National Optical Astronomy Observatories
Kitt Peak National Observatory Mayall 4m Telescope
NOAO Extra-Wide Field Infrared Mosaic Camera
http://www.noao.edu/ets/newfirm

NEWFIRM Control Software Preliminary Design

P. N. Daly
National Optical Astronomy Observatories, 950 N. Cherry Avenue, P. O. Box 26732, Tucson AZ 85726–6732, USA
pnd@noao.edu

Abstract. This document presents a preliminary design for NEWFIRM Control Software as implemented by the *Generic Manager for Instruments*.

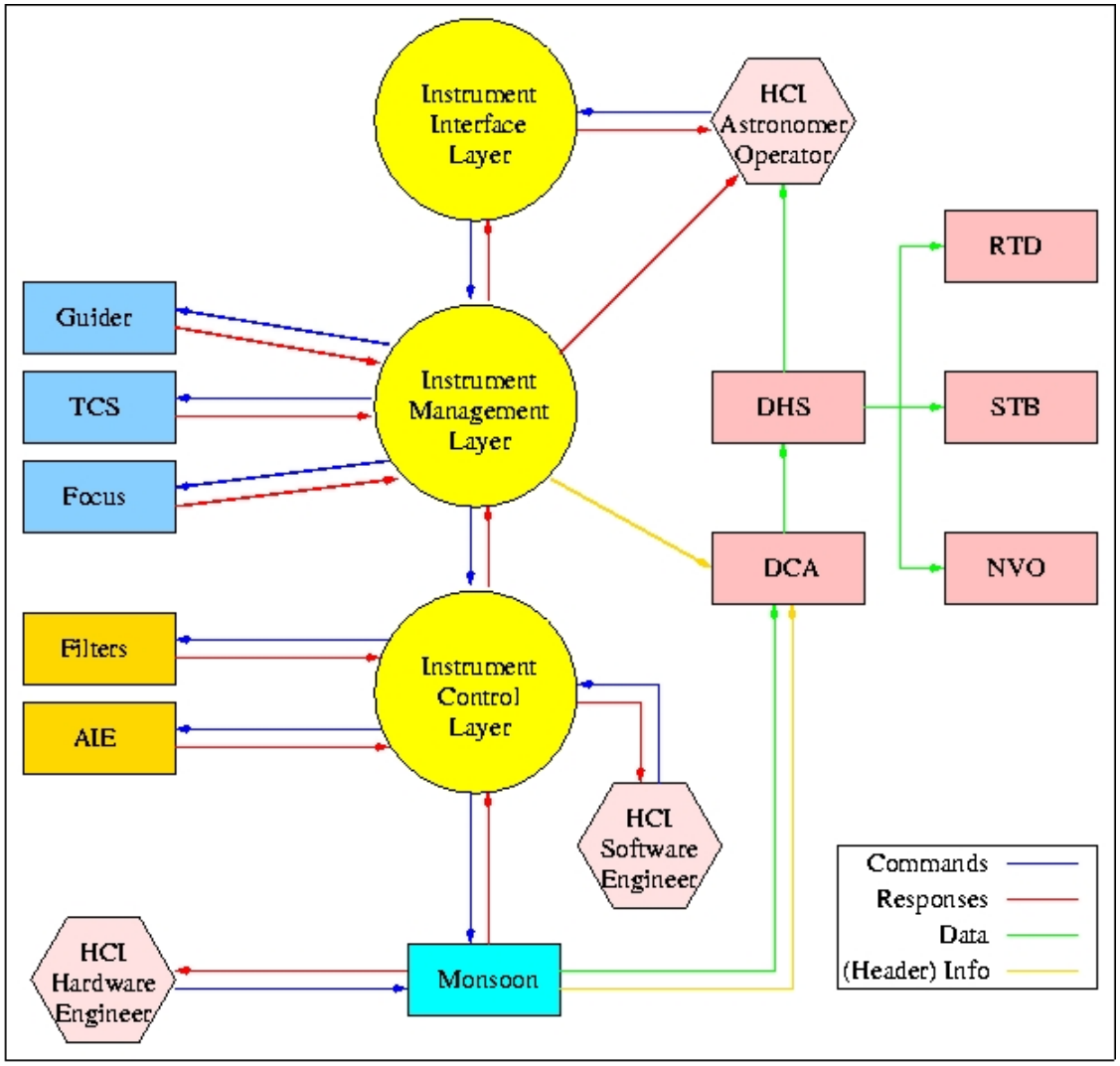
Caveat lector: This is a working document subject to change without notice. Your input is desired to make this document part of the NEWFIRM preliminary design review.

Contents

List of Figures

List of Tables

Figure 1.: NEWFIRM Conceptual Software Design



1. Introduction to NEWFIRM

The NOAO extremely wide-field infrared mosaic camera (NEWFIRM) is a 1–2.5μm camera employing four $2k \times 2k$ infrared detectors. The total, unvignetted field of view will be $\sim 25' \times 25'$ at $\sim 0.4''$ pixel^{-1} at Cassegrain focus. NEWFIRM will have a filter wheel and some associated *Auxiliary Control Electronics* for temperature sensing and/or control.

This instrument is radically different from previous instruments delivered to the Mayall 4m telescope inasmuch as NEWFIRM will, largely, be a survey instrument operated via an automated queue mechanism of some sort with little or no astronomer intervention [1, 2].

Figure 1 on page 4 shows a cartoon of the system as it stood at conceptual design. In brief, an astronomer may prepare observation sequences in advance using some sort of (as yet undefined) observing tool. Standard observing templates will be used and modified to achieve the desired

science programme. This, in effect, defines the *Instrument Interface Layer* discussed in a little more detail below.

These observation sequences will be merged with other programmes to be carried out on the same UTC date and submitted to a queue. It is the responsibility of the *Instrument Management Layer* to coordinate all systems (internal and external) to achieve the required observations. As such this is sometimes referred to as the *Observation Management Layer*. We will return to this layer, in some detail, later in this document.

The *Instrument Control Layer* handles specific hardware associated with the instrument like filter wheels plus all communications with the *Generic Pixel Server* [3] shown here as a *Monsoon* implementation [4].

2. Desireable Attributes of the Software Design

In a broad sense, we seek to design and implement the NEWFIRM software in such a way that lessons learned from the past can be re-implemented and the code base serve as a model for the future. So, we desire to make the architecture:

MODULAR: by writing the code in a modular way, we can partition the software into distinct, well defined, units. If the interfaces between such units are well understood and documented, such modules can be replaced and/or upgraded as future needs arise without compromising the whole system.

EXTENDIBLE: by writing the code using simple, building blocks we can lay down a framework whereby future enhancements to the system may be easily achieved. For example, there is no specific requirement for polarimetry with NEWFIRM but the addition of such an internal or external optical element should be accommodated within the software with the minimum of changes.

PORTABLE: if the code is written without regard to a particular operating system or device, we can leverage the price/performance ratios for future instruments to our advantage.

MAINTAINABLE: for the above to work, as well as to maintain the system post-delivery, the software should be well documented using common techniques.

RE-USEABLE: by breaking the software down into units, and possibly utilizing advanced programming techniques such as object orientation, code re-use becomes a natural consequence of the software architecture.

LIMITED SCOPE: the scope of the software should be clearly defined with interfaces between internal and external systems identified and published.

Naturally, the question arise as to whether or not some system abstraction aids in this process and, if so, to what end should we strive to produce a generic system? Following this train of thought, various observation states have been identified as shown in table 1 on page 6. In this table, the major subsystems of concern are the detector (configuration), the instrument (configuration) and the telescope (configuration). Note that, in this table, 'Fixed' means just that and 'Variable' means that that subsystem *must* vary.

At present, table 1 is at odds with a similar concept in the NEWFIRM System Architecture Document [5]. This will have to be reconciled. The reason for suggesting a firmer definition of these 'sequencing levels', and changing the concept to 'states', is that:

Table 1.: Observation States

Observation State	Detector Configuration	Instrument Configuration	Telescope Configuration
0	Variable	Variable	Variable
1	Variable	Variable	Fixed
2	Variable	Fixed	Fixed
3	Fixed	Variable	Fixed
4	Fixed	Fixed	Variable
5	Fixed	Fixed	Fixed

1. When specific states are identified, as opposed to levels, there may be some mileage in considering a state-machine approach.
2. Although NEWFIRM has no requirement for a complex scheduler to dynamically select the next observation (being the closest match to the present observing configuration, for example), a state definition table does present the possibility for future enhancement.
3. State 0 can be considered a virtual, degenerate state as it is the state in which states 2, 3 and 4 co-exist.

How much mileage we can extract from these models remains to be seen.

3. NEWFIRM Control Software Preliminary Design

Figure 2 on page 7 shows the Yourdon-deMarco compliant data flow diagram for the preliminary design of the *Generic Manager for Instruments* as part of the NEWFIRM software effort. Unless otherwise noted all communications are socket based including inter-process communications. Although this is less efficient than using shared memory when the cooperative processes run on the same machine, it is platform and operating system independent and thus satisfies the portability goal.

The various items seen in figure 2 are described below.

3.1. External Entities

The following external entities are defined:

AIE: the *Auxiliary Instrument Electronics*. This has yet to be fully defined.

DHS: the portal to the *Data Handling System*. Note that the GMIodb process pushes data towards the *DHS* during and after observations. Some aspects of this interface have yet to be defined.

GPX: the *Generic Pixel Server* is, in this context, the *Monsoon Image Acquisition System* as described elsewhere [3, 4].

Figure 2.: NEWFIRM Preliminary Design

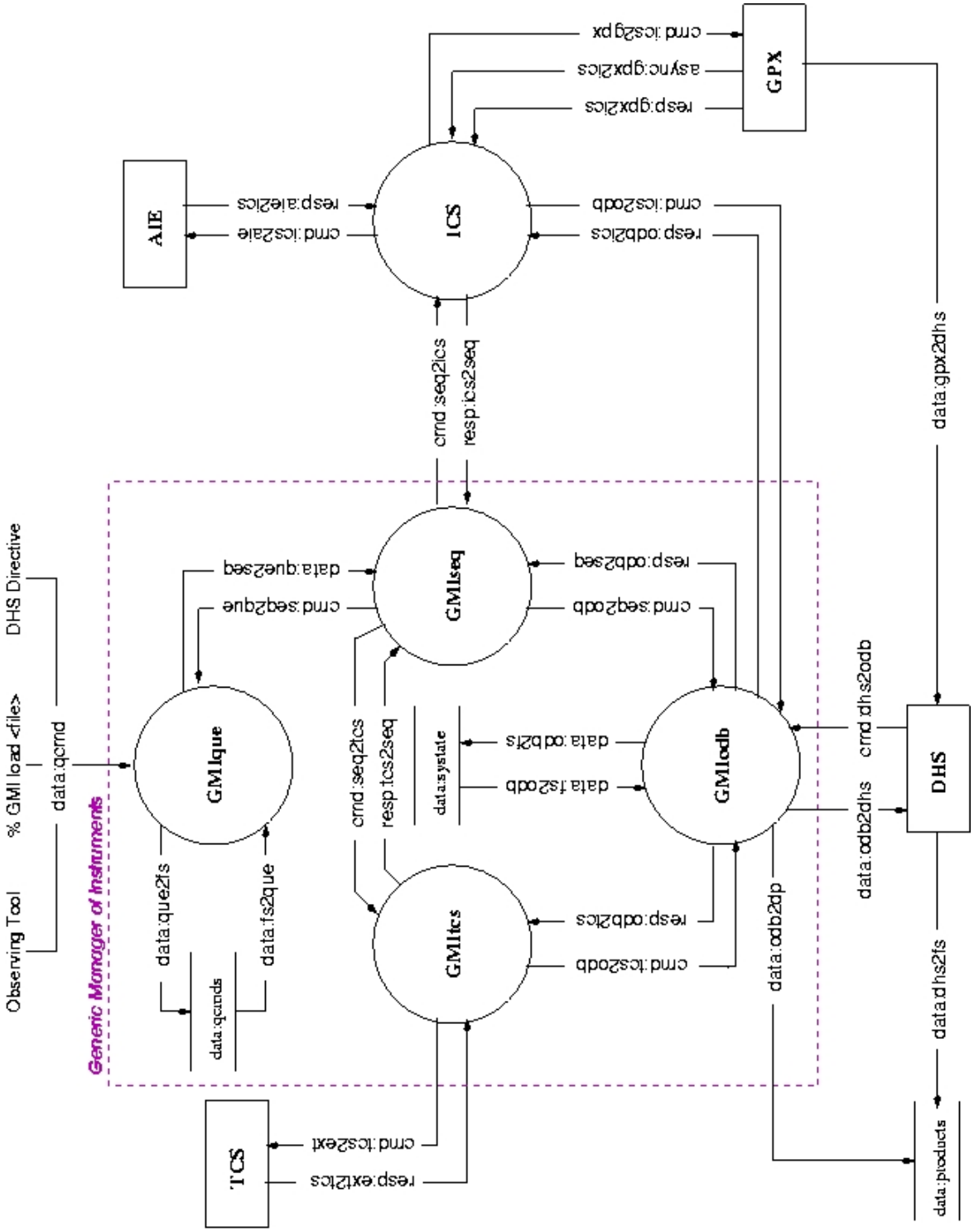
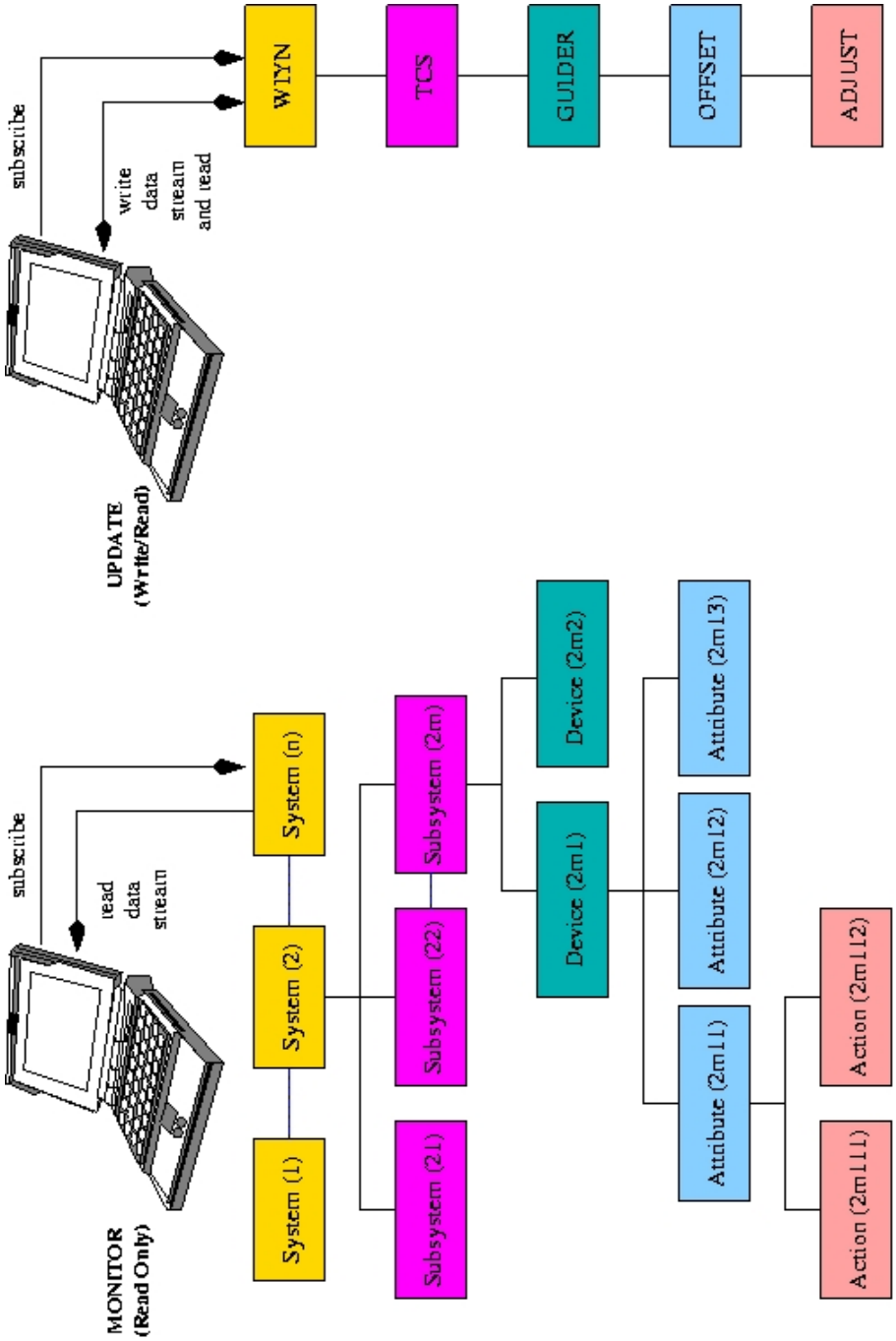


Figure 3.: GWC Hierarchy and Example



TCS: the *Telescope Control System*. Communications with this system runs via the *Generic WIYN Client* software (also known as the *MPG Router Interface*). This interface is described elsewhere [6].

3.2. Processes

The following processes[1] are defined:

ICS: This process controls all aspects of the NEWFIRM instrument and is responsible for configuring the *Generic Pixel Server* and handling the *Auxiliary Instrument Electronics*.

GMIQUE: This process controls and maintains access to the queue of known observations. Upon request it will furnish the next command.

GMITCS: This process handles all communications with the (external) telescope control system. As such, it converts between any internal protocol and the *Generic WIYN Client* software as show in the cartoon in figure 3 on page 8.

GMIODB: This process handles all data required by the *DHS* to correctly identify and manipulate the data. It flushes the data at the end of every observation as well as saving the information to a file.

GMISEQ: This process sequences the desired observation by co-coordinating the telescope, instrument upon receipt of a suitable command from the observation queue.

Each of these systems is defined in greater detail in later sections.

3.3. File Stores

The following file stores are defined:

DATA:QCMDS The store of observation commands.

DATA:SYSTATE The store of system state and header information.

DATA:PRODUCTS The store of derived data products.

3.4. Data Flows, Sources and Sinks

The following data flows are defined:

DATA:QCMD Incoming commands to the observation queue.
Source: Observing tool, *DHS* directive, GMI load program; *Sink:* GMIque.

DATA:QUE2FS Incoming commands to file store.
Source: GMIque process; *Sink:* The file store data:qcmds.

DATA:FS2QUE Returned commands from file store.
Source: The file store data:qcmds. *Sink:* GMIque process.

[1] This term is used loosely to mean a process, module or task as the implementation may allow.

CMD:SEQ2QUE Re-files a failed observation or lists queue.
Source: GMIseq process; *Sink:* GMIque process.

DATA:QUE2SEQ Returned observation command from queue.
Source: GMIque process; *Sink:* GMIseq process.

CMD:SEQ2TCS Configuration directive for telescope.
Source: GMIseq process; *Sink:* GMItcs process.

RESP:TCS2SEQ Status, error from configuration directive.
Source: GMItcs process; *Sink:* GMIseq process.

CMD:SEQ2ODB Initialization directive for new observation.
Source: GMIseq process; *Sink:* GMIodb process.

RESP:ODB2SEQ Status, error from initialization directive.
Source: GMIodb process; *Sink:* GMIseq process.

CMD:TCS2ODB Telescope status, error for current observation.
Source: GMItcs process; *Sink:* GMIodb process.

RESP:ODB2TCS Telescope status, error acknowledgement.
Source: GMIodb process; *Sink:* GMItcs process.

DATA:ODB2FS Observation and system status to file store.
Source: GMIodb process. *Sink:* The file store data:systate.

DATA:FS2ODB Observation and system status from file store.
Source: The file store data:systate. *Sink:* GMIodb process.

CMD:TCS2EXT *TCS* directives via *Generic WIYN Client*.
Source: GMItcs process; *Sink: TCS* (router).

CMD:EXT2TCS *TCS* status via *Generic WIYN Client*.
Source: TCS (router). *Sink:* GMItcs process.

RESP:EXT2TCS Telescope status, error acknowledgement.
Source: GMIodb process; *Sink:* GMItcs process.

DATA:ODB2DP Observation and system status to (external) file store.
Source: GMIodb process. *Sink:* The file store data:products.

DATA:ODB2DHS Observation and system status to *DHS*.
Source: GMIodb process. *Sink: DHS*.

CMD:DHS2ODB Request for re-transmission of observation tag data.
Source: DHS. *Sink:* GMIodb..

DATA:DHS2FS *DHS* reduction data and/or products.
Source: DHS. *Sink:* The file store data:products.

DATA:GPX2DHS *GPX* data destined for *DHS*
Source: GPX. Sink: DHS.

CMD:SEQ2ICS Configuration and command directives to instrument.
Source: GMIseq process *Sink:* NEWFIRM *ICS* process.

RESP:ICS2SEQ Status, error from instrument.
Source: NEWFIRM *ICS* process. *Sink:* GMIseq process

CMD:ICS2ODB Instrument status, error for current observation.
Source: NEWFIRM *ICS* process; *Sink:* GMIodb process.

RESP:ODB2ICS Instrument status, error acknowledgement.
Source: GMIodb process; *Sink:* NEWFIRM *ICS* process.

CMD:ICS2AIE auxiliary instrument electronics directives.
Source: NEWFIRM *ICS* process; *Sink: AIE*.

RESP:AIE2ICS Instrument status, error acknowledgement.
Source: AIE. Sink: NEWFIRM *ICS* process;

CMD:ICS2GPX Instrument *GPX* directives.
Source: NEWFIRM *ICS* process; *Sink: GPX*.

RESP:GPX2ICS Instrument *GPX* responses.
Source: GPX. Sink: NEWFIRM *ICS* process;

ASYNC:GPX2ICS Asynchronous *GPX* responses.
Source: GPX. Sink: NEWFIRM *ICS* process;

4. Observation Tags

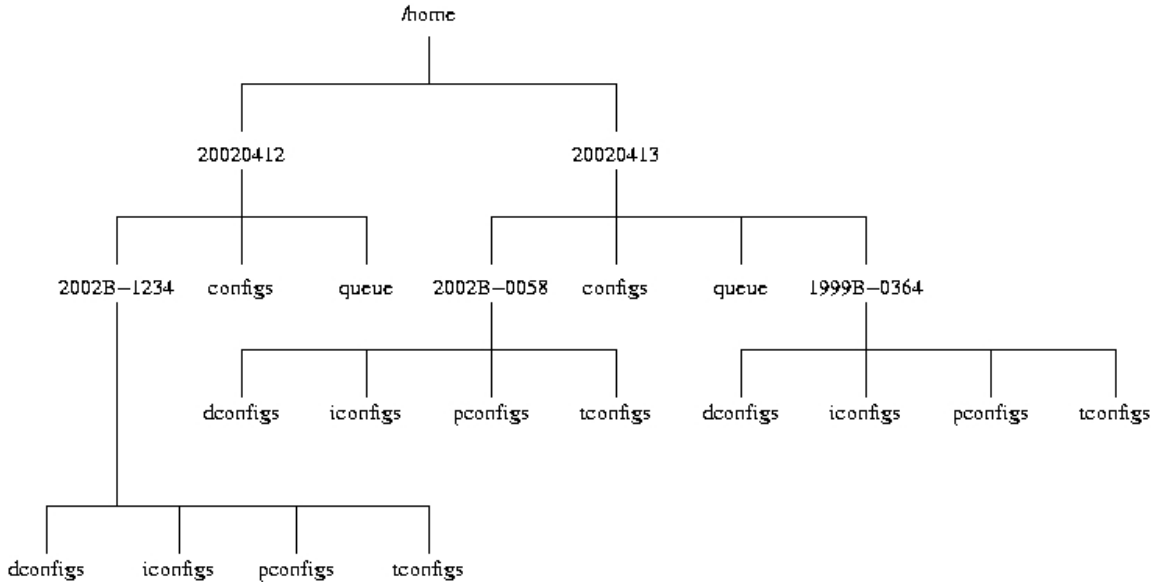
Since the basic unit of data *above* the level of the *GPX* is the *observation*, it is useful to identify each such unit by a unique observation tag. The convention adopted here is a string of the fixed form *yyyymmdd_ooooo* where *yyyymmdd* is the UTC date and *ooooo* is a monotonically increasing observation number. Thus, the first observation at the time of writing would be tagged, 20020412_00001 and the second 20020412_00002 and so on. The reason for adopting fixed format numbers is to make directory listings and various Unix utilities receive observation sequences in alphanumeric order.

It also aids in locating critical system files as shown in figure 4 on page 12. In this figure we see a top level /home[2] with various *UTC* sub-directories. Within each *UTC* directory are one or more *proposal-ID* sub-directories, one *config* sub-directory and one *queue* sub-directory. This scheme allows for several proposals to be handle within a single observing night.

Within each *proposal-ID* directory are the configuration file directories described below. In the case of 20020412 in figure 4 these configuration files will simply be copied from the *proposal-ID* directories to the *configs* and *queue* directories where the GMI will locate them. For 20020413, a

[2]This can be changed by re-assigning the environmental variable $GMI_HOME.

Figure 4.: GMI Configuration and Queue Directories



tool will be provided to merge the two proposals consecutively in a pre-defined order so that, once again, the *configs* and *queue* directories are only populated by files needed during the observing night.

4.1. The DCONFIG Files

This directory contains the detector configuration files for each observation tag. These files are of the form *yyyymmdd_ooooo*.dcfg and are plain ASCII[3]. As of yet, a template *dconfig* file has yet to be defined.

4.2. The ICONFIG Files

This directory contains the instrument configuration files for each observation tag. These files are of the form *yyyymmdd_ooooo*.icfg and are plain ASCII. As of yet, a template *iconfig* file has yet to be defined.

4.3. The PCONFIG Files

This directory contains the programme configuration files for each observation tag. These files are of the form *yyyymmdd_ooooo*.pcfg and they contain high-level information regarding the science programme that will not change unless the proposal being scheduled has been changed. Such files are straightforward and can be populated from the following template:

[3] Although the use of XML for all configuration files has not been ruled out.

```
SIMPLE  =                    T / File conforms to FITS standard
BITPIX  =                   32 / Number of bits per pixel
NAXIS   =                    2 / Number of data axes
NAXIS1  =                 4096 / Length of data axis 1
NAXIS2  =                 4096 / Length of data axis 2
COMMENT FITS (Flexible Image Transport System) format defined in Astronomy and
COMMENT Astrophysics Supplement Series v44/p363, v44/p371, v73,p359, v73,p365.
COMMENT Contact the NASA Science Office of Standards and Technology for the
COMMENT FITS definition document #100and other FITS information.
BSCALE  =                   1.0 / True_value = BSCALE * FITS_value + BZERO
BZERO   =                   0.0 / True_value = BSCALE * FITS_value + BZERO
BLANK   =           -2147483648 / Array value whose physical value is undefined
BUNIT   = 'ADU per integration' / Units of primary array
TELESCOP= 'Mayall 4m, KPNO, AZ' / Telescope name
INSTRUME= 'NEWFIRM            ' / Instrument
OBSERVER= 'Philip N. Daly     ' / Observer name(s)
PROPOSER= 'Philip N. Daly     ' / Proposer name(s)
PROPOSAL= 'NEWFIRM T&E        ' / Proposal name
PROPID  = '2002B-1234         ' / Proposal ID
USERID  = 'pnd                ' / User identification code
OBSREF  = 'NEWFIRM T&E        ' / Observer reference
ORIGIN  = 'NOAO, Tucson, AZ   ' / Organization creating the FITS file
DATE    = '2002-04-12         ' / Date (UTC) when FITS file was created
```

4.4. The TCONFIG Files

This directory contains the telescope configuration files for each observation tag. These files are of the form *yyyymmdd_ooooo*.tcfg. These files are straightforward and can be populated from the following template:

```
RA      = '08:08:29.84        ' / RA of observation (hr)
DEC     = '28:51:48.67        ' / DEC of observation (deg)
EQUINOX =               2000.0 / Equinox of object position FK5 (years)
TFOCUS  =               5555.0 / Telescope focus position (microns)
TRAOFF  =                 80.0 / Telescope offset RA at equinox (arcsec)
TDECOFF =                 80.0 / Telescope offset DEC at equinox (arcsec)
GRAOFF  =                  0.0 / Guider offset RA at equinox (arcsec)
GDECOFF =                  0.0 / Guider offset DEC at equinox (arcsec)
```

After an initial target has been acquired, only those items that change need be specified in the *tconfig* file.

5. The Generic Manager for Instruments: GMI

In figure 2 on page 7 we have delimited an area of the diagram and labelled it *Generic Manager for Instruments*. This is a further exploration of the bubble labelled *Instrument Management Layer* in figure 1 on page 4 also called the *Observation Management Layer* [5]. By partitioning the problem in this way, we can isolate those functions associated with the specific instrument under consideration with the more general task of scheduling *any* instrument on the telescope.

Note that all communications are socket streams using 'well known' port numbers that have yet to be allocated. In the following narrative the term 'observation commands' refers to arbitrary-length strings maintained in the DATA:QCMDS file store. The term 'action' means a command that the individual GMI process will respond to.

5.1. Message Streams

All message streams will conform to the *GPX* proposed standard [4] which is a network byte-ordered stream with the following protocol:

HEADER 12 bytes as shown in table 2 on page 14.

Table 2.: Message Protocol Header

BYTE 0	BYTE 1	BYTE 2	BYTE 3
type	protocol	exposure id	
BYTE 4	BYTE 5	BYTE 6	BYTE 7
message body length (unsigned int)			
BYTE 8	BYTE 9	BYTE 10	BYTE 11
unknown	unknown	unknown	IP address

BODY 1 to 4294967295 bytes.

CHECKSUM 4 byte CRC.

Note that the message types are: control (0x01), keyword (0x02), event (0x03), pixel data (0x04), FITS header data (0x05) and fixed block data (0x06).

5.2. Error Codes

Unless otherwise specified, the following response codes are defined:

PROGRESS magic number, progress of observation.

DONE magic number, end of observation.

ACK magic number, temporary acknowledgement.

OK 0, normal return.

ERROR -1, abnormal return.

WARNING -2, abnormal return.

FATAL -3, abnormal return.

5.3. GMIque Specification and Design

GMIque Overview The main function of the queue manager is to accept a list of desired observation commands and store them in an appropriate order until asked for by the sequencer. To do this, it will maintain a time-stamped queue along a similar design to the Qman software [7].

GMIque Functionality The functionality required is:

1. Verify that the requesting client is authorized.
2. Accept observation commands comprising variable length strings.
3. Time stamp observation commands in such a way that it is placed at the head or tail of the queue as requested.
4. Maintain the queue and its integrity.
5. Periodically flush the queue to disk in case of unexpected errors.
6. Provide a save and restore facility.
7. Return the next observation command from the queue as requested by an authorized client.
8. Provide a list of queue entries in some requested order.

GMIque Actions and Responses The GMIque process will respond to the following actions:

CONNECT connect to and authenticate client.
Command: gmique.connect(host,mode)
Parameter(s): host = client name, mode = read or write.
Default(s): host = 127.0.0.1, mode = read.
Response: OK or ERROR.
Comment(s): Only 1 read client is allowed, multiple write clients.

INIT initialize the queue contents.
Command: gmique.init()
Parameter(s): none.
Default(s): none.
Response: OK or ERROR.
Comment(s): Invoked, automatically, at system start up.

LIST list the queue contents.
Command: gmique.list(order)
Parameter(s): order = increasing or decreasing.
Default(s): order = increasing.
Response: OK or ERROR.
Comment(s): Order refers to time stamp keys.

WRITE write a new observation command to the queue at requested position.
Command: gmique.write(command,position)
Parameter(s): command = observation command, position = head or tail.
Default(s): command = none, position = tail.
Response: OK or ERROR.
Comment(s): ‘head’ refers to the latest filed observation command.

READ read an observation command from the queue.
Command: gmique.read(position)
Parameter(s): position = head or tail.
Default(s): position = head.
Response: OK or ERROR.
Comment(s): 'head' refers to the latest filed observation command.

GMIque Implementation Notes To achieve these goals, the queue will be maintained as an associative array of {key:value} pairs with the keys being time stamps and the values being the observation command strings. The time stamp will be derived from the well known function returning the number of seconds since the start of Unix time plus some slight modification to handle sub-second time stamps and unique keys [7]. A read of the queue is a *destructive* process so any observation command that fails may be re-filed by specifying 'head' as the queue position. Rather than re-order the queue (which might be quite long), the simplest scheme is to file these 'head' observation commands by negating the time stamp. In this scheme the 'head' of the queue is the key with the *minimum* value. The 'tail' is the key with the maximum value.

The observation commands that may be written to the queue are described elsewhere (see section 5.4.) since the GMIque process does not require knowledge of the substance of the incoming observation command (it just stores the string in the appropriate place in the queue).

5.4. GMIseq Specification and Design

GMIseq Overview The main job of the GMIseq process is to coordinate data taking activity amongst all tasks in the GMI system. As such is can read an observation command queue and instruct the instrument or telescope to set to some configuration. When all subsystems are ready, the GMIseq process tells the instrument to take data and listens for asynchronous messages regarding progress and completion. Upon completion, the GMIseq process instructs the GMIodb process to flush header and status data to the message bus and disk.

The command dictionary is small and extendible:

PCONFIG <OBSERVATION TAG>: sets up programme configuration data via GMIodb.

ICONFIG <OBSERVATION TAG>: sets up instrument configuration data via *ICS*.

DCONFIG <OBSERVATION TAG>: sets up detector configuration data via *ICS*.

TCONFIG <OBSERVATION TAG>: sets up telescope configuration data via GMItcs.

OBSERVE <OBSERVATION TAG>: starts the observation for the given tag.

GMIseq Functionality The functionality required is:

1. Verify that the requesting client (user interface) is authorized.
2. Connect to and verify all other GMI processes.
3. Connect to and verify communication with the *ICS*.
4. Read an observation command via the GMIque process.
5. Act upon known commands and wait for completion as required.

6. Start an observation for a given tag.
7. Report asynchronous status and completion messages.
8. Issue a DONE <OBSERVATION TAG> directive to appropriate GMI processes.

GMIseq Actions and Responses

CONNECT connect to and authenticate client.
Command: gmiseq.connect(host)
Parameter(s): host = client name.
Default(s): host = 127.0.0.1
Response: OK or ERROR.
Comment(s): Only client(s) are the user interfaces.

INIT initialize the process.
Command: gmiseq.init()
Parameter(s): none.
Default(s): none.
Response: OK or ERROR.
Comment(s): Invoked, automatically, at system start up.

READQ requests the next observation command from the queue.
Command: gmiseq.read(position)
Parameter(s): position = the head or tail of the queue position.
Default(s): position = tail.
Response: OK or ERROR.
Comment(s): Obtains the next observation command.

SEND sends a command to another process.
Command: gmiseq.send(command,process)
Parameter(s): command = directive, process = recipient.
Default(s): none.
Response: OK or ERROR.
Comment(s): main method of inter-process communication.

REPORT sends status and error messages to requesting client.
Command: gmiseq.report()
*Parameter(s):*none.
Default(s): none.
Response: OK or ERROR.
Comment(s): Only client(s) are the user interfaces.

GMIseq Implementation Notes The GMIseq maintains a set of global variables regarding the state of the subsystems. If we adhere to table 1 on page 6 then the GMIseq process will consider the telescope, instrument and detector subsystems to be 'ready' (fixed) or 'not ready' (varying). An observation can be started only when all 3 subsystems are 'ready'. This process will spend a lot of time in a loop reading the queue, setting up the observation and waiting for completion. At present, the maximum cadence is expected to be a complete observation every 3 seconds although this limit is hardware bound and should not be reflected in the software. Any software limit will come from

using sockets for inter-process communications and this will have to be benchmarked to assess the latency.

5.5. GMItcs Specification and Design

GMItcs Overview The main function of the GMItcs process is to maintain and interpret communication with the telescope. As such it acts as a protocol converter between internal message streams and the *Generic WIYN Client* telescope control system.

GMItcs Functionality The required functionality is:

1. Verify that the requesting client is authorized.
2. Accept observation tags and locate the associated TCONFIG file.
3. Load a TCONFIG and set the telescope to that configuration.
4. Communicate error and status information to the GMIodb process at 1 Hz.

GMItcs Actions and Responses

CONNECT connect to and authenticate client.
Command: gmitcs.connect(host)
Parameter(s): host = client name.
Default(s): host = 127.0.0.1
Response: OK or ERROR.
Comment(s): Only client is GMIseq.

INIT initialize the process.
Command: gmitcs.init()
Parameter(s): none.
Default(s): none.
Response: OK or ERROR.
Comment(s): Invoked, automatically, at system start up.

LOAD load a TCONFIG for a given observation tag.
Command: gmitcs.load(tag)
Parameter(s): tag = observation tag.
Default(s): none.
Response: OK or ERROR.
Comment(s): Locates and loads TCONFIG file.

UPDATE updates current information.
Command: gmitcs.update(tag,data)
Parameter(s): tag = observation tag, data = associated data.
Default(s): none.
Response: OK or ERROR.
Comment(s): Updates all items declared in section 4.4..

GMItcs Implementation Notes The GMItcs process will maintain a data structure for every observation tag consisting of the following items: ra, dec, equinox, tfocus, traoff, tdecoff, graoff and gdecoff (as described in section 4.4..). Upon receipt of a TCONFIG directive, it will locate the file, read the values and configure the telescope to the desired state. It will maintain that configuration and that observation tag until a new tag is received. The data will be flushed to the GMIodb process at 1 Hz (this being a constraint of the *Generic WIYN Client* router). Upon receipt of a new observation tag, the old data will be flushed and a new data structure created and so on. Should the new observation tag file consist of fewer items than the 8 known possibilities, the previous values will be used for the omitted items. In this way a dither set may be created by specifying all items in the base set but only the offsets in further tag files.

5.6. GMIodb Specification and Design

GMIodb Overview The main function of the GMIodb process is to maintain a data structure for every observation tag. This data structure will provide sufficient and necessary information to the *Data Handling System*. It will also flush the data structure to disk at the end of every observation. It will *not* request information from other processes but will allow such processes to update its database. For clarity, the list of items is documented in Appendix A and is based upon recognized defaults [8] although these will have to be reconciled with the local way of doing things.

GMIodb Functionality The functionality required is:

1. Verify that the requesting clients are authorized.
2. Create and maintain a data structure for every observation tag.
3. After completion of an observation, flush the data to the *DHS* and to an engineering log file on disk.
4. List the current observation tag structure upon request.
5. Accept re-transmission requests for observation tags.

GMIodb Actions and Responses

CONNECT connect to and authenticate clients.
Command: gmiodb.connect(host)
Parameter(s): host = client name.
Default(s): host = 127.0.0.1
Response: OK or ERROR.
Comment(s): Allowed clients are GMItcs, GMIseq and *ICS*.

INIT initialize the process.
Command: gmiodb.init()
Parameter(s): none.
Default(s): none.
Response: OK or ERROR.
Comment(s): Invoked, automatically, at system start up.

LIST list an observation tag.
Command: gmiodb.list(tag)
Parameter(s): tag = observation tag.
Default(s): none.
Response: OK or ERROR.
Comment(s): Lists the contents of the given observation tag.

UPDATE updates observation tag information.
Command: gmiodb.update(tag,data)
Parameter(s): tag = observation tag, data = associated data.
Default(s): none.
Response: OK or ERROR.
Comment(s): Updates all items declared in Appendix A.

REPEAT re-transmits observation tag information.
Command: gmiodb.repeat(tag,data)
Parameter(s): tag = observation tag, data = associated data.
Default(s): data = all.
Response: OK or ERROR.
Comment(s): Re-transmits one or all items declared in Appendix A.

GMIodb Implementation Notes The data structure(s) for this process must be dynamically extendible for future expansion. At present, the data is flushed to disk and the *DHS* upon receipt of a DONE <OBSERVATION TAG> directive from the GMIseq process. The process will also maintain a list of the last *n* observation tag structures in memory for re-transmission where the value of *n* is to be decided.

6. NEWFIRM Instrument Control System

Overview The NEWFIRM instrument control system handles all communications with the *auxiliary instrument electronics* for filter wheel control *etc* and all communications with the *generic pixel server*.

ICS Functionality This section has yet to be written.

ICS Actions and Responses This section has yet to be written.

ICS Implementation Notes This section has yet to be written.

7. NEWFIRM Typical Use Case

In many cases, NEWFIRM will be used for dithered mapping of large areas of the sky. This is shown in figure 5 on page 21 where each map position (m1,m2,m3,m4) is offset from the others by a relatively large amount (∼28') but around each such position a smaller dither (d0,d1,d2,d3,d4) is performed with offsets ∼100 pixels (∼40").

How does this scheme operate in the above architecture?

Figure 5.: NEWFIRM Dither Mapping Pattern

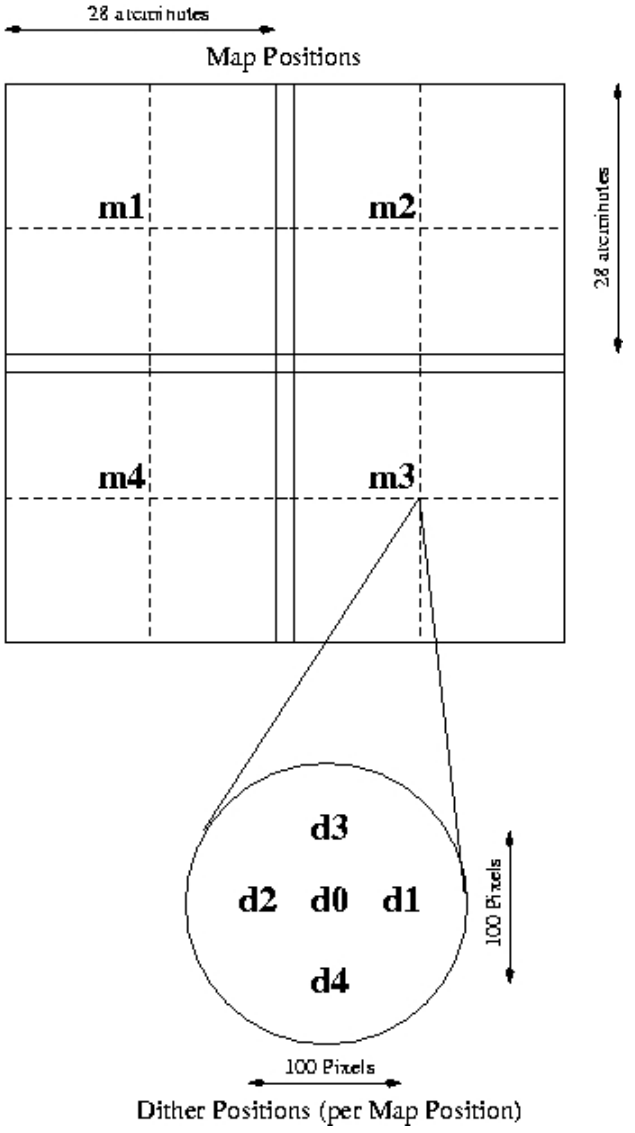


Table 3.: Observation Use Case: Mapped Dither Set

Line # (col 1)	State (col 2)	Time Stamp (col 3)	Observation Command (col 4)	Comment (col 5)
001	5	101898910.31	pconfig 20020412_00001	Set programme config
002	2	101898910.97	dconfig 20020412_00001	Set detector config
003	3	101898911.42	iconfig 20020412_00001	Set instrument config
004	4	101898911.88	tconfig 20020412_00001	Set telescope config
005	5	101898912.35	observe 20020412_00001	Observe m1, d0
006	4	101898912.76	tconfig 20020412_00002	Set telescope offset
007	5	101898913.19	observe 20020412_00002	Observe m1, d1
008	4	101898913.54	tconfig 20020412_00003	Set telescope offset
009	5	101898913.92	observe 20020412_00003	Observe m1, d2
010	4	101898914.25	tconfig 20020412_00004	Set telescope offset
011	5	101898914.55	observe 20020412_00004	Observe m1, d3
012	4	101898914.89	tconfig 20020412_00005	Set telescope offset
013	5	101898915.20	observe 20020412_00005	Observe m1, d4
004	4	101898916.88	tconfig 20020412_00001	Set telescope config
005	5	101898917.35	observe 20020412_00001	Observe m2, d0
006	4	101898917.76	tconfig 20020412_00002	Set telescope offset
007	5	101898918.19	observe 20020412_00002	Observe m2, d1
008	4	101898918.54	tconfig 20020412_00003	Set telescope offset
009	5	101898918.92	observe 20020412_00003	Observe m2, d2
010	4	101898919.25	tconfig 20020412_00004	Set telescope offset
011	5	101898919.55	observe 20020412_00004	Observe m2, d3
012	4	101898919.89	tconfig 20020412_00005	Set telescope offset
013	5	101898920.20	observe 20020412_00005	Observe m2, d4

First, let us use the configuration files specified in sections 4.3. and 4.4.. Add to these some extra TCONFIG files which consist *only* of TRAOFF and TDECOFF directives.

To implement the dithered set mapping programme, the observation queue will appear as columns 3 and 4 in table 3 on page 22. In this table we show only the commands for the initial configuration and map position m1 and m2 with dithers d0, d1, d2, d3 and d4. Note that we have not said how the queue was populated so we just accept (for now) that it is.

Tables 4, 5 and 6 on pages 24, 25 and 26 respectively show the observation sequence chart for the likely data flow, actions and responses for the first 8 commands read off the queue. This cannot be a precise analysis because the processes can run independently and concurrently. This sequence chart is as probable as any other one might envisage.

To read these tables, note that the sequence number appears in the first column and several processes can share the same sequence number when actions or data flows proceed in parallel. The data flows are consistent with those shown in figure 2 on page 7. The actions are consistent with those defined for GMI processes in section 5. on page 5. whilst the others are, as yet, undefined but should give a flavour of what is required. The response is noted where appropriate.

8. Error and Exception Handling

This section has yet to be written.

9. User Interface(s)

This section has yet to be written.

10. Document Revision History

11 April 2002, PND: Original version.

Acknowledgments. PND would like to thank N. C. Buchholz for some useful talks.

References

Autry, R. G. *et al.*, 2001, *NEWFIRM Functional Performance Requirements Document*, NEWFIRM Project, August Draft.

Autry, R. G. *et al.*, 2001, *NEWFIRM Operational Concepts Definition Document*, NEWFIRM Project, July Draft.

Buchholz, N. C., Daly, P. N. and Starr, B. M. 2002, *Generic Pixel Server: Communications, Command/Response and Data Stream Interface Description*, v0.1.2, March.

Buchholz, N. C. 2002, *Generic Pixel Server: Data Status Stream Interface Description*, v0.1, March.

Starr, B. M., Daly, P. N., Buchholz, N. C., Tody, D. and Shaw, R. A. 2002, *NEWFIRM System Architecture Document*, v0.1, April (draft).

Table 4.: Observation Sequence: Mapped Dither Set (Part I)

Seq #	Process	From	To	Action	Response
01	GMIseq		cmd:seq2que	gmique.read(head)	
02	GMIque	data:fs2que	cmd:que2seq		pconfig 20020412_00001
03	GMIseq		cmd:seq2odb	gmiodb.update(20020412_00001)	
04	GMIodb	resp:odb2seq			ack update 20020412_00001
05	GMIseq		cmd:seq2que	gmique.read(head)	
05	GMIodb		data:odb2fs	create 20020412_00001	
06	GMIque	data:fs2que	cmd:que2seq		dconfig 20020412_00001
06	GMIodb		data:odb2fs	read 20020412_00001.pcfg	
07	GMIseq		cmd:seq2ics	ics.dconfig(20020412_00001)	
07	GMIodb		data:odb2fs	gmiodb.update(20020412_00001)	
08	ICS	resp:ics2seq			ack dconfig 20020412_00001
08	GMIseq		cmd:seq2que	gmique.read(head)	
08	GMIodb	resp:odb2seq			ok update 20020412_00001
09	ICS			read 20020412_00001.dcfg	
10	GMIque	data:fs2que	cmd:que2seq		iconfig 20020412_00001
10	ICS		cmd:ics2gpx	gpxConfigure 20020412_00001	
11	GMIseq		cmd:seq2ics	ics.iconfig(20020412_00001)	
11	ICS	resp:gpx2ics			ack gpxConfigure 20020412_00001
12	ICS	resp:ics2seq			ack iconfig 20020412_00001
12	GMIseq		cmd:seq2que	gmique.read(head)	
12	ICS	resp:gpx2ics			ok gpxConfigure 20020412_00001
13	GMIque	data:fs2que	cmd:que2seq	tconfig 20020412_00001	
13	ICS	resp:ics2seq			ok dconfig 20020412_00001
14	GMIseq		cmd:seq2tcs	gmitcs.load(20020412_00001)	
15	GMItcs	resp:tcs2seq			ack tconfig 20020412_00001
15	ICS			read 20020412_00001.icfg	
16	GMIseq		cmd:seq2que	gmique.read(head)	
16	ICS		cmd:ics2aie	aie.configure 20020412_00001	

Table 5.: Observation Sequence: Mapped Dither Set (Part II)

Seq #	Process	From	To	Action	Response
16	GMItcs			read 20020412_00001.tcfg	
17	GMIque	data:fs2que	cmd:que2seq		observe 20020412_00001
17	ICS	resp:aie2ics			ack configure 20020412_00001
17	GMItcs		cmd:tcs2ext	configure 20020412_00001	
18	GMItcs	resp:ext2tcs			ack configure 20020412_00001
18	ICS	resp:aie2ics			ok configure 20020412_00001
19	ICS	resp:ics2seq			ok iconfig 20020412_00001
19	GMItcs	resp:ext2tcs			ok configure 20020412_00001
20	GMItcs		cmd:tcs2odb	gmiodb.update(20020412_00001)	
21	GMIodb	resp:odb2tcs			ack update 20020412_00001
22	GMIodb	resp:odb2tcs			ok update 20020412_00001
23	GMItcs	resp:tcs2seq			ok tconfig 20020412_00001
24	GMIseq		cmd:seq2ics	observe 20020412_00001	
25	ICS	resp:ics2seq			ack observe 20020412_00001
26	ICS		cmd:ics2gpx	gpxStart 20020412_00001	
27	ICS	resp:gpx2ics			ack gpxStart 20020412_00001
28	ICS	async:gpx2ics			progress 20020412_00001 33%
29	ICS	async:gpx2ics			progress 20020412_00001 66%
30	ICS	resp:gpx2ics			done 20020412_00001
31	ICS	resp:ics2seq			done 20020412_00001
32	GMIseq		cmd:seq2odb	gmiodb.update(done 20020412_00001)	
33	GMIseq		cmd:seq2que	gmique.read(head)	
33	GMIodb	resp:odb2seq			ack done 20020412_00001
34	GMIseq		cmd:seq2que	gmique.read(head)	
34	GMIodb		data:odb2dhs	gmiobd.list(20020412_00001)	
35	GMIque	data:fs2que	cmd:que2seq		tconfig 20020412_00002
35	GMIodb		data:odb2fs	gmiobd.list(20020412_00001)	
36	GMIseq		cmd:seq2tcs	gmitcs.load(20020412_00002)	
36	GMIodb	resp:odb2seq			ok done 20020412_00001

Table 6.: Observation Sequence: Mapped Dither Set (Part III)

Seq #	Process	From	To	Action	Response
37	GMItcs	resp:tcs2seq			ack tconfig 20020412_00001
38	GMIseq		cmd:seq2que	gmique.read(head)	
38	GMItcs			read 20020412_00002.tcfg	
39	GMIque	data:fs2que	cmd:que2seq		observe 20020412_00002
39	GMItcs		cmd:tcs2ext	configure 20020412_00002	
40	GMItcs	resp:ext2tcs			ack configure 20020412_00001
41	GMItcs		cmd:tcs2odb	gmiodb.update(20020412_00002)	
41	GMItcs	resp:ext2tcs			ok configure 20020412_00002
42	GMIodb	resp:odb2tcs			ack update 20020412_00002
43	GMIodb		data:odb2fs	create 20020412_00002	
44	GMIodb		data:odb2fs	copy 20020412_00001 20020412_00002	
45	GMIodb		data:odb2fs	gmiodb.update(20020412_00002)	
46	GMIodb	resp:odb2tcs			ok update 20020412_00002
47	GMItcs	resp:tcs2seq			ok tconfig 20020412_00002
48	GMIseq		cmd:seq2ics	observe 20020412_00002	
49	ICS	resp:ics2seq			ack observe 20020412_00002
50	ICS		cmd:ics2gpx	gpxStart 20020412_00001	
52	ICS	resp:gpx2ics			ack gpxStart 20020412_00002
53	ICS	async:gpx2ics			progress 20020412_00002 33%
54	ICS	async:gpx2ics			progress 20020412_00002 66%
55	ICS	resp:gpx2ics			done 20020412_00002
56	ICS	resp:ics2seq			done 20020412_00002
57	GMIseq		cmd:seq2odb	gmiodb.update(done 20020412_00002)	
58	GMIseq		cmd:seq2que	gmique.read(head)	
59	GMIodb	resp:odb2seq			ack done 20020412_00002
60	GMIque	data:fs2que	cmd:que2seq		tconfig 20020412_00003
60	GMIodb		data:odb2dhs	gmiobd.list(20020412_00002)	
61	GMIodb		data:odb2fs	gmiobd.list(20020412_00002)	
62	GMIseq		cmd:seq2tcs	gmitcs.load(20020412_00003)	

Gillies, K. 1996, *Programmer's Guide to the Generic WIYN Client*, v2, February.

Daly, P. N. 1995, *Porting CGS4DR to Unix*, in Astronomical Data Analysis Software and Systems IV, ASP Conference Series, Vol. 77, R. A. Shaw, H. E. Payne and J. J. E. Hayes (editors), pp375–378.

Currie, M. 1999, *UKIRT FITS Headers*, orac016-fith, draft 3, May.

A Observation Data Maintained by GMIodb

```
################################################################################
# NEWFIRM: PCONFIG data
################################################################################
SIMPLE  =                           T / File conforms to FITS standard
BITPIX  =                          32 / Number of bits per pixel
NAXIS   =                           2 / Number of data axes
NAXIS1  =                        4096 / Length of data axis 1
NAXIS2  =                        4096 / Length of data axis 2
COMMENT FITS (Flexible Image Transport System) format defined in Astronomy and
COMMENT Astrophysics Supplement Series v44/p363, v44/p371, v73,p359, v73,p365.
COMMENT Contact the NASA Science Office of Standards and Technology for the
COMMENT FITS definition document #100and other FITS information.
BSCALE  =                         1.0 / True_value = BSCALE * FITS_value + BZERO
BZERO   =                         0.0 / True_value = BSCALE * FITS_value + BZERO
BLANK   =                -2147483648 / Array value whose physical value is undefined
BUNIT   = 'ADU per integration    ' / Units of primary array
TELESCOP= 'Mayall 4m, KPNO, AZ    ' / Telescope name
INSTRUME= 'NEWFIRM                ' / Instrument
OBSERVER= 'Philip N. Daly         ' / Observer name(s)
PROPOSER= 'Philip N. Daly         ' / Proposer name(s)
PROPOSAL= 'NEWFIRM T&E            ' / Proposal name
PROPID  = '2002B-1234             ' / Proposal ID
USERID  = 'pnd                    ' / User identification code
OBSREF  = 'NEWFIRM T&E            ' / Observer reference
ORIGIN  = 'NOAO, Tucson, AZ       ' / Organization creating the FITS file
DATE    = '2002-04-12             ' / Date (UTC) when FITS file was created
################################################################################
# NEWFIRM: TCONFIG data
################################################################################
RABASE  = '08:08:29.84            ' / Offset zero-point in RA, FK5 [h]
DECBASE = '28:51:48.67            ' / DOffset zero-point in Dec, FK5 [deg]
RA      = '08:08:29.84            ' / RA of observation (hr)
DEC     = '28:51:48.67            ' / DEC of observation (deg)
EQUINOX =                    2000.0 / Equinox of object position FK5 (years)
TFOCUS  =                    5555.0 / Telescope focus position (microns)
TRAOFF  =                      80.0 / Telescope offset RA at equinox (arcsec)
TDECOFF =                      80.0 / Telescope offset DEC at equinox (arcsec)
GRAOFF  =                       0.0 / Guider offset RA at equinox (arcsec)
GDECOFF =                       0.0 / Guider offset DEC at equinox (arcsec)
################################################################################
# NEWFIRM: ICONFIG data
################################################################################
OBSTAG  = '20020412_00081         ' / Observation tag
OBSNUM  =                        81 / Observation number
DITHSET =                        75 / Dither set number
DITHNUM =                         7 / Sequence number within dither set
MAPSET  =                        75 / Map set number
OBSTYPE = 'OBJECT                 ' / Type {BIAS|DARK|ARC|FLAT|CALIB|OBJECT|SKY}
STANDARD=                         F / True if the object is a known standard
EXP_TIME=                      10.0 / Exposure time for each co-add [s]
INT_TIME=                      60.0 / Total integration time (sum of all co-adds) [s]
NEXP    =                         6 / Number of exposures in the integration
MODE    = 'STARE                  ' / Observing mode {STARE|NDSTARE}
SPD_GAIN= 'Normal                 ' / Readout speed {normal|fast} or high gain
GAIN    =                         6 / Electrons per data number
FILTER  = 'K                      ' / Combination filter name
FILTER1 = 'Open                   ' / Filter in position 1
FILTER2 = 'Kocli                  ' / Filter in position 2
```

```
################################################################################
# NEWFIRM: detector data
################################################################################
DETECTOR= 'Orion                ' / Detector name
DETSIZE = '[1:4096,1:4096]      ' / Detector size for DETSEC
NARRAYS =                    4 / Number of detectors (arrays)
NAMPS   =                   16 / Number of amplifiers
PIXSIZE1=                 15.0 / Pixel size for axis 1 [microns]
PIXSIZE2=                 15.0 / Pixel size for axis 2 [microns]
PIXSCAL1=                0.423 / Pixel scale for axis 1 [asec/pix]
PIXSCAL2=                0.423 / Pixel scale for axis 2 [asec/pix]
TEMP_ARR=                  4.0 / Temperature of array [K]
TEMP_ST1=                  9.4 / Temperature of stage 1 [K]
TEMP_ST2=                  4.7 / Temperature of stage 2 [K]
################################################################################
# NEWFIRM: astrometric and environmental data
################################################################################
DATE-OBS= '2002-04-12T13:24:24.53Z' / UTC of start of observation
DATE-END= '2002-04-12T13:24:25.39Z' / UTC of end of observation
CTYPE1  = 'RA---TAN             ' / Equatorial tangent-plane projection
CRPIX1  =                2048.5 / Pixel at reference point along axis 1
CRVAL1  =               52.9533 / RA at reference point [deg]
CDELT1  =             0.0000247 / Increment per pixel at reference point [deg]
CUNIT1  = 'deg                  '/ Physical units of axis 1
CTYPE2  = 'DEC---TAN            ' / Equitorial tangent-plane projection
CRPIX2  =                2048.5 / Pixel at reference point along axis 2
CRVAL2  =               52.9533 / Declination at reference point [deg]
CDELT2  =             0.0000247 / Increment per pixel at reference point [deg]
CUNIT2  = 'deg                  '/ Physical units of axis 2
CROTA2  =                  -1.1 / Angle of Dec axis wrt axis 2 measured ccw [deg]
AMSTART =              1.628028 / Airmass at start of observation
AMEND   =              1.628071 / Airmass at end of observation
UTSTART =              13.40972 / UTC at start of observation [h]
UTEND   =              13.42750 / UTC at end of observation [h]
AIR_TEMP=                   2.6 / External air temperature [Celsius]
DOMETEMP=                   3.2 / Dome air temperature [Celsius]
HUMIDITY=                  13.5 / Percentage air humidity
WIND_SPD=                   7.3 / Wind speed [km/h]
WIND_DIR=                   123 / Wind direction azimuth [deg]
################################################################################
# NEWFIRM: STATUS data
################################################################################
TERR    =                     0 / 32-bit telescope error flag
IERR    =                     0 / 32-bit instrument error flag
SERR    =                     0 / 32-bit sequence error flag
QUALITY =                     0 / 32-bit data quality flag
```

B Glossary

AIE: auxiliary instrument electronics.
ASCII: American Standard Code for Information Interchange.
CRC: cyclic redundancy check, a 2-byte or 4-byte checksum for verifying data integrity after transmission.
DCA: data capture agent, a process that listens for data and re-assembles DCONFIG: a detector configuration.
it from independent chunks.
DHS: data handling system (*a.k.a* the data reduction pipeline).
FITS: Flexible Image Transport System, a standard for image interchange in astronomy.
GMI: generic manager for instruments, a conceptual, instrument independent, observation sequencer.
GMIQUE: the GMI-compliant queue manager process.
GMITCS: the GMI-compliant *TCS* manager process.
GMISEQ: the GMI-compliant sequence manager process.
GMIODB: the GMI-compliant observation manager process.
GPX: generic pixel server, a proposed open-source image acquisition system being developed by NOAO.
GWC: generic WIYN client, a client-server message routing architecture used a the WIYN, Mayall and Blanco telescopes.
ICONFIG: an instrument configuration.
ICS: instrument control system.
MONSOON: an implementation of the *Generic Pixel Server* (*GPX*) being developed at NOAO.
NEWFIRM: NOAO extremely wide-field infrared mosaic camera.
NOAO: National Optical Astronomy Observatories.
PCONFIG: a proposal/programme configuration.
RTD: real-time display.
STB: save the bits, NOAO's standard archiving system.
TCONFIG: a telescope configuration.
TCS: telescope control system.
UTC: universal time (*a.k.a.* zulu or GMT).
WIYN: Wisconsin Indiana Yale and NOAO consortium.