports will be used as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Address/bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART_status_port</td>
<td>00 bit 0</td>
<td>TX FIFO half full</td>
</tr>
<tr>
<td></td>
<td>00 bit 1</td>
<td>TX FIFO full</td>
</tr>
<tr>
<td></td>
<td>00 bit 2</td>
<td>RX FIFO half full</td>
</tr>
<tr>
<td></td>
<td>00 bit 3</td>
<td>RX FIFO full</td>
</tr>
<tr>
<td></td>
<td>00 bit 4</td>
<td>RX FIFO not empty (data available)</td>
</tr>
<tr>
<td>UART_data_port</td>
<td>01</td>
<td>Read/write UART data</td>
</tr>
<tr>
<td>mstimer_lo_port</td>
<td>02</td>
<td>Millisecond utility timer (low 8 bits)</td>
</tr>
<tr>
<td>mstimer_hi</td>
<td>03</td>
<td>Millisecond utility timer (upper 8 bits)</td>
</tr>
<tr>
<td>ext_port_hi</td>
<td>04</td>
<td>External logic port upper data byte</td>
</tr>
<tr>
<td>(external addresses)</td>
<td>80..FF</td>
<td>External port access (addresses defined by Bill)</td>
</tr>
</tbody>
</table>

The external logic simulates a 16-bit wide data path to permit access to 16-bit external registers. The external address is 7 bits.

Procedure for 16-bit write:

- Write upper 8 bits to ext_port_hi
- Write lower 8 bits to address 80...FF (the low 7 address bits are the desired external address)

Procedure for 16-bit read:

- Read lower 8 bits from address 80..FF (upper 8 bits are captured in an external register)
- Read upper 8 bits from ext_port_hi
Appendix B – PRELIMINARY Register List from Bill

Updated 2/5/09 – Subject to change
Control Registers in the Bio-Sensor FPGA

William E. Earle

February 5, 2009

This document describes the 16 control registers used to control the operation of the bio-sensor FPGA, or, more specifically, the part of the logic in the FPGA that communicates with the bio-sensor so as to collect and process sensor data. Most of the registers hold operating parameters such as the various bias voltages, with a few others used to supply processed readout data.

Before discussing the details of the control registers, it is helpful to provide a brief description of the bio-sensor as it relates to the FPGA logic, starting with the block diagram of Fig. 1. The bio-sensor itself consists of eight specially constructed FET transistors, divided into two groups of four each, one group called the sample FET’s and the other the reference FET’s. The sample FET’s are immersed in a sample solution that also contains a solution electrode. Similarly, the reference FET’s are immersed in a reference solution that also contains a solution electrode.

As shown in Fig.1, the four drains of the sample FET’s are connected together and driven by a signal called sample_drive, consisting of a dc bias, called sample_bias, plus a small ac signal. The four drains of the reference FET’s are also connected together and driven by a signal called reference_drive, consisting of a dc bias, called reference_bias, plus the same ac signal used by the sample FET’s.

The sources of all of the FET’s are connected to individual selectable-gain transimpedance amplifiers that convert the source currents into voltage signals which feed the analog input circuit shown in block form in Fig.2.

Fig. 2 shows multiplexers that select one of the four sample transimpedance signals and one of the four reference transimpedance signals. These signals are summed with dc sample_offset and reference_offset signals and then amplified by gains of 100 before being sent to an A/D converter. The A/D
The converter has an input multiplexer that not only allows conversion of a sample or reference signal but also the sample_drive or reference_drive signals, these latter included to provide some circuit testing capability.

1 The 16 Registers

The data formats for the registers are shown in Fig. 3. The following describes the operation of the registers:

- **A_D INPUT**: This register controls the selection of the signals to the A/D converter. As the FPGA processes and stores bio-sensor data, A/D conversions alternate between an A input and a B input as specified by this register. The B input is simple, it allows for one of the four reference FET’s or a test 1 Megohm resistor to be selected. The A input allows the selection of one of the four sample FET’s or a 1 Megohm test resistor, but it also allows for four differential inputs plus the sample and reference drive signals. The differential inputs cause the A/D to convert the difference between the selected sample FET and the selected reference FET.

- **SAMPLE_BIAS**: Writing to this register causes a serial data stream to be sent to a DAC that provides the dc sample bias. The value ranges from -2.5V to +2.5V. While the serial data is being sent to the DAC, the “serial_busy” bit in the COMMAND/STATUS register remains high.

- **REFERENCE_BIAS**: Same as SAMPLE_BIAS, but provides the reference bias.

- **SAMPLE_SOLUTION**: Same as SAMPLE_BIAS but drives the sample solution electrode.

- **REFERENCE_SOLUTION**: Same as SAMPLE_BIAS but drives the reference solution electrode.

- **SAMPLE_OFFSET**: Similar to SAMPLE_BIAS but has a range of -1.048V to +1.048V. Used to remove most of the dc component from the A input to the A/D. May be written directly, but normally is set by the zero command. After a “zero” operation, the value of SAMPLE_OFFSET is a measure of the dc current through the sample FET.
• **REFERENCE_OFFSET**: Same as **SAMPLE_OFFSET** but removes the dc component from the B input.

• **A_OUT_INPHASE**: This is a read-only register that is updated at the end of each cycle of the ac sinewave signal. It contains the amplitude of the in-phase component from the A input.

• **A_OUT_QUADRATURE**: Same as **A_OUT_INPHASE** but contains the quadrature component from the A input.

• **B_OUT_INPHASE**: Same as **A_OUT_INPHASE** but contains the in-phase component from the B input.

• **B_OUT_QUADRATURE**: Same as **B_OUT_INPHASE** but contains the quadrature component from the B input.

• **A_RAW_DATA**: This is a FIFO-like read-only register that contains raw A/D data from the A input over one cycle (256 values) of the ac sinewave signal. This register should be read 256 times to retrieve its contents. The values in this register are only updated after a save_one_cycle command is executed. Normally this register is only used for system test and trouble-shooting. One use for the RAW_DATA values is for determining how well the “zero” operation succeeded in zeroing the dc inputs to the A/D converter. If the RAW_DATA values are close to midscale (8000 hex) then the “zero” operation was successful.

• **B_RAW_DATA**: Same as **A_RAW_data** but contains data from the B input.

• **PHASE_INCREMENT_L**: The lower 16 bits of the 24 bit phase increment that determines the frequency of the ac sinewave. **PHASE_INCREMENT_H** contains the upper eight bits. A value of 2,199,023 (218DEF hex) will set the frequency to 100 Hz and a value of 21,990 will give a 1 Hz frequency. For 17 Hz, the value would be 17/100 times 2,199,023, or 373,834.

• **COMMAND/STATUS**: A one written to the zero bit causes the **SAMPLE_OFFSET** and **REFERENCE_OFFSET** values to be adjusted so as to set the dc levels of the A and B inputs as close as possible to midscale (1.25V). The ac drive signal is turned off during this zeroing
operation. A one written to the save bit causes one cycle of the A and B inputs (256 values each) to be saved in the \texttt{A\_RAW\_DATA} and \texttt{B\_RAW\_DATA} registers (pseudo FIFO's). The \texttt{cycle\_done} bit is a symmetrical square wave with the rising-edge indicating new data in the INPHASE and QUADRATURE registers. There is also a \texttt{serial\_busy} bit that shows when serial data is being sent to a DAC. Writing to a DAC register can only be initiated when this bit is low.
Fig. 1 Bio-sensor electrical connections
Fig. 2 A/D input connections
Fig. 3a Register formats
<table>
<thead>
<tr>
<th>Address</th>
<th>Register</th>
<th>Read/Write</th>
<th>Bits</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>SAMPLE_SOLUTION_BIAS</td>
<td>read–write</td>
<td>15</td>
<td>unsigned 12 bits</td>
</tr>
<tr>
<td>4</td>
<td>REFERENCE_SOLUTION_BIAS</td>
<td>read–write</td>
<td>15</td>
<td>unsigned 12 bits</td>
</tr>
<tr>
<td>5</td>
<td>SAMPLE_OFFSET</td>
<td>read–write</td>
<td>15</td>
<td>unsigned 12 bits</td>
</tr>
<tr>
<td>6</td>
<td>REFERENCE_OFFSET</td>
<td>read–write</td>
<td>15</td>
<td>unsigned 12 bits</td>
</tr>
<tr>
<td>7</td>
<td>A_OUT_INPHASE</td>
<td>read–only</td>
<td>15</td>
<td>16 bits 2’s complement</td>
</tr>
<tr>
<td>8</td>
<td>A_OUT_QUADRATURE</td>
<td>read–only</td>
<td>15</td>
<td>16 bits 2’s complement</td>
</tr>
<tr>
<td>9</td>
<td>B_OUT_INPHASE</td>
<td>read–only</td>
<td>15</td>
<td>16 bits 2’s complement</td>
</tr>
<tr>
<td>10</td>
<td>B_OUT_QUADRATURE</td>
<td>read–only</td>
<td>15</td>
<td>16 bits 2’s complement</td>
</tr>
</tbody>
</table>

Fig. 3b  Register formats
**Fig. 3c  Register formats**