Tony's suggestions for SGIR Requirements based on the Spex Aladdin Array

This document summaries the parameters used in the SPEX controller based on the ixthos DSP board. This is intended as input for creating the requirements document for the SG-IR controller.

This document describes concepts about Spex Array Controller Parameters:
http://irtfweb.ifa.hawaii.edu/~spex/computers/userdocs/usersguide_vol_2.pdf

AboutPatters - provide an description of spex's clocking patterns & data tables used in the controller: http://irtfweb.ifa.hawaii.edu/~spex/computers/techdocs/patterns/AboutPatterns.html
Review of **BitSummary.html** and **PatternSummary.html** are relevant!

1. Parameter Specifying an Aladdin Image:

Controller should support these command for specifying readouts.

Itime - The integration time, or time between Reset&Sample (not the pause time between Array readouts). Integration time input range is 0.0001 to 1800.000 seconds, millisecond resolution. (rayner requested 3600 max). Actual minimum time is the time to clock out the Array.
Time to wait is based on CDMode..

Coadd - coadds of integrations (1 to 32000) (rayner requirements is 1000 coadds).

Subarrays - Support up to 3 subarrays.

- Subarray array are specified by: x y wid hgt.
- Subarray can overlap, it the job of the controller to read out the pixel specified by the sub array parameters.

cbmode - clock buffer mode controls reset-sample scheme.
This mode controls the method used to produce a single exposure (an "image).
The ARC mode is also known as fowler sampling.

- ARC_S - array_reset_clocking single.
  1 coadd is: [Reset](integrate)[SamplexNDR].

- ARC_D - array_reset_clocking Double, or 1 coadd is
  [Reset][Pedestal(xNDR)](integrate)[SamplexNDR]

- CDS_PS - Corrlated Double Sample, Pedestal - Sample
  - (Read-Reset on each sample). NDR not supported.
  - When reading the array, we sample each pixel, reset it, and sample the pedestal values. NDR are not supported.
  - The aladdin array support only row-pair resets for CDS_PS. This means readout 2 row (sample), reset row pair, readout the pedestal values. The H2RG array supports resetting by pixels.

[SAMPLE1, RESET1, PEDESTAL1]
(wait)
[SAMPLE2, RESET2, PEDESTAL2]   // 1st coadd is SAMPLE2-PEDESTAL1

ndr – Number of successive non-destructive reads done during ARC mode to obtain a sample or pedestal images. Also know as Fowler sampling. Spex accepts up to 128 (but 32 is the most spex really uses).The number of NDRs are reduced to achieve the requested itime exposure.

Bigdog default NDR is 32.
GuideDog default NDR is 32.

Slowcnt - A way of adjusting or slowing the speed of the clocking wave form.
fast mode is 125 ns resolution clocks.
slow mode is 150 + (25 * slow_cnt) ns resolution clocks.

Bigdog default slowcnts is 20 (650 ns resolution)
GuideDog default slowcnts is 12 (450 ns resolution)
Frame rate to 512x512 is 0.235 seconds (1.1 MPixels per second, or 139KPixels/s per ch)

Fastmode Convert-to-Convert: 15 clock * 125 ns = 1875 ns.
1875 ns per channel, 533K samples per seconds per channel.

2. Movie mode.

Individual Images streamed to disk with virtually NO DELAY between exposes.
Minimum integration time a function of array size (or data transfer speeds).
Time of Move Event should be limited by disk space.

2048 x 2048 * 4 bytes = 16777216 or 16MBytes.
16 x 10   = 160 MBytes to hold 10 images
2 buffers = 320MBytes.

Using double buffer, we can acquire 10 images into a buffer, the controller can then switch buffers. When using subarray, move images can be buffered. When the controller switches buffer, the PC would readout the data.

Slowest link could be GB Ethernet at 26 MB/s.
Does stargrasp support transfers to PC, while accumulated data in another part of DRAM?

3. Global Reset.

The aladdin device support a Global Reset of the array (as opposed to clocking to each pixel and resetting). The parameters GResetNS specifiy the time of the global reset in nanoseconds. Range is 250 to 250,000,000 (0.25 sec). Resolution is 25 ns.

global_reset:
sets: CLOCK_VRSTG | CLOCK_VDDCL | CLOCK_VGGC
wait

clear all bits.

4. Background Reset

When idle (no being requested to take an image) the controller issue background resets to the array using Global Resets or by Clocking out the array. An example of spex’s parameters:

BGR {off|on}   - Turn off/on this feature. Default is ON.
BGR.min.ms ms - Minimum period between a global reset & the start of the GO.Default is 250 ms.
BGR.ms ms   - Period for background resets. Default is 2000 ms.
BGR.ns ns   - Specifies the time the global reset is held during a backgroup reset. Default is 40000

Resolution is 25 ns.

The Array controller should also support clocking out the devices while resetting each pixel using the same clock patterns (and timing) for the array readout. Needed for the H2RG array.

Basic support would include:
A period for the background resets (in millisecond resolutions) Typical value is 1-5 seconds..
A period between a background reset and the start of any data acquisition (A GO).
The reset can be done using global lines or an Array Clocking Sequence.

5. Graceful Aborts

Ability for user to abort an operation, ie: opps I did a GO with 1000 coadds and 300 second integration time. Can immediately abort and continue?

6. Time stamping of the data

A method is required to time sample clocking events, such as when the resset occurred for the image at a microsecond resolution. The time of the clocking should be under stood so that the time that the pixel are address and sample can be calculated using this GPS timestamp.

Spex is able embedded a GPS timestamp for each image. And provide timestamps for each ‘2D’ frame of a 3D movie image.

At minimum, a digital output that can trigger latching and recording of GPS timestamps.

7. Pseudo controller code

Some example pseudo code to illustrate how the controller clock and samples the data for an image acquisition, or GO.

```
expose_arc()
```

```
{
   // Array Reset-Clocking
   // arc_s :([RESET] ...[Sample]) x coadds
   // arc_d :([RESET][Pedestal] ...[Sample]) x coadds

   for( coadd = 0; coadd < g->coadds; coadd++ )
   {
      // if( coadd==0 )
      //    Timestamp data

      // Rest the ARRAY

      if( g->cbmode == arc_d )
      {
         // Clock out Pedestal Frame * NDR;
         for( ndr=0; ndr < g->ndr; ndr++)
         {
             Clock_Array()

            // if( coadd == 0 && ndr==0 ) write value to buffer,
            //                            else subtract value to the buffer.
         }
      }

      // Wait itime-(Frame_ms*NDR).

      // Clock out Sample Frame * NDR
      for( ndr=0; ndr < g->ndr; ndr++)
      {
         Clock_Array()
         // if( g->cbmode == arc_s && coadd==0 && ndr==0)
         //    write value to buffer
         //  else
         //    add value to buffer
      }
   }
}


expose_cds()
{
   CDS: Correlated Double Sample.

   for( coadd = 0; coadd < g->coadds+1; coadd++ )
   {
      // if( coadd )
      //    Wait itime-Frame_ms.  - no wait on 1st iteration.
      // else
      //    timestampe data. - time stamp on 1st iteration.

      // Clock out Array using row/pixels reset
      // if( coadd == 0 )
      //    1st:  subtract pedestal, ignore sample
      // else of (coadd == g->coadds )
      //    last: ignore pedestal, add sample
      // else
      //   middle: subtrack pedestal, add sample

   }
}

FOR Movie mode, we could do something like:

expose_movie()
{
   for ( buf=0, i=0; /* forever*/ ; i++; buf = i%2)  // buf is 0 or 1.
   {
      // Wait until databuffer[buf] is ready.

      // Fill up the data buffer with successive images.
      for (movie=0; g->movie; movie++ )
      {
         // User databuffer[movie]
         if( g->cbmode )  // cbmode is ARC_S, ARC_D or CDS
            expose_arc()
         else
            expose_cds()
      }

      // Notify the host that the images are ready for transfer.
      // This host should be able to transfer the image while we continue
      // data acquisition using the other buffer.
   }
}

// Sample up the Ramp – not current implemented in spex, but illustrated here:

Expose_up_the_ramp()
{
   Reset Array()

   for( rcnt=0; rcnt < g->rcnt; rcnt++ )  // rcnt is the number of ramp samples; min=2.
   {
      // wait so total exposure (reset to last_sample) equal itime, or
      // wait( (itime – (frame_ms * rcnt)) / (rcnt-1)

      Clock_Array();
      // pixel values. Write to new RAM buffer.
   }
}

// rcnt images ready for transfer to the PC.
```

8. Summary of IRTF Device for the SGIR controller.

NSFCAM2 :
- 2048x2048 H2RG
- 32 outputs.

Bigdog:
- 2048x2048 H2RG
- 32 outputs.

Guidedog:
- 512x512 Aladin II
- 8 outputs.

icshell spectragraph:
- 2048x2048 H2RG
- 32 outputs.

icshell guider:
- 512x512 Aladin
- 8 outputs.

9. Things to discuss....

8.1 Subarray... array size W H, and subarray X Y W H.

8.2 Have some concerns about CLV commands (or clocking in general).

- aladdin group row in 4 sequences called ABCD. Four slightly different ROW clocking patterns for the device. The 4 sequence are actually 2 row-pairs, giving a total of 6 variations in the row clocks (7 including initialization the slow MUX).
- For subarray, we have 1 pattern for slipping to the desired column (clocking but no sampling); then another inside the subarray (clocking with samples).
- For pixel level resets, you would need to clock the array with the same pattern (timing) used for sampling, but toggle the reset line (not sample any data).

8.3 The expose/readout command don't fit the infrared array (expose controls the shutter but readout samples the data). May solved by implementing new command based on #7 pseudo controller code

8.4 The single tasking nature of the stargrasp software.

- Array expose can be long (many minutes). Timeout on “expose” is any example of a controller command making the system busy..IO is not available, although the 'receiver' process can monitor things. Possible solution: host application uses 'command' socket to command controller but monitor using 'receiver' socket.
- How are background resets going to be handled?

8.5 Command replies:

- If receiver is connected, the all feedback (serial & telnet) get sent to receiver socket.
- When “busy” all IO is stop.
- How to send aborts.

Libstargrasp/grasp_save.c – assumes sizeof(short), what about int32 for coadd*ndr? Can math engine work on 32 bits?

Can stargrasp transfer data & clocking array simultaneously?
How can movie mode or double buffering work?
Can the ‘math engine’ operator on the DRAM data (read, math operation, write), while data is being transfer from DRAM?

Have you ever looked into running a RTOS on the hardware?