

NGAO

Hard Real Time

and

Soft Real Time

Controller

Interface

Ver. .11

Table of Contents

[1. Introduction and Overall Requirements 1](#_Toc227502878)

[2. Architecture of the HRT 2](#_Toc227502879)

[2.1 Functions of the HRT Control Processor (HRT CP) 3](#_Toc227502880)

[2.2 Functions of the HRT Computation Engine (HRT CE) 3](#_Toc227502881)

[2.3 HRT States 3](#_Toc227502882)

[3. Interface between the SRT and the HRT 4](#_Toc227502883)

[3.1 SW Protocols 4](#_Toc227502884)

[3.1.1 Introduction 4](#_Toc227502885)

[3.1.2 Basic Interface Description 4](#_Toc227502886)

[3.1.3 Message Headers 4](#_Toc227502887)

[3.1.4 Message Types 5](#_Toc227502888)

[3.1.5 Message bodies 7](#_Toc227502889)

[3.2 Physical Layer 11](#_Toc227502890)

[4. Datums Transferred between the HRT and the SRT 12](#_Toc227502891)

[4.1 AO Control Settings 12](#_Toc227502892)

[4.1.1 General Operations 12](#_Toc227502893)

[4.1.2 AO Status 13](#_Toc227502894)

[4.1.3 Current AO parameters 13](#_Toc227502895)

[4.1.4 Internal Error Conditions 13](#_Toc227502896)

[4.2 WFSs (NGS, LGS, Tip/Tilt, Truth) 14](#_Toc227502897)

[4.3 Tomography 14](#_Toc227502898)

[4.4 Command generation for DMs and Tip/Tilt 15](#_Toc227502899)

[4.5 Telemetry, Diagnostics and Disk Farm 15](#_Toc227502900)

[4.5.1 Disk Farm 15](#_Toc227502901)

[4.5.2 Telemetry and Diagnostics 15](#_Toc227502902)

[4.6 System Validation and Self Test 16](#_Toc227502903)

[4.7 Calibration and Debug 16](#_Toc227502904)

**Table of Figures**

[**Figure 1 Connection architecture for purposes of communication** 2](file:///C%3A%5CDocuments%20and%20Settings%5CMarco%5CDesktop%5C___LAO%5CNGAO%5CActive%20Docs%5CNGAO%20HRT-SRT%20Interface%204.docx#_Toc227502905)

[Figure 2: The Logon request and response message bodies. 7](#_Toc227502906)

[Figure 3: The Logoff request and response message bodies. 8](#_Toc227502907)

[Figure 4: The Symbol Lookup request and response message bodies. 8](#_Toc227502908)

[Figure 5: The Data Read request and response message bodies. 9](#_Toc227502909)

[Figure 6: The Cancel request and response message bodies. 10](#_Toc227502910)

[Figure 7: The Data Write request and response message bodies. 10](#_Toc227502911)

[Figure 8: The Server Status request and response message bodies. 11](#_Toc227502912)

[Figure 9 11](#_Toc227502913)

# Introduction and Overall Requirements

This document does not include the SRT interface to the HRT disk farm, which will be covered separately.

1. The HRT system directly controls all AO HW and performs all AO computations. During normal operation, the HRT operates as a slave to the SRT.
2. The interface between the SRT and the HRT must be simple to implement and clear to understand.

It uses simple socket interfaces for control coming from the SRT to the HRT and for data to and from the HRT.

1. Anything that can be set in the HRT can be set by the SRT and can be read back by the SRT.

Any control that can be set in the HRT by the SRT system must be directly readable by the SRT unless explicitly excused from this requirement, i.e. if the HRT buffers a control parameter for convenience, when the SRT reads the control parameters from the HRT, they must be taken from the final control variables, not the buffer.

1. Anything set on the startup of the SRT can be set again by command, except for the FPGA configuration data, which requires a partial or complete reconfiguration step. However, the non-volatile memory holding the FPGA configuration information can be loaded by the SRT. However, as stated, it will not be loaded into the FPGA’s until a reconfiguration cycle occurs.
2. There will be debug and diagnostic commands that may not be appropriate for a normal user GUI. However, the GUI should provide a mechanism (command line, etc.) that allows access to these commands.

# Architecture of the HRT

This document does not include the SRT interface to the HRT disk farm, which will be covered separately.

The HRT system directly controls all AO HW and performs all AO computations. During normal operation, the HRT operates as a slave to the SRT.

It has simple socket interfaces for control to the HRT and for data to and from the HRT.

The AO Control Processor will be the interface to the SRT. It will perform a translation of the SRT commands into machine specific control.

The HRT Computation Engine performs all real-time calculations.

On power up or system reset, it will perform all initialization required, using local non-volatile memory. This configuration data can be set by the SRT. However, to load it into the FPGA’s requires a configuration cycle that will interrupt normal operations.

It will retain the current configuration of the system and on power up or system reset it will restore that configuration.

The following states will not be restored on a reset, but will always be in the following states:

* The AO system will be disabled.
* Debug capabilities will be disabled.
* The use of dummy inputs/outputs from any module will be disabled.

**SQL Interface**

***HRT***

**HRT**

**Control**

**Processor**

**HRT**

**Computation**

 **Engine**

**Cameras**

**DMs/TT Mirrors**

**Bit-Stream**

**Interfaces**

**Disk Farm**

**SRT/HRT**

**Interfaces**

***SRT***

***SRT***

**SRT/HRT**

**Interfaces**

**Figure 1 Connection architecture for purposes of communication**

## Functions of the HRT Control Processor (HRT CP)

The HRT CP supplies asynchronous interfaces between the HRT and SRT, using socket interfaces, over industry standard HW and protocols. Commands and data will be in a TBD format, but will be simple and easy to parse.

Commands will be ASCII text. Data will be binary or ASCII as appropriate for the type of data being returned.

This allows easy communication of between the SRT and the HRT as well as giving the ability to use command line over telnet during debug or system diagnostics.

The HRT CP will parse commands sent from the SRT and pass the HRT Computation Engine (HRT CE) the correct binary streams on its interfaces synchronized to the appropriate HRT CE clocks and start of frame.

The HRT CE has no communication buffering capabilities and thus requires the HRT CP to perform both the parsing and synchronization.

The HRT CP controls the Disk Farm and will retrieve and save any configuration data or file information needed for operation under control of the SRT.

It controls other functions of the HRT AO system through the HRT PE.

During operation, the HRT CE can be instructed to automatically save data such as camera images, tomography data, etc. during operation.

The Disk Farm does not contain science data, but only AO Configuration data AO operational or analytical data stored during real-time operation. For this reason, accessing this data into the Disk Farm or accessing data from it by the SRT may only be done when the system is in the AO OFF state to avoid potential interference with the real-time operation.

## Functions of the HRT Computation Engine (HRT CE)

The HRT CE performs all real time calculations for and control of the AO system.

It runs directly under the control of the HRT CP, which is a slave to the SRT.

## HRT States

The HRT has the following states, which the SRT can view and control.

# Interface between the SRT and the HRT

Adopted from Keck Adaptive Optics Note 652, “Keck Next Generation Adaptive Optics, Real Time Computer Interface Concept”, Revision 1: April 14, 2009

## SW Protocols

### Introduction

This document describes a concept for the interface between the AO Control System and the Real Time Controller (RTC) for the NGAO system. A similar type of interface has been successfully used in the past by Jimmy Johnson at Measurex/Honeywell and was adopted as a standard by the company. It provides a robust and flexible means of bidirectional communication of commands, status, and parameter data. The concept is presented here in a primarily pictorial fashion as a starting point for more detailed design efforts.

### Basic Interface Description

The interface was originally designed for a client-server architecture with multiple clients. For the case of NGAO, the RTC implements the server and the AO Controls function implements a single client. This discussion assumes a generic communications layer between the two, which will most likely be implemented using Ethernet and TCP/IP. The basic interface consists of messages that are sent back and forth between the client and the server. The message sent from the client to the server is called a “Request” message and the message sent from the server to the client is called a “Response” message. All communication is initiated by the client, and the server only sends messages in response to client requests. Each message consists of a message header and a message body. The message headers are standardized, one type for request messages and another type for response messages. The message body contents are specific to the particular request or response, and typically consist of symbols (parameters), their associated meta-data, and their values. It is the responsibility of the client to ensure that all data are formatted appropriately for the server (data type, byte order, etc.) and to reformat data received from the server, as required.

### Message Headers

The structure of the message headers is shown below in Figure 1. For simplicity, we will not concern ourselves with the data types or field alignment in this discussion.



Figure 1: The structure of the message headers.

The message header fields are defined as follows:

* Message Length: The length of the entire message (message header and message body) in bytes. This includes the message length field itself. A single message may be sent out in several pieces. The message length is always the first field and is used by the network interface to determine how many read calls are needed to receive an entire message.
* Client ID: A unique identifier representing the client. The server does not use this field and merely copies it to the Client ID field when generating a response message. For our purposes, this field may not be needed. Our current thinking is that there will only be a single client, the AO Controls function.
* Message type: This field identifies the message type. The server uses this field to identify the requested service. Message types are described in more detail below.
* Message ID: A unique ID assigned by the client to each request message. The server copies this field to all messages generated in response to a request. The client may use the Message ID field as desired, the most obvious use being synchronization between requests and responses. Other possible uses include storing pointers or hash code values for use by the client when it processes the response messages.
* Timestamp: The timestamp field contains the server time at which the response was generated.
* Status: A field generated by the server to indicate the completion status of the request. The value will either indicate success or an error condition. Note that an error condition need not imply a fatal error, but could indicate a partial success.

### Message Types

The message body is dependent on the type of message being sent, so we now turn our attention to the different message types used in the interface. The message types described below are candidate types and may not all be required or implemented for the RTC interface. Moreover, we may choose to implement additional types, as needed. Each message type described below is implemented as a request message from the client to the server and an associated response message from the server to the client. The following message types are currently defined:

* Logon: A message used to establish a connection between the client and the server. Although it is primarily intended to implement security in the client interface, the logon message could be used in NGAO to initiate communications between the AO Controls function and the RTC.
* Logoff: A message used to terminate a connection and all communications between the client and the server. Upon receipt of this message, the server will cancel all pending requests and delete all temporary internal data structures used to support the connection. For NGAO, this could be used when putting the AO system into a standby state.
* Symbol lookup: The symbol lookup message is used to request meta-data for any of the symbols used by the server. Assuming that all the symbols used are well defined and known to both sides of the interface, this message type may not be required for the RTC interface.
* Data read: A message used to read data from the server. The data transfer for a data read can be configured to be on-demand, periodic, or event based:
	+ On-demand: The client sends a request message and the server replies with a response message.
	+ Periodic: The client sends a request message which the server uses to configure the periodic read. The server then sends response messages with data at the requested periodic rate.
	+ Event based: The client sends a request message which the server uses to configure an event based read. The event is represented by a change in a server parameter. Upon sensing the change, the server sends a response message with the requested data. The event based trigger could be used to permit the RTC to communicate a particular kind of real-time event to the AO Controls host function.
* Cancel: A message used to cancel a previous periodic or event based read request.
* Data write: A message used to write parameter data from the client to the server.
* Server status: A message used to get the current status of the server. It can also be used as a heartbeat mechanism to indicate to the server that the client is still alive.

### Message bodies

This section describes the message bodies defined above in sufficient enough detail that the interface described in this document can easily be specified with real data types and field lengths and used as the basis for an interface design and implementation. For each message type, the success of the request is returned in the status field of the response header along with a timestamp.

#### Logon

The logon request and response messages are shown in below. The request has four parameters in addition to the header, while the response consists of only the response message header. The request parameters are defined as follows:

* Logon Option: Used to determine whether the server will apply timeout control to this connection. Under timeout control, if no requests are received during some default timeout period, the connection is terminated by the server.
* Version Number: This field can be used to ensure synchronization between the version ID of the client and sever interface SW.
* User: The user name (not required for NGAO).
* Password: The password (not required for NGAO).



Figure 2: The Logon request and response message bodies.

#### Logoff

The logoff request and response messages are shown in below. The request and the reply consist solely of the message headers.



Figure 3: The Logoff request and response message bodies.

#### Symbol Lookup

The symbol lookup request and response messages are shown in below. The request body contains the number of symbols to be looked up by the server, followed by a list of the symbol names. The server response consists of the number of symbols found followed by the symbol names and the meta-data for each symbol.



Figure 4: The Symbol Lookup request and response message bodies.

#### Data read

The data read request and response messages are shown below in . The read request has two responses: an initial status response to acknowledge the read request, and a read response which contains the data. The two response message types are required to support periodic and event-based reads. In those cases, a single read request sets up the read and is acknowledged by a status response message. Subsequent to that, every time the trigger or periodic conditions are met, a read response message is sent by the server. The Message ID field is used by the client to correlate the response messages with their originating request message.

 The read request message consists of a trigger name, trigger type, and rate, which are used to set up event-based or periodic reads, followed by the number of symbols requested and a list of their names. The read status response returns information on the trigger followed by the number of symbols to be returned by the read along with their names and meta-data. The meta-data includes an offset field which is the offset in bytes of the data from the beginning of the read response message(s) that will follow. The read response message returns the number of symbols in the data, followed by the data status (read status) and the data for each symbol.



Figure 5: The Data Read request and response message bodies.

#### Cancel

The cancel request and response messages are shown in below. The request and the reply consist solely of the message headers. The Message ID field is used to indicate to the server which read request to cancel.



Figure 6: The Cancel request and response message bodies.

#### Data write

The data write request and response messages are shown below in . The write request message consists of the number of symbols followed by a list of the symbol names and their meta-data. The meta-data is used by the server to validate that the data from the client is consistent with what it expects. The data write response message consists of the number of symbols followed by a list of their names and the write result statuses.



Figure 7: The Data Write request and response message bodies.

#### Server Status

The server status request and response messages are shown below in . The status request message consists of only a message header. The status response message consists of a status word followed by the number of outstanding requests that are pending.



Figure 8: The Server Status request and response message bodies.

Figure

The SRT interfaces to the HRT through the HRT Controller Processor. During development, this will be a standalone PC based processor running Linux. However, its functionality may be absorbed into the SRT at a later stage of development or integration.

In either case, the drivers for the selection of hardware and software interfaces will be modern, high-speed, industry standard, COTS solutions. The following are only examples of what could be used.

## Physical Layer

Gbit Ethernet, PCI-X, hardwired or fiber

# Datums Transferred between the HRT and the SRT

At the lowest level, the bit-stream interface between the HRT Controller Processor and the inputs to the HRT Compute Engine, are through bit streams on ports synchronized to the start of a frame. This is because the HRT runs as a finite state machine, possibly a systolic array processor, and hence does not support standard PC communications buffering.

The HRT Controller Processor functions to absorb asynchronous commands from the SRT/HRT interface and formats them for the required bit stream interface to the HRT Compute Engine.

The HRT provides the facility for the SRT to set all parameters and to query the state of all parameters listed. Additionally, some the parameters the SRT may set may result in data being sent to the Telemetry or Diagnostic data streams for display and analysis by the SRT or storage by the disk farm.

Need to document which of these are transparent to the operation of the system and which will require that the operation of the system be interrupted

## AO Control Settings

In general, the HRT CP keeps track of what hardware is connected and reports the status to the SRT. The preferred implementation is to allow the HRT CP to determine what devices or modules are actually connected at any given time and not to rely on a table entry that might have been valid at some previous time. This is because the state of the system may change unexpectedly due to device hardware or software failure, manual removal, etc.

The following are parameters that should correspond to one or more commands to read or write them.

All parameters are read/write unless otherwise specified.

### General Operations

Set and read the following system parameters and states

* AO On/Off
* Open/Closed loop operation
* Low level restart (write only)
* Complete restart (write only)
* Abort long restart self tests (write only)
* Vibration Compensation On/Off
* Vibration compensation tables (TBD)
* Wind compensation On/Off
* Wind compensation tables (TBD)
* Number of sub apertures to use (32/64). Is this something that an observer might change during operations or just at the start of the run?
* Debug value
* Use dummy camera for WFS
* Use dummy centroid data for reconstruction
* Use dummy WF data for tomography
* Use dummy layer data for DM command generation
* Use dummy DM commands for the DM
* AO control code
* FPGA configuration data (requires a low level restart to have an effect)
* Set the fixed number of tomography iterations to perform each frame. If this number iterations is not completed in a single frame, an error condition will be set and AO processing will be stopped.

### AO Status

All status conditions are reported asynchronously to the SRT at a pre set rate. The SRT should filter any it does not want to monitor.

* Set status conditions to monitor (bit mapped word)
* Iterations to converge
* Start of frame occurred before convergence (when using a convergence limit instead of a fixed number of iterations)
* Residual tomography error per WFS
* Inlet cooling temperature
* Die temperatures
* Board Voltages

### Current AO parameters

* Current configuration data
* Number of cycles to converge
* Residual tomography errors (per WFS)

### Internal Error Conditions

All error conditions are reported automatically asynchronously to the SRT at a pre set rate. The SRT should filter any it does not want to monitor at its end.

* Error condition pending (bit mapped word)
* Error conditions pending: Identifies any error condition in the HRT system. Writing 0 clears all errors.

The error word can be set to non-zero values to test the error handling capabilities of the system.

* Set error report rate (1 to 10 Sec)
* Number of fixed iterations did not complete before the end of frame
* SEU
* Link error to disk farm
* Link error to SRT
* Over temp on cooling inlet
* Over temp on cooling outlet
* Over temp on chip (Per FPGA)
* Voltage error on board (per board, per supply)
* Parameter load exceeded frame time

## WFSs (NGS, LGS, Tip/Tilt, Truth)

* List of active cameras (bit mapped word)
* Master camera number (source of frame clocks for all cameras)
* Frame rate, gain and other camera parameters: clock freq, divisor, etc. (sent to individual cameras)
* Threshold Level (Sent to each WFS)
* Dark Level array (Sent to a WFS)
* Dark level array integration count (Sent to each WFS)
* Save Dark Level ( per camera) (Read only)
* Reference Centroids (Sent to a WFS)
* Save Reference centroids (From a WFS) (Read only)
* Bad pixel list (sent to each WFS)
* Camera Images (Raw camera data)
* List of active WFS (bit mapped word, sent to tomography engine)
* Reconstruction algorithm selection (sent to each WFS)

#### LOWFS

#### HOWFS

## Tomography

* Cn2, gain, consolidated parameters per layer
* GS shift per guide star per layer
* GS scaling per layer
* Kolmogorov filter
* Preconditioning Matrix per WFS
* Fourier tables (per sub aperture)
* Aperture function
* Distortion correction (TBD)
* List of active layers (bit mapped word)

## Command generation for DMs and Tip/Tilt

* List of active DMs (bit mapped List)
* Layer consolidation parameters (MCAO)
* Science object parameters (MOAO)
* Non-linear look up tables (per DM)
* Displacement to Force Matrixes (per DM)

## Telemetry, Diagnostics and Disk Farm

When a configuration, telemetry, diagnostic or Disk Farm command is given, no further parameters will be updated until the command has finished. If the SRT needs to give a sequence of commands, it will need to buffer them.

### Disk Farm

The data stored in the disk farm is not science data, but AO diagnostic data.

### Telemetry and Diagnostics

The following actions can be performed for telemetry and diagnostic data streams:

* Set data rate for diagnostics
* Set data to capture for telemetry (bit mapped word)
* Capture continuously, On/Off
* Set data capture length in frames (array of lengths)
* Set data to capture for diagnostics (bit mapped word)
* Set data update rate for diagnostic data (0.5 to 10 sec)

The following information is available for inclusion in the telemetry and diagnostic data streams:

* Sub millisecond accurate clock time
* Raw Camera Images
* Centroids
* Centroid intensities
* Reconstructed wave fronts
* Tomographic layer data
* DM wave fronts prior to command generation and Split Tomography consolidation
* DM commands
* Tip/Tilt values
* Tip/Tilt commands
* Split Tomography data (TBD)
* Chip temperature, board voltages, inlet and outflow temperatures
* All current settings of all AO parameters are included in a file header/attachment.

## System Validation and Self Test

The following are specific tests run at startup or on command.

* End-to-end test: to answer the question, “Given a known set of pseudo camera images, and a known set of WFS, tomography and DM parameters, do we generate the correct DM commands?”
* Perform specific predefined tests for the following. These are primarily for debug or calibration:

Cameras

WFSs

Centroids

Reconstruction

Tomography

DMs and T/T

* Detailed Debug

For detailed system debug, dump specific processor data not otherwise covered, may require specially compiled RTC code.

## Calibration and Debug

* Set alignment laser diode In/Out
* Alignment laser diode On/Off and In/Out
* Phase plate or turbulence generator In/Out
* Turbulence generator speed
* Load arbitrary system patterns from files:

Pseudo camera images

Arbitrary centroid data

Arbitrary WFS data

Arbitrary layer information

Arbitrary DM commands