SPI

- used on both SCROD and carrier:
  - SCROD:
    - two ADCs (measure voltages on board)
  - carrier:
    - two ADCs (measure ASIC I/V, other currents/voltages on board)
    - DAC (to set ASIC pedestal)
    - clock divider (to set sampling clock divider ratio)
<table>
<thead>
<tr>
<th>board</th>
<th>SPI bus</th>
<th>device</th>
<th>ICS</th>
<th>function</th>
<th>ch0</th>
<th>ch1</th>
<th>ch2</th>
<th>ch3</th>
<th>ch4</th>
<th>ch5</th>
<th>ch6</th>
<th>ch7</th>
<th>each ADC/DAC count is worth</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCROD</td>
<td>1ADC0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12 bit ADC</td>
<td>2.5V</td>
<td>3.3V</td>
<td>2.5V</td>
<td>1.8V_AUXIO</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>2 mV (approx)</td>
</tr>
<tr>
<td>SCROD</td>
<td>1ADC1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>12 bit ADC</td>
<td>1.8V_VCCO</td>
<td>1.5V_VCCO</td>
<td>1.0V_GTX</td>
<td>1.2V_GTX</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>2 mV (approx)</td>
</tr>
<tr>
<td>carrier</td>
<td>0/fanout_div1</td>
<td>0</td>
<td>116 bit DAC</td>
<td>0</td>
<td>clock fanout &amp; divider</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>2 mV (approx) or 38 uV (approx)</td>
</tr>
<tr>
<td>carrier</td>
<td>0/DAC</td>
<td>0</td>
<td>116 bit DAC</td>
<td>1</td>
<td>Vped_asic0</td>
<td>Vped_asic1</td>
<td>Vped_asic2</td>
<td>Vped_asic3</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>2 mV (approx) or 2 mA (approx)</td>
</tr>
<tr>
<td>carrier</td>
<td>1ADC0</td>
<td>0</td>
<td>16</td>
<td>16 bit ADC</td>
<td>0</td>
<td>2.5Vasic0</td>
<td>2.5Vasic1</td>
<td>2.5Vasic2</td>
<td>2.5Vasic3</td>
<td>imonasic0</td>
<td>imonasic1</td>
<td>imonasic2</td>
<td>imonasic3</td>
<td>2 mV (approx) or 2 mA (approx)</td>
</tr>
<tr>
<td>carrier</td>
<td>1ADC1</td>
<td>0</td>
<td>16</td>
<td>16 bit ADC</td>
<td>1</td>
<td>floating</td>
<td>RAW1</td>
<td>RAW2</td>
<td>RAW3</td>
<td>imonasic0</td>
<td>imonasic1</td>
<td>imonasic2</td>
<td>imonasic3</td>
<td>3.7V_vamps01</td>
</tr>
</tbody>
</table>
SPI

// AD7933 4-channel 12-bit ADC (on SPI0 SPI2)
// from page 12 and 13 of datasheet:
// control register = (1<<11"WRITE") don't ignore the following") | (1<<10"SCL=sequential conversion") | (1<<8"ADD0=1-final channel=3") | (1<<4"PMO=1-auto-shutdown") | (0<<3"SLOE=sequencial conversion")
// seems to be a single address device; one 12 bit control word in (part of a 16 bit transfer) and one 12 bit result out (in a 16 bit transfer)
// page 5 says SCLK = 10 MHz to 20 MHz
// SCLK is used for conversion
// CPHA=0; CPOL=1; (page 550 of zynq TRM has a chart and this corresponds to "din sampled on the falling edge of sclk" with clock and cs going inactive between words [btw, chart on page 550 and waveform on page 537 seem to disagree]?)
// t4 - hold time; t5 - setup time for falling edge of SCLK (from diagram on page 21)

// IDT81798188-08 fanout/divider (on carrier SPI0)
// from page 5 of the datasheet:
// 22 bit transfer
// CPHA=0; CPOL=0; (waveform on page 5 shows that mosi is sampled on rising edge)
// max SCLK is 20 MHz

// AD5686 4-channel 16-bit DAC (on carrier SPI0)
// from page 21 of the datasheet:
// 24 bit control words
// 4 command bits; 4 address bits; 16 data bits
// CPHA=0; CPOL=1; (waveform on page 6 shows it's sampled on falling edge)

// AD7689 8-channel 16-bit ADC (on carrier SPI0)
// from page 24 and 25 (configuration register) 28 (channel sequencer) and 29 (conversion without a busy indicator) of the datasheet:
// config = (0x13/CRC-overwrite configuration/) | (1<<10/INCH=7 for unipolar, referenced to GND; alternately 3 for temperature sensor?/) | (1<<7/TIN=last channel to convert is channel 0)
// t: alternately any other value 0-7/) | (1<<5/FM-1/) | (0<<3/FINT-internal, 2.5V/) | (2<<1/REQ-ch0 to ch7, (don't temp/) | (1<<0/RSP=don't read back the configuration register/)?

// CPHA=0; CPOL=0; (waveform on page 29 shows it's sampled on rising edge and clock stops and cs goes inactive in between transfers)
// needs to be setup faster than is possible based on this device's maximum conversion rate (page 8 says sclk_min_period = 25.2 ns for a 2.5 v vio, so <= 25 MHz
// similar to the 7923, it is a simultaneous 16 bit write and read with the read corresponding to result of the previous conversion
// const u32 ad7689_microvolts_per_ADC_count = pow(2, 16.0) / 3300000.0 / ((16.0/7.15)/(7.133 / // -124 microvolts per ADC count
// millivolts = (readword & 0xffff) * ad7689_microvolts_per_ADC_count;
## SPI

<table>
<thead>
<tr>
<th>board</th>
<th>device</th>
<th>function</th>
<th>what to write</th>
<th>format</th>
<th>result</th>
<th>format</th>
<th>word length (bits)</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCROD</td>
<td>ADC0</td>
<td>12 bit ADC</td>
<td>12 bit control word</td>
<td>12 bits as part of a 16 bit word</td>
<td>12 bit ADC value</td>
<td>12 bits as part of a 16 bit word</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SCROD</td>
<td>ADC1</td>
<td>12 bit ADC</td>
<td>12 bit control word</td>
<td>12 bits as part of a 16 bit word</td>
<td>12 bit ADC value</td>
<td>12 bits as part of a 16 bit word</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>carrier</td>
<td>fanout_div1</td>
<td>22 bit control word</td>
<td>22 bit control word</td>
<td>22 bit control word</td>
<td>22 bit control word</td>
<td>22 bit control word</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>carrier</td>
<td>DAC</td>
<td>24 bit control word</td>
<td>24 bit control word</td>
<td>24 bit control word</td>
<td>24 bit control word</td>
<td>24 bit control word</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>carrier</td>
<td>ADC0</td>
<td>16 bit ADC</td>
<td>16 bit control word</td>
<td>16 bit ADC value</td>
<td>16 bit control word</td>
<td>16 bit ADC value</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>carrier</td>
<td>ADC1</td>
<td>16 bit ADC</td>
<td>16 bit control word</td>
<td>16 bit ADC value</td>
<td>16 bit control word</td>
<td>16 bit ADC value</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
SPI

• todo:
  - write generic SPI functions and get basic SPI peripheral working
  - write wrappers for each of the devices to use the peripheral
  - write code snippet to detect carrier revision (E2 vs E3/E4)
  - get it working at the b2l "get_status.py" level