

Software Array Control and Image Acquisition for the NICI¹ instrument

¹ Near Infrared Coronagraphic Imager (NICI) is a Gemini compliant device to gather data regarding stars and other fun stuff.

Table of Contents

Introduction	4
1. Gemini Requirements	6
1.1. Use Cases: The Scenarios	7
1.2. High Level Control Interface -- Acquire Command Object.....	8
1.2.1. Subarray Specification	8
1.2.2. Exposure Mode Selection	8
1.2.3. Set Non-Destructive Reads	8
1.2.4. Set Reset To Pedestal Time	8
1.2.5. Set Integration Time.....	8
1.2.6. Set Number Of Exposures.....	8
1.2.7. Set DHS Sequence Attribute	9
1.2.8. Set DHS Mode	9
1.2.9. Start Exposures	9
1.2.10. Terminate Exposures After Current Exposure.....	9
1.2.11. Terminate exposures immediately.....	9
1.3. The Primitives:.....	10
1.3.1. Resets	10
1.3.2. Subarrays.....	10
1.3.3. Pedestal frames	10
1.3.4. Co-addition.....	10
1.3.5. Flat correction	11
2. Control and Information Flow	12
2.1. Clocker	13
2.1.1. Resets	13
2.1.2. Clocking pattern	13
2.1.3. Image scans.....	13
2.1.4. Voltage levels.....	13
2.2. SL240	13
2.3. Router.....	13
2.3.1. Buffer Pool for data, co-addition and pedestal correction.....	14
2.3.2. Averaging.....	14
2.3.3. Flat correction	14
2.3.4. Flat creation	14
2.3.5. Reordering	14
3. Software Organization	15
3.1. Implementation modules	15
3.1.1. Acquire	15
3.1.2. Clocker	15
3.1.3. Image Router	15
3.1.4. Out	15
3.2. The Specification Commands	15
3.2.1. Clocker	15
3.2.1.1. Image Analog control.....	15
3.2.1.2. Time sync	16
3.2.1.3. Download Clocking Pattern	16
3.2.1.4. Set Scan Mask (for external synchronization).....	16
3.2.1.5. Reset array	16
3.2.1.6. Scan – active	16
3.2.1.7. Scan – Inactive	16
3.2.2. SL240.....	16
3.2.3. Image Router – Capture specifications.....	17

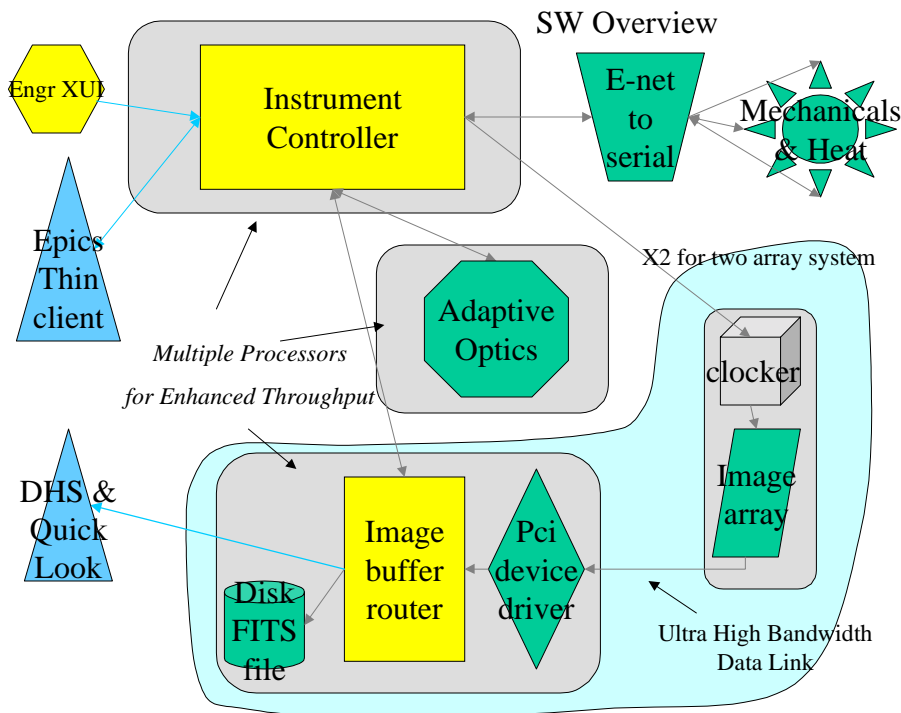
3.2.3.1.	Image configure size.....	17
3.2.3.2.	Image new	17
3.2.3.3.	Image flush	17
3.2.3.4.	Image set <buffer>.....	17
3.2.3.5.	Image addto <buffer>	17
3.2.3.6.	Image add <buffer>	17
3.2.3.7.	Image add <buffer> average	18
3.2.3.8.	Image subtractfrom <buffer>	18
3.2.3.9.	Image subtract <buffer>	18
3.2.3.10.	Image subtract <buffer> average	18
3.2.3.11.	Image pedestal <buffer>.....	18
3.2.3.12.	Image multiplyby <buffer>	18
3.2.3.13.	Image inverse <buffer>.....	18
3.2.3.14.	Image inverse <buffer> average.....	18
3.2.3.15.	Image sendto <destination>	18
3.2.3.16.	Image report <string>	19
3.3.	Command Preparation for Imaging.....	19
4.	Data Destination Handling	24
4.1.	Destinations.....	24
4.2.	Destination commands.....	24
4.2.1.	Destination Open	24
4.2.2.	Destination Close	24
4.2.3.	Destination New Image.....	25
4.2.4.	Destination Data Conversion	25
4.2.5.	Destination Configure Map	25
	Destination Decode	25
4.2.7.	Destination Header Items	25
4.2.8.	<destination> out	27
4.2.9.	<destination> map	27
4.2.10.	<destination> report <string>.....	28
5.	Status Monitor	29
6.	Appendix A: Sampling Modes And Low Level Actions	30
6.1.	Background Resets.....	30
6.2.	Single Sampling readout	30
6.3.	Fowler Sampling readout (NDR=1 = CDS).....	30
6.4.	Fowler Sampling Multiple Nondestructive Readout (#NDR=n).....	31
6.5.	Coaddition (#COADD=m), Fowler Sampling readout (NDR=1 = CDS).....	32
6.6.	Coaddition (#COADD=m),Fowler Sampling Multiple Nondestructive Readout (#NDR=n). 32	
6.7.	Multiple Coadd buffers (#COADD_BUF=a,b,c,d,e,f,g,h.....	34
6.8.	Streaming mode.....	34
7.	Appendix B: Terms	35
7.1.	Acquire	35
7.2.	Classes and objects	35
7.3.	Clocker	35
7.4.	Command Object	35
7.5.	Image	35
7.6.	Objects and classes	35
7.7.	SL240 Controller	35
7.8.	Subarray (also Region of Interest, ROI).....	35

Introduction

This document describes the instrument level control (IC) to acquire an image through the Aladdin III imaging array. Here is the discussion of the user level commands to gather the image, the software modules involved and the sequencing of information among those modules. These modules are located internal to the Instrument Controller in the IC processor and the Image Buffer Router in the Pixel Server as shown in Figure 1 below.

The imaging function of the MKIR² controller is to gather information from an image detector and combine that information with ancillary data and prepare the images for subsequent computer processing and human viewing. Some of the ancillary data is in the form of camera and telescope set-up information. This ancillary data is used to form the final DHS³ (or FITS⁴ file)

Figure 1 – NICI SW modules inside Image Buffer Router and Instrument controller



information, and is outside of the primary scope of this document.

This document will not describe any of the mechanical devices in the optical pathways, steering mechanisms or environmental control of the imaging arrays. The control and monitoring of the telescope is not discussed here, rather the details of obtaining the image and preparing it for

² Mauna Kea InfraRed, Inc. The designer and builder of the NICI instrument.

³ Data Handling System (DHS) is the Gemini provided database handler for images generated by NICI or other Gemini compliant instruments.

⁴ FITS, an image file format that is standard for the astronomical community.

Ver 1.4

external processing or human viewing are covered below. Neither will this document describe the interface to the Gemini data handling system.

1. Gemini Requirements

The key requirements for NICI are to interface to Gemini's EPICS system through a "thin-client", thereby allowing complete control of all contractually described scientific uses. Images are to be sent to the Gemini Image handling system: DHS. The imaging and timing requirements as described in Gemini Top-level performance requirements (SDN1003) are:

1. Must operate two 4 quadrant Aladdin type III style arrays
2. Connect to Gemini through a Socket for remote control
3. Must allow synchronization of readouts of the two arrays to 1 millisecond.
4. Single subarray mode minimum size 8x16 placed anywhere in the array and reflected to all four quadrants.
5. Global reset Single sampled readout mode.
6. Double correlated sampled readout mode.
7. Multiple NDR noise-reduction sampling mode.
8. State set and state read commands.
9. Populate the FITS header and ship the data to the DHS.
10. Must operate in a standalone mode or under Gemini control in a remote mode.
11. Must provide an image display in standalone mode.
12. Must provide local storage for standalone mode.
13. Must time stamp frames.
14. Must have macro capability in stand-alone mode.
15. Must be able to co-add data into a minimum of two buffers.

While the body of this document covers the grand software architecture, some specific requirements are answered directly.

1. Must operate two 4 quadrant Aladdin type III style arrays
 - HW clocking and fiber interface to image acquisition "pixel servers" described below.
2. Connect to Gemini through a Socket for remote control
 - Use of TCP/IP Server sockets via TCL
3. Must allow synchronization of readouts of the two arrays to 1 millisecond.
 - Time of acquisition standardized through HW clocking mechanism
4. Single subarray mode minimum size 8x16 placed anywhere in the array and reflected to all four quadrants.
 - Subarray specification addressed below.
5. Global reset Single sampled readout mode.
6. Double correlated sampled readout mode.
7. Multiple NDR noise-reduction sampling mode.
 - Modes and attributes passed through Acquire SW module
8. State set and state read commands.
 - All appropriate internal state is settable/readable
9. Populate the FITS header and ship the data to the DHS.
 - Fits and DHS operation described outside of this document
10. Must operate in a standalone mode or under Gemini control in a remote mode.
 - Universal socket allows remote and engineering access via same thin-client protocol
11. Must provide an image display in standalone mode.
 - Use of standard X-windows DV image viewer in stand-alone and engineering mode.
12. Must provide local storage for standalone mode.
 - Use of local disk storage is standard.
13. Must time stamp frames.

Ver 1.4

- Time of image capture specified and supplied to Hardware Clocking subsystem
- 14. Must have macro capability in stand alone mode
 - Macro capability in all modes through use of TCL as implementation language.
- 15. Co-add capability
 - Co-addition and pedestal modes are designed into the image array processing modules.

The MKIR controller is used to get pictures of stars in the infrared wavelengths of light. The driving requirements for image acquisition are given by the scientific uses of NICI, hence the requirements are derived from these use cases. When these use cases are investigated for the purpose of analyzing and developing the software to gather this image, the actual subject matter is irrelevant. What is important is the details of generating the control signals to gather the image from the imaging device (currently an Aladdin III) which is controlled by a newly developed hardware device called a 'clocker'. The raw image data is transmitted back to the MKIR controller by way of high-bandwidth fiber-optic cables to an I/O board, called the 'SL240' device. MKIR desires high throughput rates. That means that overlapped, independent processing of the control and data pathways is important. The separation of instrument control from clocking is part of this separation. Even more important is the separation of the image capture from the main control. The model requires a distributed processing system, with one processor to handle the clerical details and a second separate computer (or more, if required) to acquire the data and prepare it for storage, downline processing or human viewing. The controlling computer is called the IC (instrument controller), the computer intended to capture the image is the PS (Pixel Server.) NICI has one IC to control two PS processors, one for each imaging array.

1.1. Use Cases: The Scenarios

An image of a distant object is focused on the Aladdin array. The MKIR controller is initialized and ready to gather this image. This image is to be scanned in its entirety, sent to the pixel server, which then decodes and saves it for viewing. The exact time of the image capture is needed, as well as other information, where the telescope is pointing, the filters in the light path. These pieces of information are generally useful and will be discussed in the section FITS File Header or DHS description.

Each of the Use Cases described by the science drivers boil down to a number of primitive imaging operations or PIOs. These PIOs are sequenced and repeated with various parameters (length of integration, buffer selection, final destination) to achieve the true goals of scientific discovery. The PIOs are described in Section **Error! Reference source not found.**

- The user (Instrument sequencer or higher level entity) points the telescope at the desired object.
- The user sets an integration time.
- The user sets an exposure mode.
- The user sets an integration count.
- The user sets a repeat count.
- The user selects a storage mode (DHS, quicklook, or other)
- The user sets the subarray (region of interest) configuration.
- The user starts the exposure. The NICI begins collecting data, sends back data and signals when done.

The exposure modes are detailed in the Appendix and may affect the semantics of the integration count and repeat count.

1.2. High Level Control Interface -- Acquire Command Object⁵

At the level of the Instrument Sequencer, the 'acquire' command object is the main tool for getting images. There is to be one Command Object for each imaging array, Acquire1 and Acquire2 and are otherwise identical. The command set for these objects is described below. The details of the information flow within NICI are described in the "router" section.

1.2.1. Subarray Specification

```
acquire configure -subarray {list of subarray specifications }
```

Example:

```
Acquire1 configure -subarray { {0 0 20 40} {200 100 16 16} }
```

This sets two arrays starting at with upper left corner at (0,0) and lower right corner at (20,40), the other with upper left corner at (200,100) and lower right corner at (216, 116). The X dimension increments from left to right, the Y dimension increments from top to bottom.

1.2.2. Exposure Mode Selection

```
acquire configure -mode <modetype>
```

The legal modetypes are: arc_s (single image acquisition after array reset) or arc_d (image acquired after array pedestal noise subtracted off)

Example:

```
Acquire1 configure -mode arc_d
```

1.2.3. Set Non-Destructive Reads

```
acquire -configure ndr <ndrcount>
```

- ndrcount: the number of times to acquire data (adding into current image) between array resets.

Example:

```
Acquire2 -configure ndr 3
```

To set 3 non-destructive reads.

1.2.4. Set Reset To Pedestal Time

```
acquire -configure rtime <time>
```

-specifies the time in milliseconds to wait after the array reset before the pedestal data is read from the array.

Example:

```
Acquire2 -configure rtime 20
```

For a 20 millisecond delay after reset before a pedestal scan is to be taken.

1.2.5. Set Integration Time

```
acquire configure -integrate <time>
```

-specify the time in ms to expose array for each image scan

Example:

```
Acquire1 configure -integrate 1500
```

To integrate the arrival of photons on the imaging device for 1.5 seconds.

1.2.6. Set Number Of Exposures

⁵ A *command object* is a top-level interface module for the NICI software. User level commands are given to any of several command objects. These command objects have similar interface behavior and are used to coordinate with lower level modules that may actually perform the functions requested.

Ver 1.4

`acquire configure -exposures <count>`
-specify the number of exposures to perform for next "go" command.

Example:

```
Acquire2 configure -exposures 10
```

Specifies that a sequence of 10 complete reset-integrate-capture cycles is to be taken.

1.2.7. Set DHS Sequence Attribute

`acquire configure -sequence <identifier>`
specify the DHS identifier for this data set.

Example:

```
Acquire2 configure -sequence A333
```

The DHS sequence identifier is generated by a request for a cookie from the DHS system. This is done by the Instrument Sequencer (IS) epics interface processor as part of its normal operation.

1.2.8. Set DHS Mode

`acquire configure -destination <destination list>`
-specify the destination for the following image acquisition. The list of legal destinations are: quicklook, data dest, {fitsfile <file prefix>} {fitssocket <legal socket specification> }

Example:

```
Acquire1 configure -destination quicklook
```

Specify the image is to be sent to the quicklook system (a DHS subscriber)

1.2.9. Start Exposures

`acquire go`
start acquiring data

Example:

```
Acquire2 go
```

This will start taking the images. Images will be collected using the number of exposures, reset times, etc that have been set with the 'configure' commands.

1.2.10. Terminate Exposures After Current Exposure

`acquire stop`
stop acquisition after current image scan.

Example:

```
Acquire1 stop
```

Stop getting data after the current integration. If the integration time is excessive, this may take a while.

1.2.11. Terminate exposures immediately

`Acquire abort`
stop acquisition sequence immediately. This will result in a user level error condition. The NICI instrument must be reset before further commands will be accepted.

Example:

```
Acquire2 abort
```

Stop. Do not pass go. Do not collect \$100.

The NICI instrument will stop collecting data immediately and enter an error state.

1.3. The Primitives:

The acquire command object uses several lower level objects to perform the imaging process. The CLOCK object controls the hardware clocker. The receipt of image data is handled by the ROUTER client (in the IC processor) which coordinates with its counterpart ROUTER server in the pixel server. The back-end destinations are handled similarly by a destination client in the IC processor which co-ordinates with the destination server in the pixel server. This is intended to allow scaling of the capabilities of the software from large to small. The descriptions below specifically denote commands to the client side operation. In NICI, the server side objects will have a 1 to 1 exact duplication of the command structure. Since there are two imaging arrays, the software modules are duplicated with the identifiers as in **Error! Reference source not found.**

Table 1 – SW Modules For Control Of The Two Imaging Arrays In NICI

Imaging Array	High Level SW Module	Low Level Modules
Left	Acquire1	IDAC1
		CLOCK1
		IMAGE1
		OUT1
Right	Acquire2	IDAC2
		CLOCK2
		IMAGE2
		OUT2

1.3.1. Resets

The light coming on the imaging array is converted to electrons on the imaging sensors of the Aladdin array. These electrons stay there and will overexpose the image as well as causing other aberrations. The system will flush these extra electrons at continual and specific times. This is called 'background reset' and there is an additional reset just before the image is scanned (so that an exact duration of integration is known). Note that the time of integration of any particular pixel depends on how many pixels are ahead of it in the scan sequence as well as how fast each pixel can be scanned. This information can be computed downstream if needed, so only the time between last reset and first pixel acquisition is used to compute the duration of integration.

1.3.2. Subarrays

Some areas of the image may be deemed more important than others. To increase the speed of scanning, the important areas can be identified as subarrays. A specific scanning sequence (and the corresponding decoding sequence) must be generated that will read out only those desired regions, discarding the data from unwanted regions. The scanning sequence is called the 'Clocking Pattern' and must be sent to the clocker as needed. The image comes off of the Aladdin imaging device in a non-intuitive and hardware dependent order, thus, the image decoding sequence must correspond to the clocking pattern. This decoding sequence is sent to the pixel server to reorder the image into the rows and columns of the image.

1.3.3. Pedestal frames

When the Aladdin III device is reset to clear excess electrons, the reset itself generates some small number of electrons on the surface of the image array. If we record this noise immediately after the reset operation, we can subtract it from the subsequent scan. This results in a sharper, less noisy image. This scan is taken after the 'rtime' parameter (specified by the 'acquire configure -rtime' command described above.) Subsequent scan(s) will have this data subtracted out. Usually, if not always, each reset may need a separate pedestal. Note that the subarray specification will NOT change between pedestal and subsequent scans.

1.3.4. Co-addition

Ver 1.4

Multiple scans of the same image may be taken. These scans will result in data that needs to be accumulated (add OR subtract) into separate buffers. When all the scans are taken, the buffers may be combined by averaging, adding or subtracting, or may be written out to the final destination separately.

1.3.5. Flat correction

This is one of the standard imaging techniques for some applications. There are three images that are involved in generating the desired image. These are identified as 'flat', 'dark' and candidate. The result image is made of pixels where each pixel P is computed as:

$$P = (C - D)/(F-D)$$

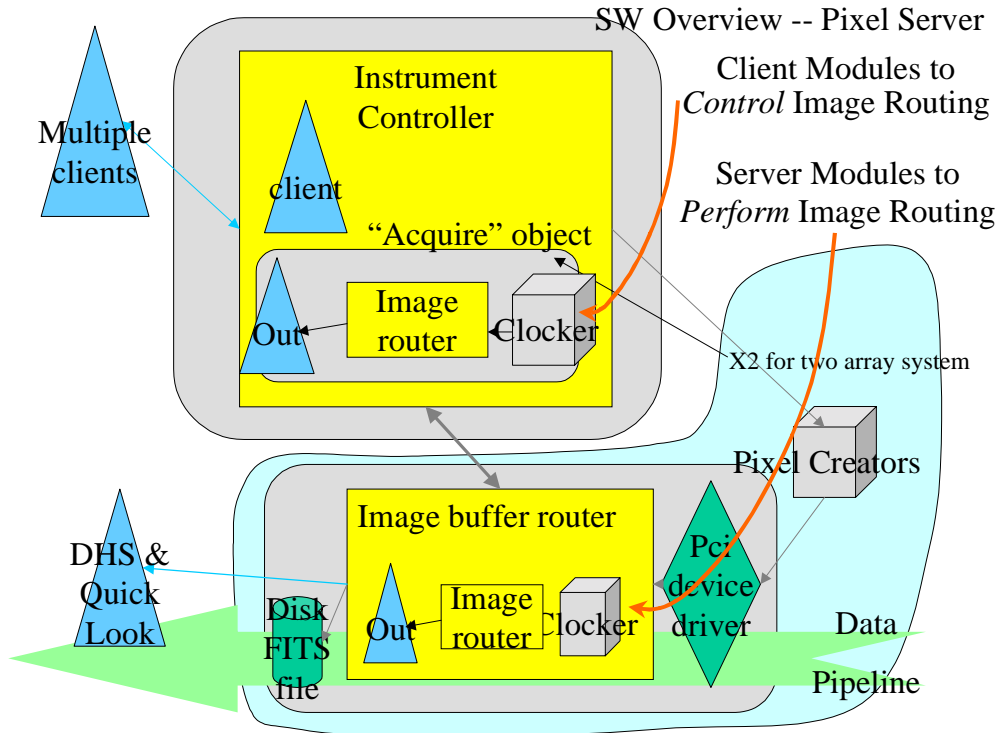
C, D and F are the corresponding pixels from Candidate, Flat and Dark. This can be made slightly faster by generating $F' = 1.0/(F-D)$, and computing:

$$P = (C - D) * F'$$

2. Control and Information Flow

The flow of commands and status responses generally goes from top left (the user) through the IC, flowing down into the command objects and back. The imaging system will generate rather large amounts of data that is completely separate from the control information. There are two

Figure 2 – hardware organization of NICI and corresponding SW modules
 Entities within the yellow region are software analogues of real-world



imaging arrays in NICI, controlled by two completely separate but identical software structures. To simplify the discussion, we will simply talk about one of these and omit the other. Be aware that there is an 'acquire1' and 'acquire2' as well as lower level modules such as 'IDAC1' and 'IDAC2', 'CLOCK1' and 'CLOCK2'.

The general flow of this large amount of data is from the imaging array, through the pixel servers, and into the DHS. Within the two pixel servers, the modules that carry the data are part of the server side `image` and `out` modules. These modules receive commands from their IC counterparts, and perform a limited range of operations at high speed.

Each scan of the Aladdin array results in some amount of data. The size in bytes of this data can be precomputed from the number of pixels scanned plus any hardware overhead. The SL240 driver will be set up to receive this number of bytes and tag it with a scan number. The data will pass into the `image` module, which will have a script dictating the actions for that scan. As a result of performing those actions, the data may pass to the `out` or destination object, which will have instructions for that image.

A useful analogy is the routing freight cars in a train yard. The yard worker consults a clipboard containing routing instructions for each car. By noting the identifier on the side of the car, the yard

worker can direct the car to the proper track. The instructions are created prior to the car arriving in the train yard. The yard worker for NICI is composed of the server modules, "image" and "out" in the pixel server. The IC's acquire uses the client modules, "image" and "out" in the IC to generate these instructions. The organization of our system is made simpler by the observation that the images are generated sequentially. The train yard simplifies to a conveyer belt or pipeline. This pipeline or conveyer belt model provides simplicity, flexibility and scalability. Each station does some simple and independent processing of the data and moves it down to the next station. As the bandwidth of the total pipeline grows, the use of parallel pipes with identical processing provides a scaled up solution. Figure 1 shows the hardware of the NICI instrument and the corresponding internal software modules.

The major processing positions are:

2.1. Clocker

This piece of hardware/firmware generates the resets and clocking sequence. It must record the exact time of each reset and scan sequence.

2.1.1. Resets

The electrons in the wells of the imaging device need to be dumped periodically. This gives need to the concept of the background resets. Once the background reset period is set, the imaging array is constantly cleared of electrons. This is required for normal operation of the array and generally will continue even through an instrument level error condition. In addition, the array needs to be reset at the beginning of each desired integration. This is so that there is a known time between the last reset and the scan for pedestal data.

2.1.2. Clocking pattern

The Aladdin III array needs a specific sequence of drivelines raised and lowered to cause data to be read from the imaging wells. This sequence is complicated and exact nanosecond timing is essential. The pattern is built as a result of setting the sub-array specification (ROI). This pattern is sent to the clocker hardware prior to a data scan.

2.1.3. Image scans

When an image is desired, the clocker is commanded to send the clocking pattern to the Aladdin III. The starting time of the scan is also specified.

2.1.4. Voltage levels

The exact voltages applied to the Aladdin III device are critical. Most of these voltages are only settable in engineering mode. However, the user of the NICI device may set some non-critical bias voltages. This is done through the IDAC Command Object.

2.2. SL240

This I/O device dumps the raw image data into the PIXEL Server's memory. The raw image data must be identified as the result of a specific scan. The pixel server receives the data and may up-convert the data from 16 bit format into 32 bit integer or floating as needed for speed (floating point being speedier on Pentium class processors than 16 bit fixed point). It will place the data into buffers in the image module for co-addition or pedestal processing. On specific command it may release one of the buffers to the next station. The SL240 will generate one set of data for each scan.

2.3. Router

The router will perform any final processing on the buffers including averaging or flat correction. It will also reorder the image into human viewable form as required. It will prepare the data for final destination. It will not tag the data with ancillary headers for the DHS or FITS file destinations. The router will consist of and provide primitives for:

2.3.1. Buffer Pool for data, co-addition and pedestal correction

These buffers are pre-allocated to be the exact size for the incoming scans. There will be enough space for at least 32 buffers. A number will identify the buffers.

2.3.2. Averaging

The content of the buffer is replaced by each pixel in the buffer divided by the number of scans (co-additions) that have been added into the buffer.

2.3.3. Flat correction

The data in the buffer is modified by the flat correction as described above.

2.3.4. Flat creation

The data in the buffer is modified into the F^- as needed for use in flat corrections.

2.3.5. Reordering

Map the pixels into a full size empty region for human viewing. This always results in a full size image, no matter how small the subarray specification

3. Software Organization

3.1. Implementation modules

Referring to Figure 1, the command object “acquire” resides entirely in the instrument controller and consists of and communicates with the IC’s `clocker`, `image` and `out` (destination) objects. These three objects communicate with the clocker hardware as well as their counterparts in the pixel server. That is, the clocker communicates with the clocking hardware and the SL240 of the pixel server to create the new image objects. The IC’s router generates a set of commands to the new image object causing it to flow into the pedestal and co-add buffer system, and possibly out to the image destination. The IC’s `out` object creates a set of commands that cause the final image to be sent to the DHS backend or local disk file system (the IC’s destination object may also send information directly to the DHS system).

3.1.1. Acquire

The `acquire` command object checks the parameters for consistency. It then activates the clocker to scan the array and create the image object. The acquire object also activates the router object to perform the pedestal or other image processing. The third SW object that the acquire command object activates is the destination object, “out,” to send the data to the final backend, if necessary. The acquire object will repeatedly activate these objects until all of the scans required by the users request are satisfied.

3.1.2. Clocker

The `clocker` SW object prepares the subarray pattern from the ROI specification and downloads the clocking pattern. It synchronizes the time of day with the HW clocker system. It instructs the clocker to perform the scan at the appropriate time. It also instructs the SL240 receiver of the pixel server to receive and create a new image object. For null images, the SL240 receiver is instructed to create an empty image object without waiting for any clocked data to arrive over the optical fiber.

3.1.3. Image Router

The `image` SW object prepares a script describing the actions required to properly handle the new image object, whether it is to be added to a buffer or sent directly to the data back-end.

3.1.4. Out

The destination object, `out`, will be activated only for images that will result in data being sent out to the final DHS or local FITS file. The destination object will send tagging information to the DHS system such as headers. It will also send commands to the pixel server’s destination object to finalize and route the image data to the DHS and quicklook systems as needed.

3.2. The Specification Commands

3.2.1. Clocker

Clocker commands are case insensitive and in ASCII. Numbers are in decimal and will fit in 32 bits. This interface has a command object `IDAC` and a lower-level `CLOCK` object. The actual parameters for the `CLOCK` object are hardware dependent and are detailed in the Clocker hardware documentation.

3.2.1.1. Image Analog control

The Aladdin III imaging array is controlled by several analog and digital signals. The digital signals are only digital in the control sense, in that there are only two meaningful voltage levels. The actual voltage levels are supplied by commands to the Image DACs and are controlled by configuration commands to the IDAC Command Object:

Ver 1.4

IDAC configure -<DAC identifier> <Voltage>

Example:

IDAC configure -DACBIAS 3.5

The DAC voltages are settable but can not be read back from the actual device. The complete list of DAC identifiers and which are settable in user mode is listed in the hardware documentation.

3.2.1.2. Time sync

CLOCK SYNC

The clocker will reply with its current time. This time will be used to synchronize the IC's actions with the rest of the system. No parameters are needed.

3.2.1.3. Download Clocking Pattern

CLOCK DSCP <size in bytes> <pattern>

The pattern itself will be in the format required by the clocker. The pattern will be composed of byte codes to select the atomic patterns in a sequential manner. The clocker will not be required to accept a scan command until a proper pattern has been loaded. The clocker will, however need to perform resets if possible.

3.2.1.4. Set Scan Mask (for external synchronization)

CLOCK SSCM <mask1> <mask2>

The scan trigger is reset. Mask1 and Mask2 are used to select and detect the external bits that will trigger the scan. Mask3 is the bit pattern to apply to the external output pins while the trigger is active.

3.2.1.5. Reset array

CLOCK RESET <cadence> <duration[JH19]>

Cadence is the number of microseconds between continual (background) resets. If this number is zero, then perform the reset immediately, and only once. Report the time of the first reset.

If the number is -1 then continually reset until the scan mask is satisfied. Report the time of the reset at the time the scan mask criterion is satisfied. Leave scan trigger active.

3.2.1.6. Scan – active

CLOCK Scan <time> [<count> <delay>]

Wait until clocker's internal time is GEQ the supplied time. Scan when the scan trigger is active. Report the time of the scan start.

If count is supplied, perform the scan that number of times (do not wait for external trigger?) with <delay> microseconds between scans.

If the time is zero, wait until the scan trigger is active, and reset the scan trigger after the completion of the scan.

3.2.1.7. Scan – Inactive

CLOCK ISCAN [<unity>]

This does not actually clock the clocker or activate the fiber optic link, but rather creates a completely empty image for the router. The <unity> parameter indicates the value of each pixel position and is 1 or 0. If omitted this value is taken as 0. This is used to initialize buffers or to flush out data to the back end handlers.

3.2.2. SL240

The SL240 will be commanded to read a specific amount of information corresponding to each scan. The data received will be made available sequentially to the next stage of the pipeline, the router. There is no user-level command to the SL240.

3.2.3. Image Router – Capture specifications

Each image scan, active or inactive, results in an image object. That object responds to the following command set. The capture of a frame is designed around the concept of a co-add buffer. The default co-add buffer is buffer 0. This co-add buffer is also the buffer that is considered the “pedestal” buffer. These buffers have a “count” attribute that assists when the data in the buffer is to be taken as the mean of all accumulated data. The data in the buffer is not averaged however, and is always available as the raw sum.

Most operations work directly from the image data. Some, like flat correction, require extra operations on buffers that already hold data.

3.2.3.1. Image configure size

IMAGE CONFIGURE –pixels number

This sets the size of the images to be processed. Once set, this quantity is used until changed explicitly. The maximum value is 1024*1024 for a whole image. This is also the default used if no value is supplied.

3.2.3.2. Image new

IMAGE NEW <dummy>

The <dummy> parameter is optional, and if omitted, this command causes the image routing system to terminate processing of current image buffer and wait for new image from clocking system. This command allows routing system to transition from image to image.

If the parameter <dummy> is supplied, which must be 0 or 1, the new image is not obtained from the SL240, but rather created internally to flush or initialize an imaging sequence. The new image will have all pixel values set to the value of <dummy>.

3.2.3.3. Image flush

IMAGE FLUSH

The flush command causes all buffers to be marked as empty. All buffers will appear as all zero.

3.2.3.4. Image set <buffer>

Image set <buffer>

Copy to co-add buffer (clear and add into co-add buffer) The buffer is marked as receiving 1 data set. If <buffer> is omitted, the default of 0 is used.

```
Buffer[buffer] := image; count[buffer] :=1;
```

3.2.3.5. Image addto <buffer>

Image addto <buffer>

Add into buffer – the buffer is marked as receiving one additional data set. If <buffer> is omitted, the default of 0 is used.

```
Buffer[buffer] += image; count[buffer] +=1;
```

3.2.3.6. Image add <buffer>

Image add <buffer>

Add the data from the indicated buffer to the image. The buffer is unchanged.

```
Image += Buffer[buffer];
```

3.2.3.7. Image add <buffer> average

Image add <buffer> average

Add the mean pixel values from the indicated buffer to the image. The buffer is unchanged.

```
Image += (Buffer[buffer] / (count[buffer]));
```

3.2.3.8. Image subtractfrom <buffer>

Image subtractfrom <buffer>

--Subtract from buffer --The buffer is marked as receiving one additional data set. If <buffer> is omitted, the default of 0 is used.

```
Buffer[buffer] -= image; count[buffer] +=1;
```

3.2.3.9. Image subtract <buffer>

Image subtract <buffer>

Subtract the buffer data from the image object.

```
Image -= Buffer[buffer];
```

3.2.3.10. Image subtract <buffer> average

Image subtract <buffer> average

Subtract the mean of each pixel position of the buffer data from the image object.

```
Image -= (Buffer[buffer] / (count[buffer]));
```

3.2.3.11. Image pedestal <buffer>

Image pedestal <buffer>

Copy negative of image to co-add buffer (for initial pedestal) – the buffer is marked as having one data set. If <buffer> is omitted, the default of 0 is used.

```
Buffer[buffer] := -image; count[buffer] := 1;
```

3.2.3.12. Image multiplyby <buffer>

Image multiplyby <buffer>

The pixel values of the image are multiplied by the corresponding values of the buffer

```
Image *= Buffer[buffer];
```

3.2.3.13. Image inverse <buffer>

Image inverse <buffer>

The pixel values of buffer are divided into one. Division by zero results in zero.

```
Buffer[buffer] = 1.0 / (Buffer[buffer]);
```

3.2.3.14. Image inverse <buffer> average

Image inverse <buffer> average

The pixel values of buffer are divided into one. Division by zero results in zero.

```
Buffer [ buffer ] = count[buffer] / (Buffer [ buffer]);
```

3.2.3.15. Image sendto <destination>

Image sendto <destination>

The image is sent to the next processing element, that is <destination>, which may be the DHS system, of the FILE system.

3.2.3.16. Image report <string>

Image report <string>

The string is sent back to the client (in the IC) for completion or debugging purposes.

Example:

Image report "debug message 1"

Image report "acquisition complete"

3.3. Command Preparation for Imaging

This section shows how the above commands will be used to take an image. For simplicity, we will not show the details of the OUT module commands, although they are completely analogous to the IMAGE commands. We will use the commonly used Fowler Sampling readout sequence.

This sequence is from section 6.3 below. The steps are:

Clock Global Reset

Delay before Pedestal Readout

Clock Pedestal Frame Readout [place in Pedestal buffer]

Delay for ITIME

Clock Signal Frame Readout [place in Signal buffer, subtract Pedestal from Signal]

The "acquire" Command Object will call a procedure based on the integration mode. In this case, "fowlerSampling". That routine as written in TCL would look like:

```
Proc fowlerSampling { } {
  Global integrationTime
  Global destination
  Global pedestalTime

  # reset the array
  CLOCK RESET

  # get the pedestal
  IMAGE NEW
  CLOCK SCAN $pedestalTime
  IMAGE PEDESTAL

  #wait for the integration time and scan
  IMAGE NEW
  CLOCK SCAN $integrationTime
  # bias data by pedestal
  IMAGE ADD 0

  # decode for final viewing and send it out
  IMAGE DECODE 0
  IMAGE SENDTO $destination
}
```

When this script gets executed it will send messages to the 'CLOCK' and 'IMAGE' objects. These objects will each receive their own messages, not the ones for the other. Given the values for pedestalTime, integrationTime, and destination as 0, 1000, and 'quicklook', the two objects will have the following commands:

Commands seen by `clock` object

```
CLOCK RESET  
CLOCK SCAN 0  
CLOCK SCAN 1000
```

Commands seen by `IMAGE` object

```
IMAGE NEW  
IMAGE PEDESTAL  
IMAGE NEW  
IMAGE ADD 0  
IMAGE DECODE 0  
IMAGE SENDTO QUICKLOOK
```

The `CLOCK` object will send these commands to the clocker hardware module for execution. This will result in two full scans of the imaging array. The `IMAGE` object, in the IC, will send these commands to its counterpart in the pixel server. When the two sets of data from the scans arrive at the SL240, the handling instruction will be waiting for them.

The actual information content that the clocker object will see is represented by the commands:

These will cause the clocker to reset the array, scan out a pedestal image immediately. The clocker will wait one thousand milliseconds, and scan a second frame. When these commands complete, the clocker will continue background resets as previously set up.

Commands seen by hardware `clocker` at **1** in Figure 3 -- Information Flow for Fowler Sampling.

```
RESET           1A  
SCAN 0          1B  
SCAN 1000      1C
```

The actual information that is seen by the image router object in the pixel server is:

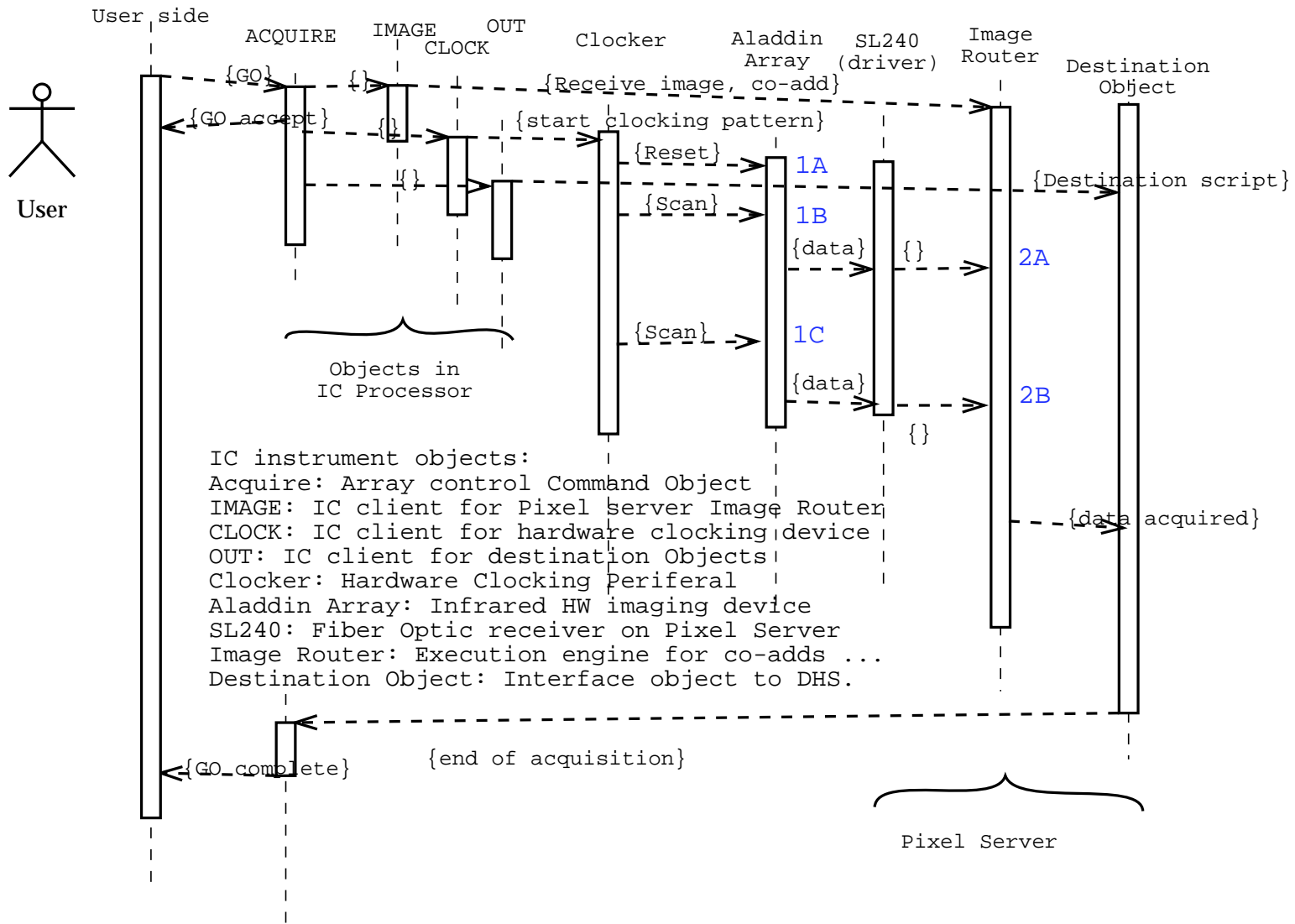
This causes the router to process two new images, one as a pedestal (stored as negative in buffer 0) and the second as data added to the pedestal, then decoded and sent to the quicklook backend. Sequencing of the actions of the clocker and router is maintained because there is one 'new' router command for each clocker 'scan' command.

Image Commands Seen by pixel server at **2** in Figure 3 -- Information Flow for Fowler Sampling

```
NEW           2A  
PEDESTAL  
NEW           2B  
ADD 0  
DECODE 0  
SENDTO QUICKLOOK
```

A complete diagram of the command and response flow up to the destination object within NICI is shown in Figure 3 -- Information Flow for Fowler Sampling. Figure 4 -- Module Structure for Images in Instrument Controller shows the software objects in the Instrument Controller and the objects in the Pixel Servers are shown in Figure 5 – SW Classes in the Pixel Server.

Figure 3 -- Information Flow for Fowler Sampling



User initiates GO command through ACQUIRE to perform fowler sampling.

Figure 4 -- Module Structure for Images in Instrument Controller

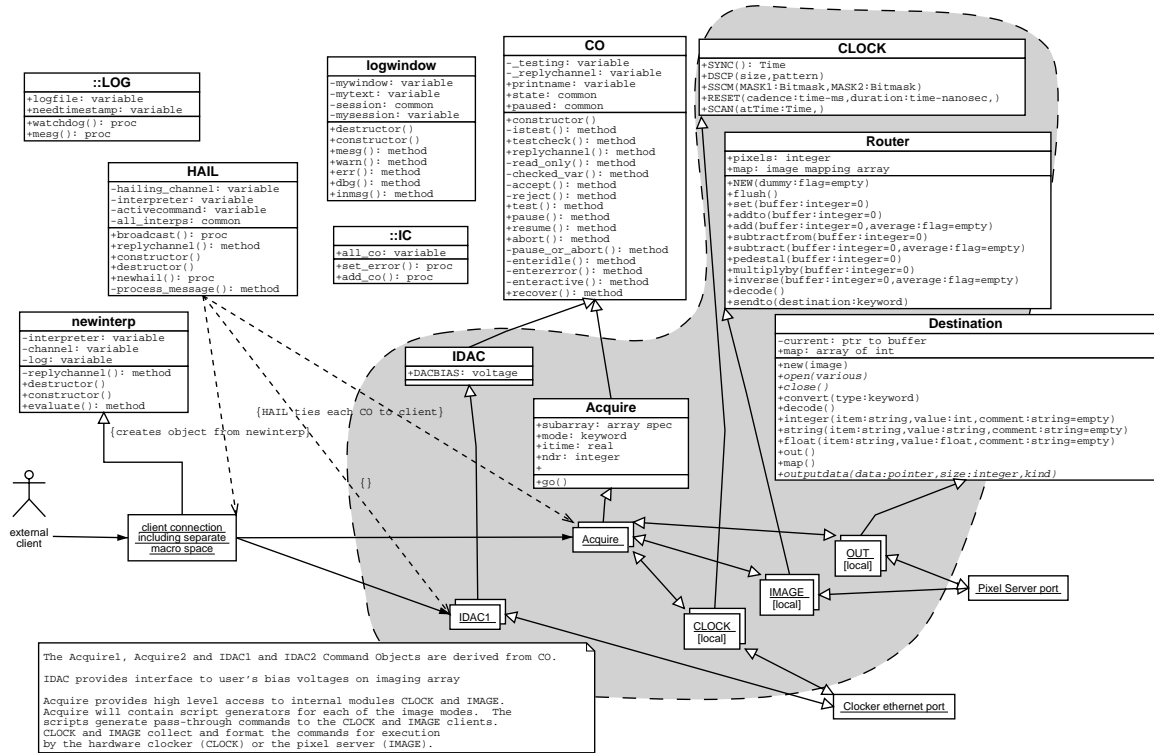
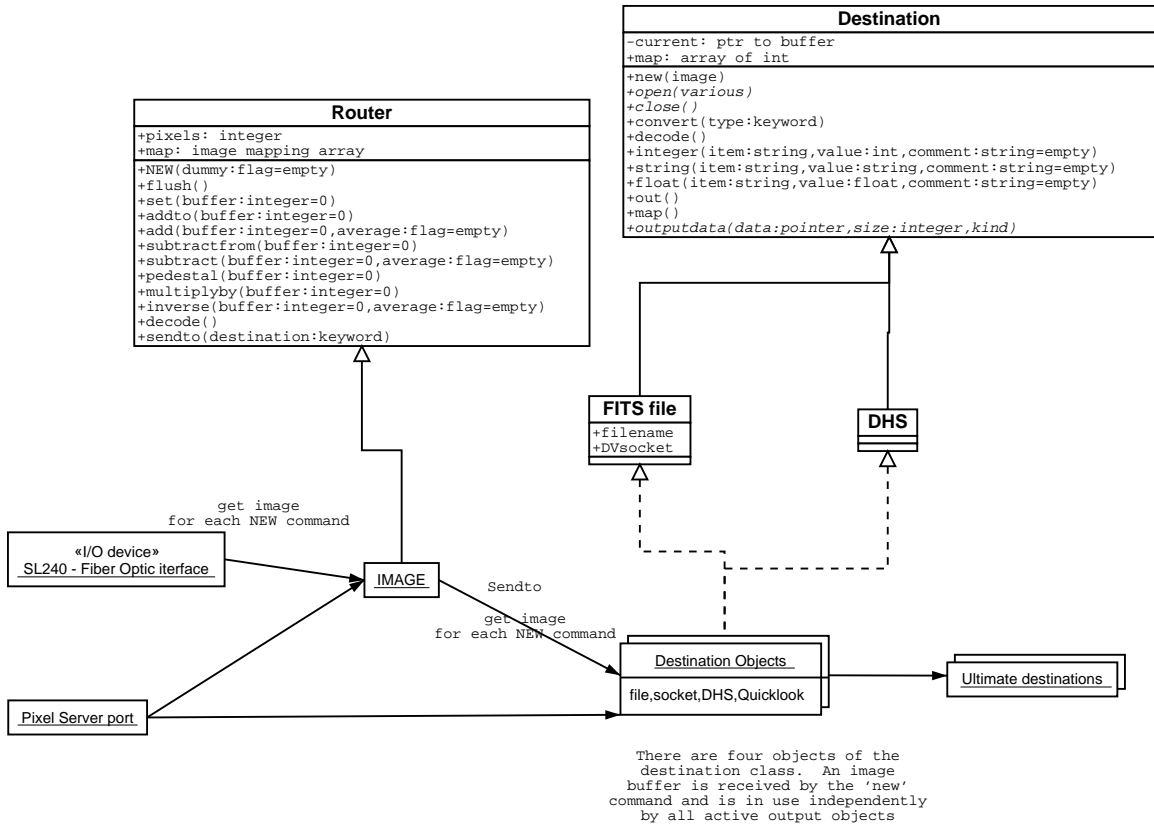


Figure 5 – SW Classes in the Pixel Server



4. Data Destination Handling

Along with the image, specific information needs to be passed along to the final destination. This information consists of exposure conditions, modes, temperature of the imaging arrays, etc. The specific format of this header information depends on the specific back-end desired. For example, FITS file headers will be generated by the cfitsio library, and data for quicklook or DHS handling will be handled by the DHS interface library.

4.1. Destinations

There will be several destinations. A separate object will represent each destination.

Object	
FF	Fits File on Disk
FS	Fits File through socket.
DHS	DHS storage
QUICKLOOK (also QL)	DHS quicklook

4.2. Destination commands

These objects all have a common command set.

4.2.1. Destination Open

<Destination object> OPEN <parameter list>

Close any previous connection to the destination. Then, the object will open an external connection; the <parameter list> will have different meaning for each of the specific destinations: For FF, it will be the local file's pathname.

For FS it will be the IP and port number of the socket application (DV, for example)

For DHS and QL it will be the observation ID

The destination object will remain attached to the external connection until a CLOSE or another OPEN. There is only one external connection per destination object.

Examples:

```
FF OPEN "/dataset/F001.fits"
FS OPEN "enr.greensand.net:3242"
DHS OPEN "object ID"
QL OPEN "object ID"
```

In the case of DHS and QL, the object ID will be a cookie⁶ obtained from the DHS system.

4.2.2. Destination Close

<Destination object> CLOSE

Any previous external connection is terminated gracefully.

Examples:

```
FF CLOSE
```

⁶ A cookie is the term for a data item obtained from an external source, which has special meaning to that external source. It is intended to be used as an identifying token and must be presented in exactly the same format as originally received.

Ver 1.4

FS CLOSE
DHS CLOSE
QL CLOSE

4.2.3. Destination New Image

<Destination object> NEW

The destination object is to wait until a new image (or other data) is delivered to it by the image router's "SENDTO" command.

Examples:

FF NEW
FS NEW
DHS NEW
QUICKLOOK NEW

4.2.4. Destination Data Conversion

<Destination object> CONVERT "INT16"|"CHAR"|"INT32"|"FP"|"NONE" [NET]

The newly received data is converted from internal format to the desired format for the destination, if the optional NET keyword is appended to the command, the data is reordered at the byte level. The conversion NONE delivers the data unchanged and can be used in high-throughput or debugging situations.

Examples:

FF CONVERT INT16
FS CONVERT INT16 NET
DHS CONVERT CHAR
QUICKLOOK CONVERT CHAR

4.2.5. Destination Configure Map

<destination> CONFIGURE -map {map specification}

The raw Aladdin III data needs to be re-mapped for human viewing. A new map needs to be specified when subarrays (ROIs) are defined, and prior to image arrival. This map has one position for each active pixel in the sub-array regions. Each value is an index into the final 1024*1024 image plane. The decode operation will move the pixels in the image according to this map.

4.2.6. Destination Decode

<Destination> DECODE

The image is re-mapped from the Aladdin III pixel order into a 1024 by 1024 region for human or DHS processing. The background pixels are set to zero.

Example:

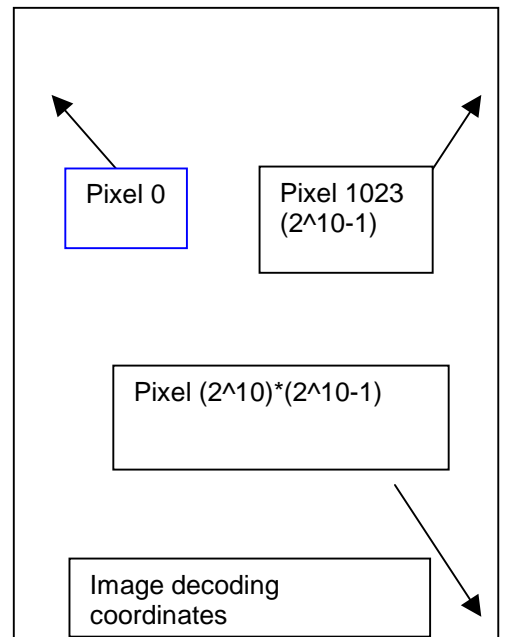
QUICKLOOK DECODE
DHS DECODE
FS DECODE

4.2.7. Destination Header Items

There are 4 subcommands to update header information to a destination channel. These commands correspond to the data type desired in the header.

```
<destination> integer <item> <value> [<comment>]  
<destination> string <item> <value> [<comment>]  
<destination> float <item> <value> [<comment>]
```

The Header item is formatted and sent out to the destination.



Ver 1.4

Examples:

```
QUICKLOOK STRING TITLE HALTOFEMPT "The Celestial Doggie"
DHS INTEGER NAXIS1 1024
FF INTEGER NAXIS1 512
```

The standard header items for DHS and Quicklook destinations are listed in ICD XX.XX (NICI2DHS ICD). This table is duplicated in **Error! Reference source not found.**, and also in **Error! Reference source not found.**. Items that are standard for engineering mode disk files or DV socket streams are in **Error! Reference source not found.**. These tables list the expected minimums and can be augmented by explicit command.

Table 2 -- Standard Header Items for DHS

Name	Type	Units and Range	Comments	Level of Responsibility – IS, IC, PS or GE
NAXIS	integer	1 – 3	FITS standard for dimensionality	
NAXIS1	integer		Size of axis 1	
NAXIS2	integer		Size of axis 2. This is only in the case of 2-D or 3-D data.	
NAXIS3	integer		Size of axis 3. This is only in the case of 3-D data.	
BUNIT	string		Data units.	
UTSTART	string	“UT“	UT at beginning of observation	
UTEND	string	“UT“	UT at end of observation	
EXPTIME	float	seconds	Total integration time.	
ELAPSED	float	seconds	Total elapsed time (including readout time etc).	
BLANK	integer		The value used to indicate a blank pixel	
WFSFOCUS	float	mm	The WFS focus setting	
TITLE	string		Title of observation	
COMMENT[0-n]	string		Comments for the observation	
DETECT	string	TBD	The source of the data	
INSTRUME	string	“NICI ”	This instrument	
OBSTYPE	string	TBD	The observation type.	
OBSID	string		The Observation ID obtained from the OCS	

Table 3 – Standard Header Items for QuickLook Destination

Name	Type	Units and Range	Comments	Level of Responsibility – IS, IC, PS or GE
NAXIS	integer	2	FITS standard for dimensionality	
NAXIS1	integer		Size of axis 1	
NAXIS2	integer		Size of axis 2.	
BUNIT	string		Data units.	
UTSTART	string	“UT“	UT at beginning of observation	
UTEND	string	“UT“	UT at end of observation	
EXPTIME	float	seconds	Total integration time.	
TITLE	string		Title of observation	
DETECT	string	Array1	The source of the data	

		Array2		
INSTRUME	string	"NICI "	This instrument	
OBSTYPE	string	"DARK" "FLAT" "IMAGE"	The observation type. DARK and FLAT are for calibration.	
OBSID	string		The Observation ID obtained from the DHS	
BLANK	integer		The value used to indicate a blank pixel	

Table 4 -- Standard Header Items for FITS disk or Socket Destinations

Name	Type	Units and Range	Comments	Level of Responsibility – IS, IC, PS or GE
ORIGIN				
TELESCOP				
INSTRUME	string	NICI	This instrument	
OBSERVER	string			
OBJECT	string			
COMMENT	string			
IRAFNAME				
BEAM				
TIME_OBS	string		(UT Time of Acquisition)	
DATE_OBS	string		(UT Date of Acquisition)	
TIME_GPS			(UT time from GPS)	
CAMMODE	string	SIM BASIC MOVIE		
ITIME	integer	millisec	(integration time)	
CO_ADDS	integer		(number of integrations)	
CYCLES	integer		(number of cycles)	
NUMARRAY	integer		(how many sub-arrays)	
ARRAY#	string	X Y Wid Hgt	(sub-array description one per array)	
OBSMODE	string		(A or B)	
SUBAB				
BBMODE			(old catcher board modes Not app.)	
CBMODE	string		(old clocker board modes ARC_S ARC_D CDS_PS)	
NDR n	integer		(number of non-destructive reads)	
SLOWCNT	integer		count (slow-down mode during array scan higher means slower scans)	
GPSTMFLG			(GPS time flag)	
GRSTCNT	integer	count	(width of Global reset pulse in 25 nsec periods)	
BGRESET	msec	count	(Background reset attributes milliseconds between resets pulse width)	

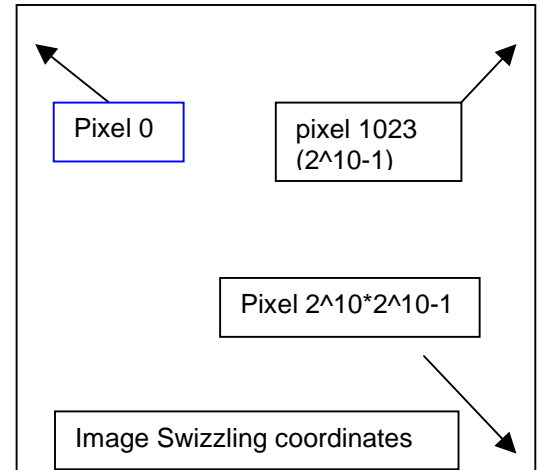
4.2.8. <destination> out

The image data is sent to the final destination. The channel remains open.

4.2.9. <destination> map

Ver 1.4

--Write the image-decoding map to the destination. The decoding map is a sequence of integers in the range of 0 to $2^{10} * 2^{10} - 1$. It is the actual destination in the human viewable image of the sequence of pixels in the buffer. The numbering scheme is x varies faster, y increments to the bottom of the picture. Unswizzled buffers are only the size of the actual number of pixels that compose the subarrays. This means that unswizzled buffers take less space on external media, and can be transferred out of the system faster. The data will be sent in a swizzled or raw data stream. The swizzled form is suitable for human viewing as a 1024 by 1024 array. Clearly for high-speed data acquisition, the raw form is required for maximum throughput.



4.2.10. <destination> report <string>

<destination> report <string>

The string is sent back to the client (in the IC) for completion or debugging purposes.

Example:

```
quicklook report "debug message 1"
```

```
DHS report "acquisition complete"
```

5. Status Monitor

Most sources of errors in the NICI system are reported through the controlling Command Objects. These are normally the proper avenues for communication: Temperature irregularities should show up from the temperature Command Object, for example. The buffering mechanism of the Pixel Servers is rather anonymous as these buffers are allocated on an as needed basis by any of the pixel processing modules. A memory problem here does not indicate a problem with an individual component, but rather a systemic problem: Either a memory leak in the controlling scripts, or a real-time throughput problem. In either case, an error must be posted at the instrument level.

The Status monitor SM will appear as a separate command object for the entire imaging subsystem: PS1 and PS2. Any error that is not reported through an existing Command Object will come through the status monitor. Errors that come through the SM will be considered instrument level errors and will result in NICI entering the ERROR State.

The Status Monitor will respond only to a RECOVER command after an error condition. Other commands will be ignored. Error conditions that are signalled are:

SM "no more buffers" error

The memory pool is exhausted.

SM "stale buffer in pipe" error

After an image is sampled, the image is passed from one pipeline position to the next. After all the output modules have finished with the images, the pixel servers should have no unclaimed images. This error would indicate that a "sendto" was executed but the corresponding "new" was not performed.

6. Appendix A: Sampling Modes And Low Level Actions

The following commonly used exposure modes are shown below and the array specification strings that will cause NICI to perform the proper exposure.

6.1. *Background Resets.*

Sequence:

- Clock Global Reset
- Delay between Resets
- or
- Delay before start Readout sequence

```
CLOCK RESET t1 t2
```

t1 and t2 are time of reset duration and reset pacing respectively, time durations are in nanoseconds.

6.2. *Single Sampling readout.*

Sequence:

- Clock Global Reset
- Delay for ITIME
- Clock Signal Frame Readout [place in Signal buffer]

Commands:

```
CLOCK RESET
IMAGE NEW # tell image system to look for new scan
CLOCK SCAN ITIME
IMAGE SET 0
IMAGE DECODE
IMAGE SENDTO <destination>
```

Reset, delay by ITIME milliseconds, capture to buffer 0, map to 1024 x 1024 viewing and move to destination.

6.3. *Fowler Sampling readout (NDR=1 = CDS).*

Sequence:

- Clock Global Reset
- Delay before Pedestal Readout
- Clock Pedestal Frame Readout [place in Pedestal buffer]
- Delay for ITIME
- Clock Signal Frame Readout [place in Signal buffer, subtract Pedestal from Signal]

Commands:

```
CLOCK RESET
IMAGE NEW
CLOCK SCAN
IMAGE PEDESTAL
IMAGE NEW
CLOCK SCAN ITIME
IMAGE ADD 0
```

Ver 1.4

```
IMAGE DECODE 0
IMAGE SENDTO <destination>
```

Reset, delay by <time1>, capture pedestal, delay by ITIME, capture image and move to destination.

6.4. Fowler Sampling Multiple Nondestructive Readout (#NDR=n).

Sequence:

```
  Clock Global Reset
  Delay before Pedestal Frame Readout
  Clock Pedestal Frame Readout(1) [place in Pedestal buffer]
  Clock Pedestal Frame Readout(2) [place in Pedestal2 buffer, add to Pedestal buffer]
  ...
  Clock Pedestal Readout(n) [add to Pedestal buffer, divide by n]
  Delay for ITIME
  Clock Signal Frame Readout(1) [place in Signal buffer]
  Clock Signal Frame Readout(2) [place in Signal2 buffer, add to Signal buffer]
  ...
  Clock Signal Frame Readout(n) [place in Signal2? buffer, add to Signal buffer
    , divide by n,
    then subtract Pedestal Average from Signal Average]
```

Commands:

```
CLOCK RESET
CLOCK SCAN <time1>
IMAGE NEW
IMAGE PEDESTAL
for { set I 1 } { expr I < n } { incr I } {
  CLOCK SCAN <time1>
  IMAGE NEW
  IMAGE SUBTRACT
}
CLOCK SCAN ITIME
IMAGE COPY 1
for {set I 1 } { expr I < m } { incr I } {
  CLOCK SCAN ITIME
  IMAGE NEW
  IMAGE ADD 1
}
#create dummy scan to retrieve compensated image
IMAGE NEW 0

IMAGE ADD 0 AVERAGE #pedestal frames accumulate in 0
IMAGE ADD 1 AVERAGE #image frames accumulate in 1
IMAGE DECODE
IMAGE SENDTO <destination>
```

6.5. *Coaddition (#COADD=m), Fowler Sampling readout (NDR=1 = CDS).*

Sequence:

- (1)Clock Global Reset
- (1)Delay before Pedestal Readout
- (1)Clock Pedestal Readout [place in Pedestal buffer]
- (1)Delay for ITIME
- (1)Clock Signal Frame - [place in Signal buffer, subtract Pedestal from Signal, place in Coadd buffer]
- (2)Clock Global Reset
- (2)Delay before Pedestal Readout
- (2)Clock Pedestal Readout [place in Pedestal buffer]
- (2)Delay for ITIME
- (2)Clock Signal Frame - [place in Signal buffer, subtract Pedestal from Signal, add to Coadd buffer]
- ...
- (m)Clock Global Reset
- (m)Delay before Pedestal Readout [place in Pedestal buffer]
- (m)Clock Pedestal Readout
- (m)Delay for ITIME
- (m)Clock Signal Frame - [place in Signal buffer, subtract Pedestal from Signal, add to Coadd buffer]

Commands:

```
CLOCK RESET
IMAGE FLUSH
for { set I 0 } { expr $I < $n } { incr I } {
    CLOCK SCAN <time1>
    IMAGE NEW
    IMAGE SUBTRACT
    CLOCK SCAN ITIME
    IMAGE NEW
    IMAGE ADD }
IMAGE DECODE
IMAGE SENDTO <destination>
```

6.6. *Coaddition (#COADD=m), Fowler Sampling Multiple Nondestructive Readout (#NDR=n).*

Sequence:

- Clock Global Reset
- (1)Delay before Pedestal Frame Readout
- (1)Clock Pedestal Frame Readout(1) [place in Pedestal buffer]
- (1)Clock Pedestal Frame Readout(2) [place in Pedestal2 buffer, add to Pedestal buffer]
- ...
- (1)Clock Pedestal Readout(n) [place in Pedestal2 buffer, add to Pedestal buffer, divide by n]
- (1)Delay for ITIME
- (1)Clock Signal Frame Readout(1) [place in Signal buffer]
- (1)Clock Signal Frame Readout(2) [place in Signal2 buffer, add to Signal buffer]
- ...
- (1)Clock Signal Frame Readout(n) [place in Signal2 buffer, add to Signal buffer]

Ver 1.4

```
, divide by n,  
then subtract Pedestal Average from Signal Average, place in Coadd buffer]  
(2)Delay before Pedestal Frame Readout  
(2)Clock Pedestal Frame Readout(1) [place in Pedestal buffer]  
(2)Clock Pedestal Frame Readout(2) [place in Pedestal2 buffer, add to Pedestal buffer]  
...  
(2)Clock Pedestal Readout(n) [place in Pedestal2 buffer, add to Pedestal buffer,  
divide by n]  
(2)Delay for ITIME  
(2)Clock Signal Frame Readout(1) [place in Signal buffer]  
(2)Clock Signal Frame Readout(2) [place in Signal2 buffer, add to Signal buffer]  
...  
(2)Clock Signal Frame Readout(n) [place in Signal2? buffer, add to Signal buffer  
, divide by n,  
then subtract Pedestal Average from Signal Average, add to Coadd buffer]  
...  
(m)Delay before Pedestal Frame Readout  
(m)Clock Pedestal Frame Readout(1) [place in Pedestal buffer]  
(m)Clock Pedestal Frame Readout(2) [place in Pedestal2 buffer, add to Pedestal buffer]  
...  
(m)Clock Pedestal Readout(n) [place in Pedestal2 buffer, add to Pedestal buffer,  
divide by n]  
(m)Delay for ITIME  
(m)Clock Signal Frame Readout(1) [place in Signal buffer]  
(m)Clock Signal Frame Readout(2) [place in Signal2 buffer, add to Signal buffer]  
...  
(m)Clock Signal Frame Readout(n) [place in Signal2? buffer, add to Signal buffer  
, divide by n,  
then subtract Pedestal Average from Signal Average,  
add to Coadd buffer]
```

Commands:

```
CLOCK RESET  
For {set J 1 } {expr J < m} {incr J} {  
CLOCK SCAN <time1>  
IMAGE NEW  
IMAGE PEDESTAL  
for { set I 1 } { expr I < n } { incr I } {  
CLOCK SCAN <time1>  
IMAGE NEW  
IMAGE SUBTRACT  
}  
CLOCK SCAN ITIME  
IMAGE COPY 1  
for {set I 1 } { expr I < n } { incr I } {  
CLOCK SCAN ITIME  
IMAGE NEW  
IMAGE ADD 1  
}  
#create dummy scan to retrieve compensated image  
IMAGE NEW 0  
  
IMAGE ADD 0 AVERAGE #pedestal frames accumulate in 0
```

Ver 1.4

```
IMAGE ADD 1 AVERAGE #image frames accumulate in 1
IMAGE ADDTO 2
}
IMAGE DECODE
IMAGE SENDTO <destination>
```

6.7. Multiple Coadd buffers (#COADD_BUF=a,b,c,d,e,f,g,h

6.8. Streaming mode

7. Appendix B: Terms

7.1. *Acquire*

The software Object corresponding to high level (instrument sequencer) control of the NICI image acquisition process. This command object manipulates the lower level objects automatically as described in this report.

7.2. *Classes and objects*

These are SW concepts for the abstraction of real-world things, real objects or activities. From a strict SW vocabulary a class is a bunch of lines of source code that describes data and the actions to be performed and gives rise to objects only when executed by a CPU. The objects are directly related to these same lines of source code, but are actually acting and interacting as scripted by those source lines.

7.3. *Clocker*

The hardware device to generate the control signals to the Aladdin III imaging chip. It has a single Ethernet interface to the MKIR controller for bi-directional communication.

7.4. *Command Object*

A command object is a software entity in the IC module of NICI. There is one command object for each of the NICI subsystems that the user can interact with. Each command object provides a consistent style of operation and error reporting, making it easy for NICI to be commanded by human or robotic control. Only command objects are “visible” to the user of NICI in standard operating modes.

7.5. *Image*

The software Object corresponding to the data scanned from the Aladdin III array. The data object is manipulated by commands (messages) sent to the image object.

7.6. *Objects and classes*

These are SW concepts for the abstraction of real-world things, real objects or activities. From a strict SW vocabulary a class is a bunch of lines of source code, and gives rise to objects only when executed by a CPU. The objects are directly related to these same lines of source code, but are actually acting and interacting as scripted by those source lines.

7.7. *SL240 Controller*

The hardware interface that receives data from the Aladdin III array. This hardware is programmed to automatically create Image objects.

7.8. *Subarray (also Region of Interest, ROI)*

Rectangular subregions of the Aladdin II image that contain data deemed important by the observer. Other areas of the image are not to be scanned. This can result in shorter integration time or faster exposures since the whole Aladdin III array does not need to be scanned.