University of Durham
Astronomical Instrumentation Group

# NAOMI

## Nasmyth Adaptive Optics for Multi-purpose Instrumentation

## The Real Time Control System
## *User Guide*

wht-naomi-25

Version: 0.1   2 August 2002

Author: Richard Myers[1], Stepehn Goodsell[2]

[1] University of Durham, Department of Physics,
[2] Isaac Netwon Group of Telescope

# 1. Scope

A separate document "*Real Time Control System Programmer's Guide*" provides a description of the structure and details of the C40 software itself. For a general perspective on the C40 software, its purposes and development history, as well as a glossary of common terms, one should also consult the early, introductory sections of that document.

# 2. Overview

The NAOMI real-time C40 software is normally accessed via higher-level graphical user interfaces (such as `TopGui`) or scripts (such as `AutoConfig`). These applications use the NAOMI sequencer and EPM (Electra process monitor) to coordinate manipulation of the C40s with control of other NAOMI systems (for example, EPICS mechanism control or reconfiguration of the SDSU Wavefront Sensor CCD controllers). There is, however, a set of commands that access the C40s directly, bypassing the Sequencer and EPM. The purpose of these commands is for engineering access, troubleshooting, or for unusual activities such as detailed component recalibration. This document describes these commands.

There is also a suite of GUI applications at the sub-sequencer level (`WFSAlign`, `MirrorMimic`, `ReconMimic`, `DataDiag`), inherited from Electra, and now superceded by `TopGui`. These are not documented here, but will be included in the little-used software section of the *Real Time Control System Programmer's Guide*.

The commands described in this document are divided into three main categories: `WFS`, `SG` and `RT`. These denote Wavefront Sensor, Strain Gauge, and (generic) RealTime commands respectively. The appropriate prefix (`WFS`, `SG` and `RT`) must be prepended to the commands as indicated in the syntax summaries below. The other frequently used command of this kind is `c40Run`, which is currently only available on aocontrol1`.` It takes a single parameter which is the name of a C40 *realtime configuration* to boot and set up. The default realtime configuration is `Naomi` and it is used by the `NaomiRestart` script and the corresponding buttons on `TopGui`.

## 2.1 *Bug on Power Cycle of aocontrol1*

There is currently a problem using `c40Run Naomi` (or the equivalent use of just `c40Run` or `NaomiRestart` or the boot/reset buttons on `TopGui`) after a power cycle of aocontrol1. It is recommended logging into aocontrol1 (for example by `rsh` from the `naomi` account on navis) and using the command:

```
c40Run NaomiInterleaveZeiss.
```

This will fix the problem until the next power cycle of aocontrol1. The reason for this behaviour is unknown at present.

# 3. Commands

If entered with missing parameters these commands will print a syntax indication as part of the error message. If the prefix is entered alone then a list of commands, each normally accompanied by a short help string, will be printed. Some commands may be printed that have been removed from the list below. These are *very* obscure and should be avoided.

Where command arguments have been indicated below [<arg>] stands for an optional argument.

## 3.1 WFS commands

```
WFS AddWFSoffsets
```

adds (optionally gain-multiplied) offsets from a file to the current ones

```
WFS BreakTransaction
```

Break the (real-time level) WFS transaction lock

```
WFS ClearHiccup
```

Clear RAL WFS hiccup - ELECTRA only

```
WFS CloseLoop
```

close the WFS loop

```
WFS ConvertFromOldWFSoffsets
```

convert a WFS offset dataset from the old offset format to configutilities format

```
WFS ConvertToOldWFSoffsets
```

convert a WFS offset dataset from the configutilities format to the old file format

```
WFS EstimateFrameInterval
```

Estimate WFS frame gap

```
WFS GetBackgroundFluxMemory
```

Get background flux memory (temporal filter constant)

```
WFS GetCentroidBias
```

Get Centroid Bias

```
WFS GetCentroidDecimate
```

Get Centroid Decimate

```
WFS GetCentroidWeight
```

Print pixel weighting for centroiding

```
WFS GetCentroids
```

Print Live Centroid XY data

`WFS GetDelay`

Get the inter-frame interval

`WFS GetFluxMemory`

Get flux memory (temporal filter constant)

WFS GetFrameProcess

Get Frame processing (SDSU)

`WFS GetGain`

Get segment loop gains

`WFS GetLiveTipTilt`

print out tip-tilt value from diagnostics

`WFS GetLoopState`

Gets current state of segment control loop

`WFS GetPixelDecimate`

GetPixel Decimate

`WFS GetPixels`

Print Live pixel data

`WFS GetReconstructorMatrix`

Get the reconstructor matrix

`WFS GetSDSUstatus`

Get SDSU status information

`WFS GetSelectApp10`

Get App10 tip-tilt only mode select switch

`WFS GetSurrogateWFSapp`

Get Surrogate WFS mode which the C40s simulate from full frame data

`WFS GetTTonlyGains`

Get the global tip-tilt only mode x and y gains

`WFS GetTTonlyLimit`

Get the global tip-tilt only mode limit

`WFS GetTipTiltGain`

Get the global tip-tilt offload x and y gains

`WFS GetWFSoffsets`

Get the WFSoffsets in configutilities format

`WFS LockDAC`

Control the WFS loop state (closed/open)

`WFS MultiplyWFSoffsets`

Multiply the WFSoffsets by a gain

`WFS OpenLoop`

open the WFS loop

`WFS PrintLiveXYZtoABC`

print the geometry matrices in a reasonable format

`WFS RalSetupCentroidBias`

Determine the bias values for centroiding

`WFS RingInit`

Initialise WFS ring - not required in normal operation

`WFS RingStart`

Starts realtime ring running. Argument is number of iterations to do

`WFS RingStatus`

Get WFS ring status

`WFS RingStop`

Stop WFS ring

`WFS RingTest`

WFS ring test - not appropriate for normal operation

`WFS SaveCentroidBias`

save Centroid Bias to a dated configutilities format file

`WFS SaveState`

Save RT parameters

`WFS SaveWFSoffsets`

save WFS offsets to a dated configutilities format file

`WFS SetBackgroundFluxMemory`

Set up decay constant for background flux low-pass filter

`WFS SetBackgroundWeight`

Set up background weighting for centroiding

`WFS SetCentroidBias`

Sets the WFS centroid offsets from a named ConfigUtilities file

`WFS SetCentroidDecimate`

Set the centroid decimation factor

`WFS SetCentroidWeight`
Set up pixel weighting for centroiding
`WFS SetDelay`
Set the inter-frame interval

`WFS SetFluxMemory`
Set up decay constant for flux low-pass filter

`WFS SetFrameProcess`
Setup Frame processing (SDSU)

`WFS SetGain`
Set the segment loop gains

`WFS SetInterval`
Set timer interval for dummy algorithm

`WFS SetPixelDecimate`
Change the decimation of pixel data in the WFS ring

`WFS SetQuickPistons`
Switches on/off the option to process pistons in the current
rather than the next frame

`WFS SetSegmentTiltLimit`
Set limit on segment tilt excursions

`WFS SetSelectApp10`
Switches on/off the option to interpret SDSU 0 frames as mode 10
rather than mode 8

`WFS SetState`
Restore saved RT parameters

`WFS SetSurrogateWFSapp`
Set Surrogate WFS mode which the C40s simulate from full frame data

`WFS SetTTonlyGains`
Set the Tip-tilt only mode x and y gains

`WFS SetTTonlyLimit`
Set the Tip-tilt only mode limit

`WFS SetTipTiltGain`
Set the global tip-tilt offload x and y gains

`WFS SetWFSoffsets`
Sets the WFS centroid offsets from a named ConfigUtilities file

`WFS SetWFSoffsetsToCurrentCentroids`

Sets the WFS offsets to the current centroids

`WFS SetupCentroidBias`
Setup the WFS centroid bias using current pixel values

`WFS SetupDefaultWFSoffsets`
Saves the WFS centroid offsest into a new version of the default ConfigUtilities file

`WFS Start`
Starts WFS ring running. See also RingStart

`WFS Status`
Get WFS ring status

`WFS Stop`
Stop WFS ring

`WFS TestWFSoffsets`

WFS WFSdummyMode
Set the WFS ring into dummy mode

`WFS WFSringBoot`
Boot and setup the wavefront sensor ring

`WFS ZeroCentroidBias`
zero WFS centroid bias

`WFS ZeroReconstructorMatrixEntries`
Zeros reconstructor matrix rows relating to segment list

`WFS ZeroReconstructorMatrixMap`
Zeros reconstructor contributions/pistons for matrix according to seg maps

`WFS ZeroWFSoffsets`
Zeros the WFS offsets

`WFS test`
test command line interface

## *3.2 SG commands*

`SG BreakTransaction`
Break a SG transaction

`SG ChanPrint [<DACchannel>]`
Print information about a DAC channel. This includes the mirror segment, ID and corresponding ADC processing CPU, if applicable.

`SG Close`

`SG DAtoSegAct`
`SG GetADCcapture`
Get SG ADC capture diagnostics

`SG GetADCcaptureDecimate`
Get SG ADC capture diagnostics decimation value

`SG GetADCcaptureSettings`
report StrainGauge ADC capture channels
for requested CPUs (default: all SG ADC CPUs)

`SG GetADCsnapshot`
Get SG ADC snapshot diagnostic data

`SG GetADCsnapshotDecimate`
Get SG ADC snapshot decimate value

`SG GetCurrentADC`
Get live SG ADC values

`SG GetDACsnapshot`
Get DAC snapshot diagnostic data

`SG GetDACsnapshotDecimate`
Get SG DAC snapshot decimate value

`SG GetFeedforward`
get the SG feedforward flag

`SG GetFinalDemand`
retrieve mirror (final demand) flat from SG mirror interface CPU

`SG GetFinalTipTilt`
Get mirror final demand from the SG mirror interface CPU

`SG GetGain`

`SG GetInputDemand`
Get mirror input demand flat from SG mirror interface CPU

`SG GetInterval`
Get the SG cpu interrupt interval

`SG GetSynchronise`
Get the SG synchronise flag and last synch clocks parameter

`SG GetTemperature`
Get the current temperature probe values

`SG GetWaveform`

`SG GrabMirror`
Retrieve mirror (input demand) from SG mirror interface CPU
and write back as a new flat to the WFS ring mirror control CPU

`SG Log`
print log on SG cpu

`SG MultiplyCalOffsets`
Multiply current SG cal offsets by factor

`SG OffsetDrift`
Figure out how much the strain gauge offsets have drifted

`SG Open`
Open the SG control loops

`SG PrintPass`
Print the SG passwthrough table

`SG RestoreOffsets`
Restore SG cal offsets to defaults

`SG Run`
boot and route the SG ring CPUs

`SG SGsetInterval`
Set the SG CPU interrupt interval

`SG SaveMirror`
Save mirror (input demand) from SG mirror interface CPU

`SG SetADCcaptureDecimate`
Set the SG ADC capture diagnostic decimation

`SG SetADCsnapshotDecimate`
Set the SG ADC snapshot diagnostic decimation

`SG SetCapture`
select a StrainGauge ADC contiguous capture diagnostics channel
the argument is a DAC-order channel number. The appropriate CPU
is selected and any previous selection on that CPU is overridden

`SG SetDACsnapshotDecimate`
Set the SG ADC capture diagnostic decimation

`SG SetFeedforward`
Set the SG loop feedforward flag

`SG SetGain`

Set the SG loop gain

`SG SetRectangularWaveform`

`SG SetSineWaveform <dchan> <freq> <baseline> <amplitude>`

Inject a sine wave onto a DAC output. Used to drive a DM segment
Or FSM channel with a single frequency in order to perform bandwidth tests.
The parameters are as follows:
<dchan> is the DAC channel number. 30 and 31 are FSM tip (X) and tilt (Y).
<freq> the Sine wave frequency in Hz
<baseline> the base value of the sine wave in DAC units (0 – 8191)
<amplitude> the amplitude of the sine wave in DAC units (0 – 8191)

It is possible to have one waveform set up per SG ADC CPU. However it is necessary to know which DAC channels are processed by which ADC CPUs to set this up.
To find out the ADC CPU (and mirror segment ID if applicable)assosicated with a DAC channel number use :

```
SG ChanPrint <dacChan>
```

To stop a waveform use `SG StopWaveform`

`SG SetSynchronise`

set the WFS/SG loop synchronisation flag

`SG SetWaveform`

set a waveform output channel
arguments are the DAC channel number, the repeat flag (1=repeat), and the waveform list itself

`SG SetupCalOffset`

Setup default SG cal offset

`SG SmoothClose`

Close SG loops whilst maintaining mirror position

`SG SmoothOpen`

open the strain gauge loops smoothly (by transferring the final demand back to the WFS loop).

`SG Start`

Start framing

`SG Status`

print SG CPU ring status

`SG Stop`

Stop framing

```
SG StopWaveform [<CPU>]
```
switch off a waveform output channel
argument is a list of CPU IDs (default: all)

```
SG TestMirror
```

```
SG test
```
test command line interface

### 3.3 RT commands

```
RT BreakSGtransaction
```
Break an SG ring transaction

```
RT BreakWFStransaction
```
Break a WFS ring transaction

```
RT DebugComport
```

```
RT GetAlgorithmVersion
```
Print the version number of the running algorithm

```
RT PrintDebugLog
```
Print the realtime debug log buffer

```
RT test
```

## 4. Bandwidth measurement tools

To inject a test sine wave onto the FSM or a DM segment use the commands

```
SG ChanPrint [<DACchan>]
SG SetSineWaveform <dchan> <freq> <baseline> <amplitude>
SG StopWaveform [<CPU>]
```

The full syntax for these commands is described above. If it is desired to inject a sine wave into the whole system then this can be done with the Nasmyth Calibration Unit (NCU). An internal pinhole may be deployed under EPICS control by the “Mask” control in the Calibration Unit panel on Mechanisms page of TopGui. The light from this mask bounces off a special additional tip tilt mirror manufactured by PhysikeInstrumente (PI). A PI controller is available in the NAOMI mechanisms rack (about halfway down in an individually mounted box). Sine or pink noise test waveforms may be injected into this controller using a special test unit built by Simon Tulloch. Particular attention should be paid to whether the quality of the depoloyable mask pinholes is appropriate for the work to be undertaken.

Responses to waveforms (open or closed loop) may be measured using the WFS or an external detector, such as JOSE. An example of measuring bandwidth using the WFS is under development in the python procedure:

```
Navis:/home/rmm/Electra/RealTime/pythonModules/MeasureBandwidth.py
```

To set up as a developer in order to have a private staging area version of this software, consult the *Real Time Control System Programmer's Guide*. To invoke the procedure, go to the directory and use

```
./MeasureBandwidth.py bandwidthSweepNaomi.py
```

where `bandwidthSweepNaomi.py` is an editable configuration file which is used to specify the measurements to be taken. This software is under development so consult Richard Myers (Durham) if problems are encountered. The software uses the GIST graphics package.

The Diagnostics page of TopGui can launch live traces and power spectra of many of the system variables, and can be used in order to examine the effects of injecting a sine wave, for example.