Register Functions for the
2006 ATLAS Pixel Opto-link SLHC Test Stand
D.S.S. 2/22/07

#define OPTO_RESET 0x0

Writing any value to this register sends a predefined length (~500us) reset signal to the optoboard.

#define SET_TEST 0x1

The value written to this register is sent to J321 nets spare1 - spare8.

#define IP_SEL 0x2

The value read from this register is the value from the IPSEL switch, SW1.

VDC Control Registers

The VDCs on the test stand are controlled by two programmable pots, one for setting the VCSEL bias current and the other for setting the VCSEL modulation current. Each pot has 10 bit resolution.

#define POT_LSB 0x3

The value written to this register will be loaded into the low byte of the pot selected by the POT_SELECT register following a write to the LOAD_POT register.

#define POT_MSB 0x4

The value written to the lower two bits of this register will be loaded into the high byte of the pot selected by the POT_SELECT register following a write to the LOAD_POT register.

#define POT_SELECT 0x5

Writing a value to this register selects which pot will be loaded upon a write to the LOAD_POT register.

<table>
<thead>
<tr>
<th>Value</th>
<th>Pot to be loaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Modulation Pot Channel 1</td>
</tr>
<tr>
<td>1</td>
<td>Bias Pot Channel 1</td>
</tr>
<tr>
<td>2</td>
<td>Modulation Pot Channel 2</td>
</tr>
<tr>
<td>3</td>
<td>Bias Pot Channel 2</td>
</tr>
<tr>
<td>4</td>
<td>Modulation Pot Channel 3</td>
</tr>
<tr>
<td>5</td>
<td>Bias Pot Channel 3</td>
</tr>
<tr>
<td>6</td>
<td>Modulation Pot Channel 4</td>
</tr>
<tr>
<td>7</td>
<td>Bias Pot Channel 4</td>
</tr>
<tr>
<td>8</td>
<td>Modulation Pot Channel 5</td>
</tr>
</tbody>
</table>
9  Bias Pot Channel 5
10  Modulation Pot Channel 6
11  Bias Pot Channel 6
12  Modulation Pot Channel 7
13  Bias Pot Channel 7

#define LOAD_POT 0x6

Writing anything to this register loads the values present in POT_LSB and POT_MSB into the pot selected by POT_SELECT. A single initial write to this register (the value will not be reflected on any of the pots) is required to initialize the register operation (there is a bug in the FPGA logic).

#define MUX_CHANNEL 0x7

Writing a value to this register selects which channel will be fed through the MUX to the ADC.

<table>
<thead>
<tr>
<th>Value</th>
<th>Channel connected to ADC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BIASMON VDC Channel 1</td>
</tr>
<tr>
<td>1</td>
<td>Data Optical Power Channel 1</td>
</tr>
<tr>
<td>2</td>
<td>Clock Optical Power Channel 1</td>
</tr>
<tr>
<td>3</td>
<td>BIASMON VDC Channel 2</td>
</tr>
<tr>
<td>4</td>
<td>Data Optical Power Channel 2</td>
</tr>
<tr>
<td>5</td>
<td>Clock Optical Power Channel 2</td>
</tr>
<tr>
<td>6</td>
<td>BIASMON VDC Channel 3</td>
</tr>
<tr>
<td>7</td>
<td>Data Optical Power Channel 3</td>
</tr>
<tr>
<td>8</td>
<td>Clock Optical Power Channel 3</td>
</tr>
<tr>
<td>9</td>
<td>BIASMON VDC Channel 4</td>
</tr>
<tr>
<td>10</td>
<td>Data Optical Power Channel 4</td>
</tr>
<tr>
<td>11</td>
<td>Clock Optical Power Channel 4</td>
</tr>
<tr>
<td>12</td>
<td>BIASMON VDC Channel 5</td>
</tr>
<tr>
<td>13</td>
<td>Data Optical Power Channel 5</td>
</tr>
<tr>
<td>14</td>
<td>Clock Optical Power Channel 5</td>
</tr>
<tr>
<td>15</td>
<td>BIASMON VDC Channel 6</td>
</tr>
<tr>
<td>16</td>
<td>Data Optical Power Channel 6</td>
</tr>
<tr>
<td>17</td>
<td>Clock Optical Power Channel 6</td>
</tr>
<tr>
<td>18</td>
<td>BIASMON VDC Channel 7</td>
</tr>
<tr>
<td>19</td>
<td>Data Optical Power Channel 7</td>
</tr>
<tr>
<td>20</td>
<td>Clock Optical Power Channel 7</td>
</tr>
<tr>
<td>21</td>
<td>Spare 1</td>
</tr>
<tr>
<td>22</td>
<td>Spare 2</td>
</tr>
<tr>
<td>23</td>
<td>Spare 3</td>
</tr>
<tr>
<td>24</td>
<td>Spare 4</td>
</tr>
<tr>
<td>25</td>
<td>Spare 5</td>
</tr>
<tr>
<td>26</td>
<td>Spare 6</td>
</tr>
<tr>
<td>27</td>
<td>Spare 7</td>
</tr>
<tr>
<td>28</td>
<td>VVDC current</td>
</tr>
<tr>
<td>29</td>
<td>IPIN</td>
</tr>
<tr>
<td>30</td>
<td>ISET</td>
</tr>
<tr>
<td>31</td>
<td>VVDC</td>
</tr>
</tbody>
</table>
#define CONVERT 0x8

Writing any value to this register will initiate an ADC conversion.

#define ADC_LOW_BYTE 0x9

The value read from this register is the low byte value from the last ADC sample.

#define ADC_HIGH_BYTE 0xA

The value read from this register is the high byte value from the last ADC sample.

#define LCD_DATA 0xB

The value written to this register is sent to the LCD connector (J320) as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LCD4 (pin 9)</td>
</tr>
<tr>
<td>1</td>
<td>LCD5 (pin 10)</td>
</tr>
<tr>
<td>2</td>
<td>LCD6 (pin 11)</td>
</tr>
<tr>
<td>3</td>
<td>LCD7 (pin 12)</td>
</tr>
<tr>
<td>4</td>
<td>NC</td>
</tr>
<tr>
<td>5</td>
<td>NC</td>
</tr>
<tr>
<td>6</td>
<td>LCDRW (pin 7)</td>
</tr>
<tr>
<td>7</td>
<td>LCDRS (pin 6)</td>
</tr>
</tbody>
</table>

#define LCD_E 0xC

The value written to the lowest bit of this register is reflected on the LCDE line (J320 pin 8).

#define KEY_PAD 0xD

The value read from this register contains the front panel control information as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UP (J268)</td>
</tr>
<tr>
<td>1</td>
<td>DOWN (J269)</td>
</tr>
<tr>
<td>2</td>
<td>LEFT (J270)</td>
</tr>
<tr>
<td>3</td>
<td>RIGHT (J271)</td>
</tr>
<tr>
<td>4</td>
<td>ENTER (J265)</td>
</tr>
<tr>
<td>5</td>
<td>BACK (J266)</td>
</tr>
<tr>
<td>6</td>
<td>SPARE 1 (J272)</td>
</tr>
<tr>
<td>7</td>
<td>SPARE 2 (J273)</td>
</tr>
</tbody>
</table>
#define SPIFFS 0xEE

The value read from bit 0 of this register reflects what is present at J263, bit 1 reflects J264.

***********************************************************************
Per Channel Control Registers
(n) denotes the channel you are controlling

CHANNEL_BEHAVE(n)

The value written to this register changes various behavior for the channel:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Selects whether the value from the pattern register or a pseudo-random bit stream is sent to the VDC. 0 = PRBS, 1 = pattern register</td>
</tr>
<tr>
<td>1</td>
<td>Selects whether the outgoing bit stream is BPM encoded or not. 0 = not encoded, 1 = BPM encoded</td>
</tr>
<tr>
<td>2</td>
<td>Selects whether the incoming bit stream should be decoded from BPM for comparison. 0 = don't decode, 1 = decode</td>
</tr>
<tr>
<td>3</td>
<td>Selects internal loopback mode for bit stream comparison. 0 = don't loopback, 1 = loopback</td>
</tr>
<tr>
<td>4</td>
<td>Clear the error count register. 0 = don't clear the error count, 1 = clear the error count</td>
</tr>
<tr>
<td>5</td>
<td>Disables the channel's VDC. 0 = don't disable, 1 = disable</td>
</tr>
<tr>
<td>6</td>
<td>Selects the polarity of the received data. 0 = normal, 1 = inverted</td>
</tr>
<tr>
<td>7</td>
<td>Selects the polarity of the received clock. 0 = normal, 1 = inverted</td>
</tr>
</tbody>
</table>

COMP_LSB(n)

The value written to this register is used to set the number of bit cycles to delay the comparison of the received signal versus the transmitted signal.

COMP_MSB(n)

The value written to this register performs the following:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>Upper bits for channel comparison delay (forms a 10 bit register with the value in COMP_LSB)</td>
</tr>
</tbody>
</table>
2-3 Selects which clock is used for clocking the received data into the FPGA. 0 = received clock, 1 = transmit clock, 2 = transmit clock inverted, 3 = received clock inverted

PATTERN_LSB(n)

The value written to this register will appear as the first 8 bits in the outgoing data stream (16 bits) if the pattern register is selected in the CHANNEL_BEHAVE register.

PATTERN_MSB(n)

The value written to this register will appear as the last 8 bits in the outgoing data stream (16 bits) if the pattern register is selected in the CHANNEL_BEHAVE register.

ERRORS_LSB(n)

The value read from this register contains the lower byte of the error counter. This counter records the number of errors in the comparison of the received data stream versus the delayed (COMP_MSB and COMP_LSB) transmitted stream.

ERRORS_MSB(n)

The value read from this register contains the upper byte of the error counter.

Channel Behavior Addresses for all channels
#define CHANNEL_BEHAVE1 0xF
#define CHANNEL_BEHAVE2 0x10
#define CHANNEL_BEHAVE3 0x11
#define CHANNEL_BEHAVE4 0x12
#define CHANNEL_BEHAVE5 0x13
#define CHANNEL_BEHAVE6 0x14
#define CHANNEL_BEHAVE7 0x15

Comparison low byte addresses for all channels
#define COMP_LSB1 0x16
#define COMP_LSB2 0x17
#define COMP_LSB3 0x18
#define COMP_LSB4 0x19
#define COMP_LSB5 0x1A
#define COMP_LSB6 0x1B
#define COMP_LSB7 0x1C

Comparison high byte addresses for all channels
#define COMP_MSB1 0x1D
#define COMP_MSB2 0x1E
#define COMP_MSB3 0x1F
#define COMP_MSB4 0x20
#define COMP_MSB5 0x21
#define COMP_MSB6 0x22
#define COMP_MSB7 0x23
Pattern low byte addresses for all channels
#define PATTERN_LSB1 0x24
#define PATTERN_LSB2 0x25
#define PATTERN_LSB3 0x26
#define PATTERN_LSB4 0x27
#define PATTERN_LSB5 0x28
#define PATTERN_LSB6 0x29
#define PATTERN_LSB7 0x2A

Pattern high byte addresses for all channels
#define PATTERN_MSB1 0x2B
#define PATTERN_MSB2 0x2C
#define PATTERN_MSB3 0x2D
#define PATTERN_MSB4 0x2E
#define PATTERN_MSB5 0x2F
#define PATTERN_MSB6 0x30
#define PATTERN_MSB7 0x31

Error counter low byte addresses for all channels
#define ERRORS_LSB1 0x32
#define ERRORS_LSB2 0x33
#define ERRORS_LSB3 0x34
#define ERRORS_LSB4 0x35
#define ERRORS_LSB5 0x36
#define ERRORS_LSB6 0x37
#define ERRORS_LSB7 0x38

Error counter high byte addresses for all channels
#define ERRORS_MSB1 0x39
#define ERRORS_MSB2 0x3A
#define ERRORS_MSB3 0x3B
#define ERRORS_MSB4 0x3C
#define ERRORS_MSB5 0x3D
#define ERRORS_MSB6 0x3E
#define ERRORS_MSB7 0x3F