

KOSMOS ICD 6.1 Instrument Controller to Data Handling System

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 describes the interface between the instrument controller and the data handling system. Primarily, this identifies the various levels of interactions between the MONSOON Torrent, DHS and NOCS software.

Contents

1. Document Revision History	3
2. Purpose And Scope Of Document	3
3. WBS Element(s) and NOAO Account Code(s)	3
4. System Architecture	3
4.1. Message Sequence Chart	5
4.2. Hardware Configuration	6
5. Interface Description	6
5.1. Internal Command(s)	6
init	6
help	6
test	6
5.2. Meta-Data Command(s)	6
newobs	6
endobs	6
5.3. The GWC Interface	6
5.4. Application Programming Interface	8
dhsSysOpen(long *,char *,long *,ulong)	8
dhsSysClose(long *,char *,long)	9
dhsOpenConnect(long *,char *,long *,ulong,fpConfig_t *)	9
dhsCloseConnect(long *,char *,long)	9
dhsOpenExp(long *,char *,long,fpConfig_t *,double *,char *)	10
dhsCloseExp(long *,char *,long,double)	10
dhsSendMetaData(long *,char *,long,void *,size_t,mdConfig_t *,double *,char *)	10
dhsSendPixelData(long *,char *,long,void *,size_t,fpConfig_t *,double *,char *)	11
dhsReadImage(long *,char *,long)	11
dhsIOCtl(long *,char *,long,ulong,double *,char *,...)	11
5.5. Tcl Interface	12
dhs::help	12

dhs::version	12
dhs::SysOpen	12
dhs::SysClose	12
dhs::OpenConnect	12
dhs::CloseConnect	12
dhs::OpenExp	12
dhs::CloseExp	12
dhs::IOctl	12
dhs::ReadImage	12
dhs::PixelData	12
dhs::MetaData	12
5.6. Example(s)	12
testnohs	12
Tcl Example(s)	12
A Observation Header(s) System GWC Interface	15

1. Document Revision History

6 July 2010: original version, (pnd@noao.edu).

2. Purpose And Scope Of Document

This document identifies the interface between the instrument controller and the data handling system. It is intended for software and/or hardware engineers on either side of this divide to clearly identify their rôle within the overall system architecture and to implement any sub-system to achieve the desired objective.

This document is under (manual) revision control. All corrections, additions or comments should be addressed to the principal author.

3. WBS Element(s) and NOAO Account Code(s)

KOSMOS WBS v1.9^[1] shows this interface to be described by element D.N.9.3 (Data Management Interface) described therein as “design and coding of software for storing data according to observatory requirements”. The appropriate NOAO account code is N-MR210-D93. This is a loose definition so we concentrate here on acquiring the data and producing a fully-assembled FITS image on disk for ready ingestion into a data analysis pipeline or archive front-end.

4. System Architecture

The original implementation of the NOCS¹ is described elsewhere^[2]. The KOSMOS implementation of this architecture is shown in the cartoon of Figure 1 on page 4. The main elements of the NOCS are:

NGUI defines the observation scripting engine and is the *only* interface into the system. The motivating document behind NGUI is the *Operating Concepts Definition Document*^[3].

NICS (NOCS Instrument Control System) defines the interface to the instrument, specifically the hardware elements. The motivating documents behind NICS are ICD 3.1^[7] and ICD 3.1^[8].

NSML (NOCS Monsoon Supervisor Layer) defines the interface to the detector system. The motivating document behind NSML is ICD 4.1 and describes the detailed interface of an implementation of the generic pixel server dictionary^[4].

NTCS (NOCS Telescope Control System) defines the interface to the telescope control system. The motivating document behind NTCS is the ICD 5.1^[6] (this document).

NOHS (NOCS Observation Header System) defines the interface to the environmental and observational meta-data which is published to the DHS. The motivating document behind NOHS is the ICD 6.1^[5].

The guiding principles behind the NOCS design are (were):

- *Simplicity*. The amount of coding required is the minimum necessary to achieve the science objective;

¹This section is common to several ICDs so that each ICD can be read as a standalone document.

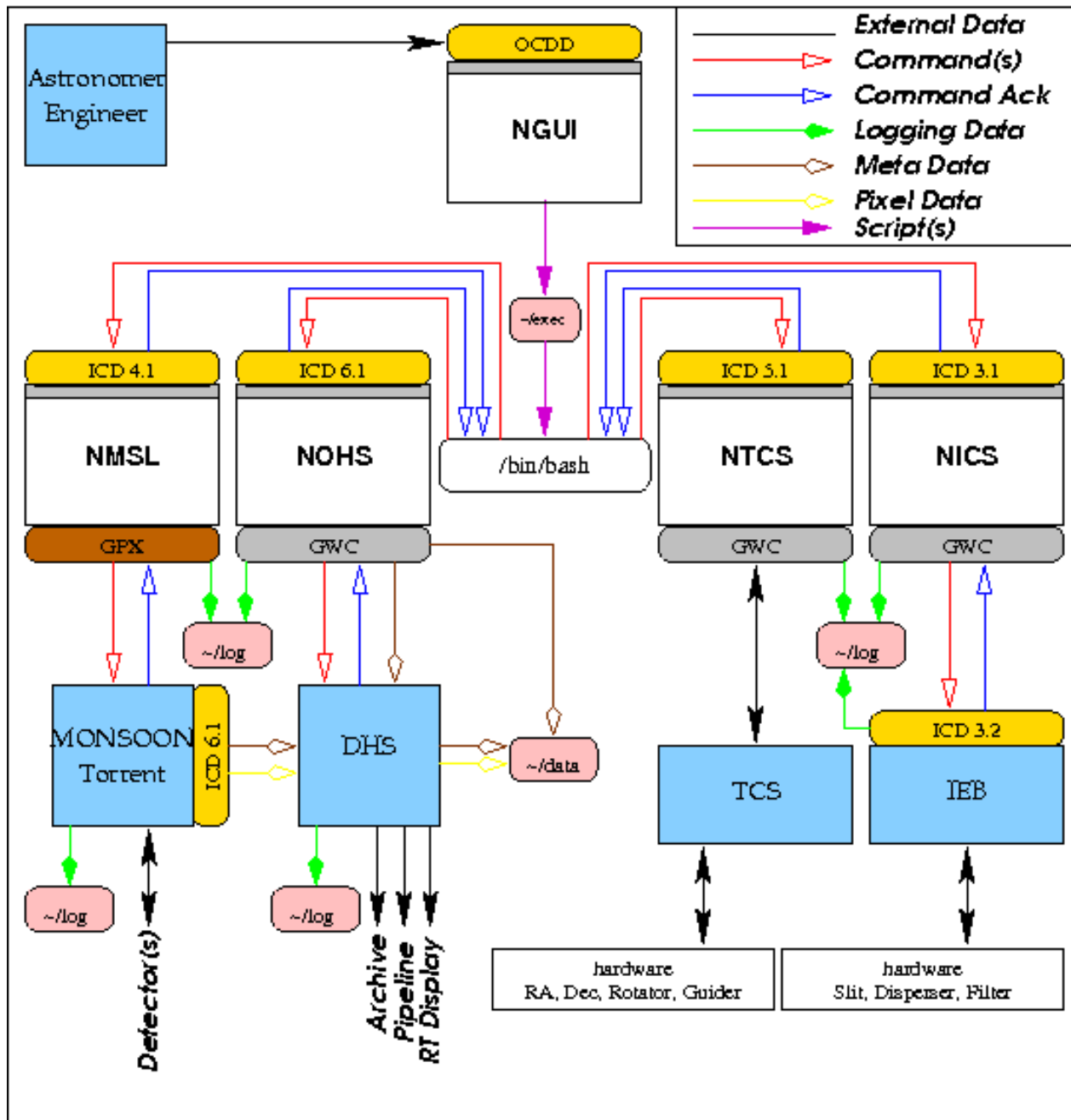


Figure 1.: KOSMOS Implementation of the NOCS Architecture

- *Modularity*. Only those parts of the system required at any specific instance need be started;
- *Supportability*. The system is written primarily in Tcl/tk which is widely used at the Mayall 4m;
- *Separability*. The system has clearly defined interfaces;
- *Repeatability*. Any script should be re-usable;
- *Consistency*. The science scripts ensure data is taken in a consistent manner with the minimum of human intervention and present ‘well known’ data products form ingestion downstream of the acquisition (*e.g.*, data analysis pipelines and/or data archives).

Further, various levels of simulation exist within the system so that laboratory testing can proceed without external hardware (or software) being present.

4.1. Message Sequence Chart

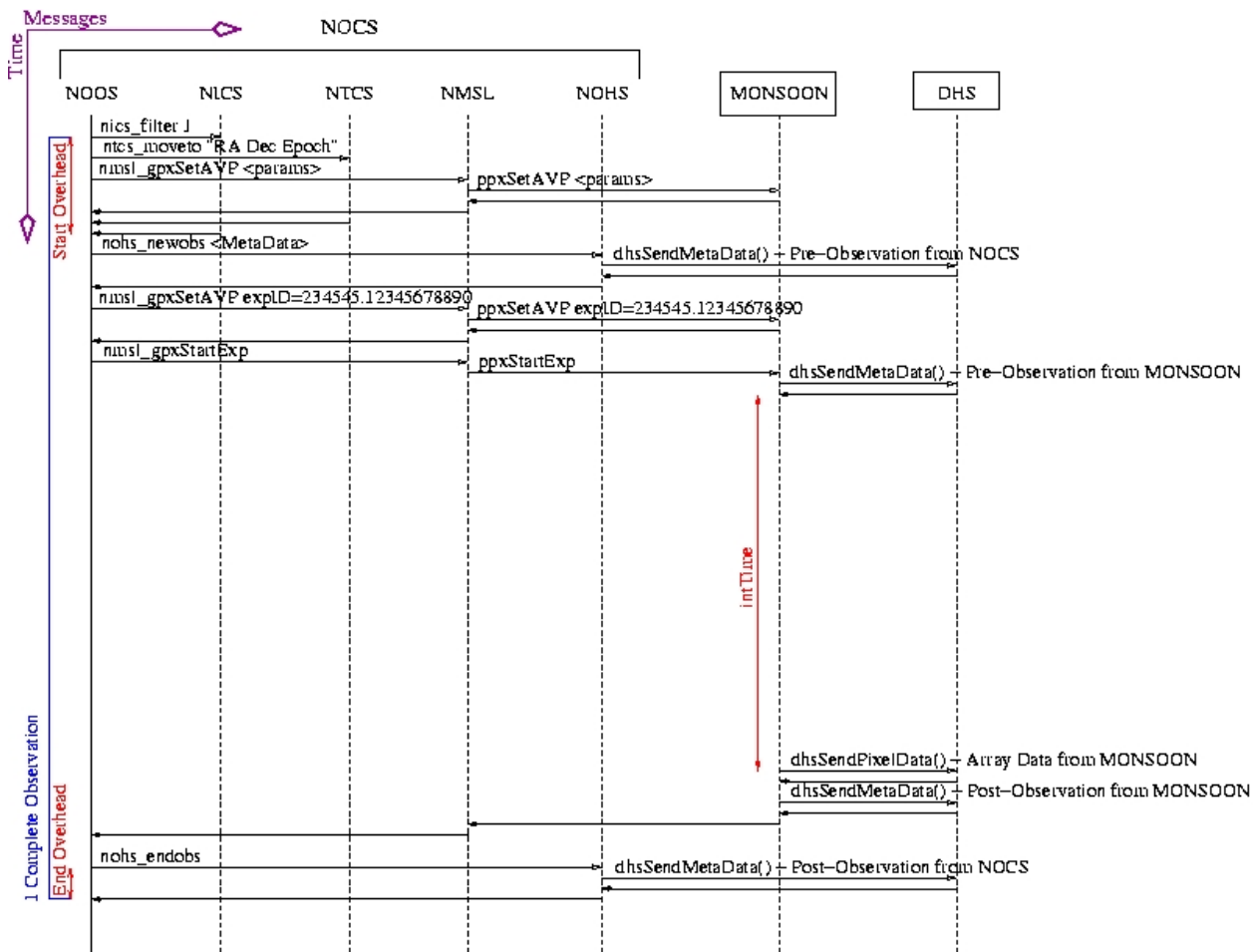


Figure 2.: NOCS Message Sequence Chart

The fundamental unit of the *NOAO Observation Control System* is, by definition, the observation. This unit is mapped to the message sequence chart of Figure 2 on page 5. In short, it is the job of NGUI to re-produce a science-oriented recipe as an executable script and, essentially, create a series of linked observations to achieve the given science objective. Although the GUIs are

written in Tcl/tk, the underlying re-scheduling mechanism is DRAMA^[9]. Thus, a script is a series of *ditscmd* directives to the other tasks in the system. The GUIs do the work ‘behind the scenes’ and return a simple SUCCESS or FAILURE status on completion of the action. If successful, the script continues. If not, the script aborts with an indication as to where in the script is failed and the task executing at that moment.

It should be heuristically obvious from the above, that the NOCS employs a demarcation paradigm inasmuch as NGUI will create a script but will *not* run it. Executing a script is a manual operation and clearly separates the creation of the script from its execution. A favourable side-effect is that NGUI is *not* required at the telescope and can be delivered to the astronomer prior to his/her observing run for self-paced familiarisation.

4.2. Hardware Configuration

Figure 3 on page 7 shows a typical hardware and network configuration of a NOCS implementation. Specifically, the PAN talks to the DHE (and hence the focal plane array) over an SL100 fiber interface but transfers the data to the primary DHS machine via the private network. Inside the same rack will be a KVM switch (TK-802R) and an Ethernet power controller (ECR2 DLI).

5. Interface Description

The KOSMOS instrument controller to data handling system interface has enough functionality to bring up the system in a known, safe, state ready to take data. The following commands are currently supported and there are no additions envisaged.

5.1. Internal Command(s)

init This command re-initializes the interface and returns the system interface to a ready state:
`ditscmd nohs nohs_init`

help This command returns simple help text on the functionality contained within the interface:
`ditscmd nohs nohs_help`

test This command performs both the *init* and *help* actions to test the underlying communications path:
`ditscmd nohs nohs_test`

5.2. Meta-Data Command(s)

The following set of meta-data commands are available. However, such commands are usually ‘hidden’ either behind a GUI-button or within a script and are rarely executed manually.

newobs This command sends a *newobs* directive to the DHS:
`ditscmd nohs nohs_newobs Argument1="metadata"`
 where *metadata* is a string of NOCS-specific meta-data automatically generated within scripts.

endobs This command sends an *endobs* directive to the DHS:
`ditscmd nohs nohs_endobs`

5.3. The GWC Interface

The NOHS is a GWC-aware^[10] client and subscribes to telemetry streams to deliver during a *newobs* directive, environmental and instrument-specific meta-data to the DHS. Since the NOHS

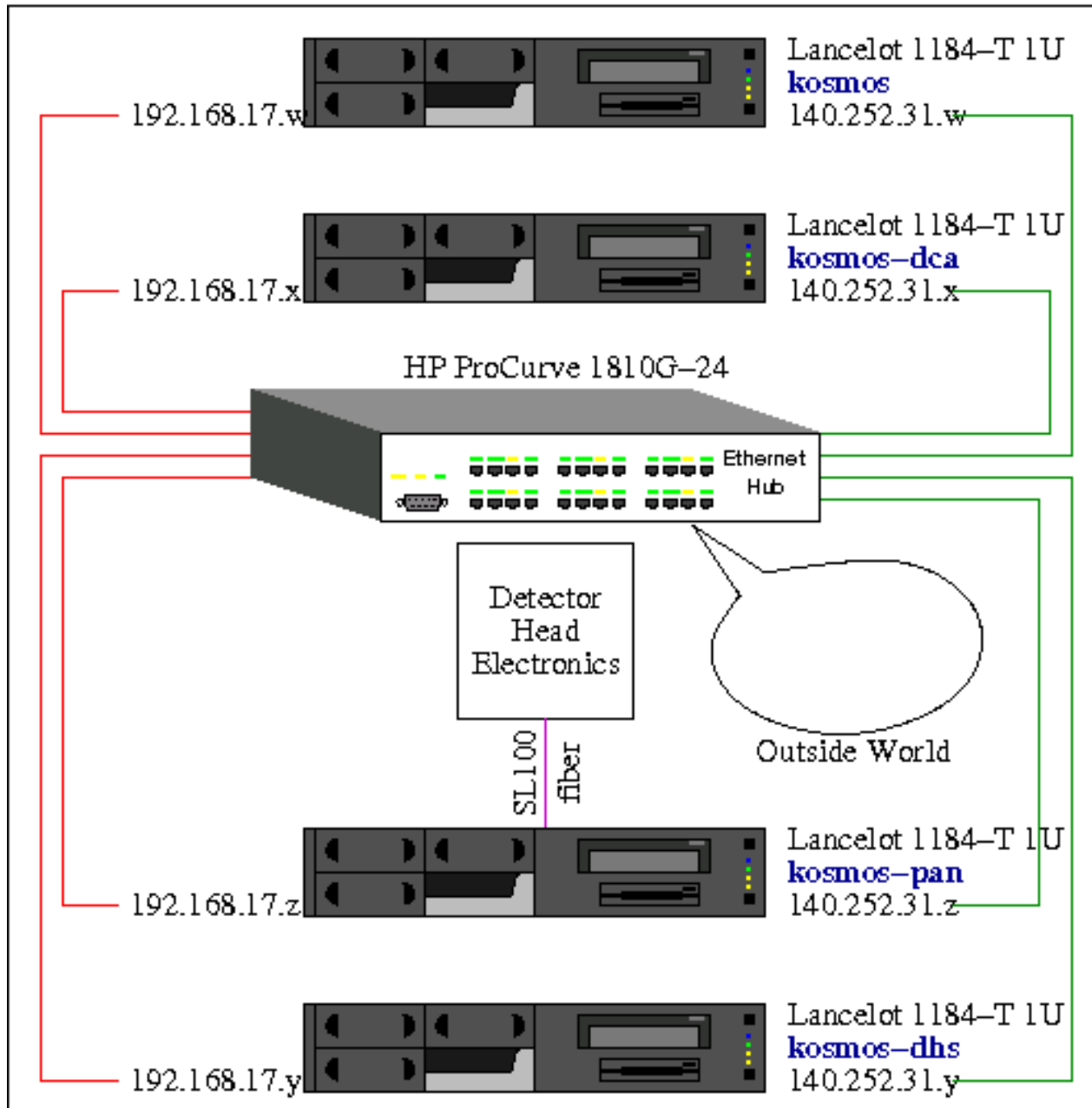


Figure 3.: Typical NOCS Hardware and Network Implementation

is entirely *passive*, it is controlled by a well-known configuration file an example of which is reproduced in appendix A. Note that the last 15 items in appendix A are indicative of KOSMOS-style meta-data items but will, in fact, be dictated by the contents of ICD 3.1^[7].

5.4. Application Programming Interface

The following structures are defined:

```
typedef struct fpConfig_str { /* focal plane configuration */
    ulong xSize; /* row size in focal plane (# of cols) */
    ulong ySize; /* col size in focal plane (# of rows) */
    ulong xStart; /* column index of 1st pixel in plane */
    ulong yStart; /* row index of 1st pixel in plane */
    long dataType; /* data type of the pixels */
} fpConfig_t, *fpConfig_p, **fpConfig_s;

typedef struct mdConfig_str { /* meta data configuration */
    ulong metaType; /* conceptual type meta data */
    ulong numFields; /* number of fields array */
    ulong fieldSize[DHS_MAXFIELDS]; /* number of items in field */
    ulong dataType[DHS_MAXFIELDS]; /* data type of the values */
} mdConfig_t, *mdConfig_p, **mdConfig_s;
```

All functions within the API return void. The library implements the concept of inherited status (*long *I*) and the function returns straight away if the input status is bad. If the input status is good, the function returns its exit status via the same variable. In either case a “reason” may be returned in the *char *R* argument.

The following systems are identified:

```
#define DHS_IAMOCS 0xFACE /* ID for observation control system */
#define DHS_IAMMSL 0xCAFE /* ID for monsoon supervisor layer */
#define DHS_IAMPAN 0xABCD /* ID for pixel acquisition node */
#define DHS_IAMWHO 0xFEED /* ID yet to be established */
```

The following IOCTL calls are defined:

```
#define DHS_IOC_OBSCONFIG 0x0100 /* define an obsetID set */
#define DHS_IOC_MDCONFIG 0x0200 /* define mdConfig set */
#define DHS_IOC_FPCONFIG 0x0400 /* define fpConfig set */
#define DHS_IOC_KEYWORD_TRANS 0x0800 /* define keyword translation */
#define DHS_IOC_DEBUG_LVL 0x1000 /* define debug level */
#define DHS_IOC_SIMULATION 0x2000 /* define simulation level */
#define DHS_IOC_SETFILENAME 0x4000 /* set file name */
```

*dhsSysOpen(long *,char *,long *,ulong)*

void dhsSysOpen(long *I,char *R,long *F,ulong W);

*long *I* an inherited status value, returns without action if input is bad otherwise returns exit status of function;

*char *R* a “reason string” propagated when the function status is an error condition;

*long *F* a returned file descriptor for future communications with the DHS;

ulong W an identifier to establish the caller as either a PAN or the NOCS;

Notes This is the supervisor level open command and returns a suitable file descriptor to the calling routine.

*dhsSysClose(long *,char *,long)*

void dhsSysClose(long *I,char *R,long F);

long *I an inherited status value, returns without action if input is bad otherwise returns exit status of function;

char *R a “reason string” propagated when the function status is an error condition;

long F an already open file descriptor for closing;

Notes This is the supervisor level close command and accepts an open file descriptor for closure.

*dhsOpenConnect(long *,char *,long *,ulong,fpConfig_t *)*

void dhsOpenConnect(long *I,char *R,long *F,ulong W,fpConfig_t *C);

long *I an inherited status value, returns without action if input is bad otherwise returns exit status of function;

char *R a “reason string” propagated when the function status is an error condition;

long *F a returned file descriptor for future communications with the DHS;

ulong W an identifier to establish the caller as either a PAN or the NOCS;

fpConfig_t *C an input focal plane configuration structure;

Notes This is the system level open command and returns a suitable file descriptor to the calling routine.

*dhsCloseConnect(long *,char *,long)*

void dhsCloseConnect(long *I,char *R,long F);

long *I an inherited status value, returns without action if input is bad otherwise returns exit status of function;

char *R a “reason string” propagated when the function status is an error condition;

long F an already open file descriptor for closing;

Notes This is the system level close command and checks the input file descriptor for validity before closing.

*dhsOpenExp(long *,char *,long,fpConfig_t *,double *,char *)*

void dhsOpenExp(long *I,char *R,long F,fpConfig_t *C,double *E,char *O);

long *I an inherited status value, returns without action if input is bad otherwise returns exit status of function;

char *R a “reason string” propagated when the function status is an error condition;

long F an already open file descriptor;

fpConfig_t *C an input focal plane configuration structure;

double *E a unique exposure identifier for this data;

char *O a unique observation identifier for this data;

Notes This is the exposure level open command and checks the input file descriptor for validity and performs housekeeping functions.

*dhsCloseExp(long *,char *,long,double)*

void dhsCloseExp(long *I,char *R,long F,double E);

long *I an inherited status value, returns without action if input is bad otherwise returns exit status of function;

char *R a “reason string” propagated when the function status is an error condition;

long F an already open file descriptor;

double E a unique exposure identifier for this data;

Notes This is the exposure level close command and checks the input file descriptor for validity and performs housekeeping functions.

*dhsSendMetaData(long *,char *,long,void *,size_t,mdConfig_t *,double *,char *)*

void dhsSendMetaData(long *I,char *R,long F,void *A,size_t S,mdConfig_t *C,double *E,char *O);

long *I an inherited status value, returns without action if input is bad otherwise returns exit status of function;

char *R a “reason string” propagated when the function status is an error condition;

long F an already open file descriptor;

void *A the (starting) memory address of the meta-data block;

size_t S the total size of the meta-data block;

mdConfig_t *C an input meta-data configuration structure identifying the type and layout of the meta-data block;

double *E a unique exposure identifier for this data;

char *O a unique observation identifier for this data;

Notes This is the delivery mechanism to the DHS of block of meta-data in a well-known format.

*dhsSendPixelData(long *,char *,long,void *,size_t,fpConfig_t *,double *,char *)*

void dhsSendPixelData(long *I,char *R,long F,void *A,size_t S,fpConfig_t *C,double *E,char *O);

long *I an inherited status value, returns without action if input is bad otherwise returns exit status of function;

char *R a “reason string” propagated when the function status is an error condition;

long F an already open file descriptor;

void *A the (starting) memory address of the pixel-data block;

size_t S the total size of the pixel-data block;

fpConfig_t *C an input pixel-data (focal plane) configuration structure identifying the type and layout of the pixel-data block;

double *E a unique exposure identifier for this data;

char *O a unique observation identifier for this data;

Notes This is the delivery mechanism to the DHS of block of pixel-data in a well-known format.

*dhsReadImage(long *,char *,long)*

void dhsReadImage(long *I,char *R,long F);

long *I an inherited status value, returns without action if input is bad otherwise returns exit status of function;

char *R a “reason string” propagated when the function status is an error condition;

long F an already open file descriptor;

Notes This read an image from a file descriptor but is now obsolete.

*dhsIOctl(long *,char *,long,ulong,double *,char *,...)*

void dhsIOctl(long *I,char *R,long F,ulong X,double *E,char *O,...);

long *I an inherited status value, returns without action if input is bad otherwise returns exit status of function;

char *R a “reason string” propagated when the function status is an error condition;

long F an already open file descriptor;

ulong X the desired ioctl function;

double *E a unique exposure identifier for this data;

char *O a unique observation identifier for this data;

... depending on the ioctl function, variable arguments are supplied here;

Notes This is the delivery mechanism to the DHS of block of pixel-data in a well-known format.

5.5. Tcl Interface

dhs::help This internal command returns a list of available commands.

dhs::version This internal command returns the version number of the library.

dhs::SysOpen This is a wrapper to the API function described by example in section 5.6.

dhs::SysClose This is a wrapper to the API function described by example in section 5.6.

dhs::OpenConnect This is a wrapper to the API function described by example in section 5.6.

dhs::CloseConnect This is a wrapper to the API function described by example in section 5.6.

dhs::OpenExp This is a wrapper to the API function described by example in section 5.6.

dhs::CloseExp This is a wrapper to the API function described by example in section 5.6.

dhs::IOctl This is a wrapper to the API function described by example in section 5.6.

dhs::ReadImage This is a wrapper to the API function described by example in section 5.6.

dhs::PixelData This is a wrapper to the API function described by example in section 5.6.

dhs::MetaData This is a wrapper to the API function described by example in section 5.6.

5.6. Example(s)

testnohs This is a wrapper that allows the end-user to test the system without creating a specific script. The simplest invocation is:

```
% testnohs
```

Tcl Example(s)

```
#!/bin/sh
# the next line restarts using wish \
exec `which wish` "$0" "$@"

load /usr/local/lib/libfitstcl.so
load /usr/local/lib/libdhsTcl.so
load /usr/local/lib/libmsdTcl.so

# define some data
set exID [msd::msd]
set obID "Observation Set #1"
set nelms 100
set nlines 5
set nitems 4
set array [list {0 1 2 3 4 5 6 7 8 9} {0 1 2 3 4 5 6 7 8 9} \
                {0 1 2 3 4 5 6 7 8 9} {0 1 2 3 4 5 6 7 8 9} \
                {9 8 7 6 5 4 3 2 1 0} {9 8 7 6 5 4 3 2 1 0} \
                {9 8 7 6 5 4 3 2 1 0} {9 8 7 6 5 4 3 2 1 0} \
                {9 8 7 6 5 4 3 2 1 0} {9 8 7 6 5 4 3 2 1 0}]
set fits [list TELESCOP "KPNO Mayall 4m" "Kitt Peak National Observatory" \
```

```
    ASTRONOM "Joe Bloggs" "MONSOON Test System Username" \  
    INSTRUME "MONSOON+KOSMOS" "MONSOON Image Acquisition" \  
    DETECTOR "aladdin_III" "aladdin_III 4096x4096" ]  
set avp [list intTime 1.09365 "Integration time in milliseconds" \  
    biasVoltage -5.0982521 "Bias voltage for array" \  
    dataDir /MNSN/soft_dev/data/ "Current data directory" \  
    fileName ${exID}.fits "Current output filename" \  
    creg 0x02000000 "Control register in HEX representation" ]  
  
# show version / help  
dhs::version  
dhs::help  
  
# SysOpen (0xFACE=OCS, 0xCAFE=MSL)  
set sFD [dhs::SysOpen 0xFACE]  
puts "dhs::SysOpen (sFD=${sFD}) [expr {$sFD >= 0 ? "OK" : "FAILED"}]"  
  
# OpenConnect (0xABCD=PAN 0xFEED=WHO)  
set cFD [dhs::OpenConnect 0xFEED {10 10 0 0 8 1 1 1 1 1 1}]  
puts "dhs::OpenConnect (cFD=${cFD}) [expr {$cFD >= 0 ? "OK" : "FAILED"}]"  
  
# OpenExp  
set eFD [dhs::OpenExp $cFD {10 10 0 0 8 1 1 1 1 1 1} $exID $obID]  
puts "dhs::OpenExp (eFD=${eFD}) [expr {$eFD >= 0 ? "OK" : "FAILED"}]"  
  
# IOctl (0x1000=debug)  
set sOK [dhs::IOctl $eFD 0x1000 $exID $obID]  
puts "dhs::IOctl (eFD=${eFD}) [expr {$sOK == 0 ? "OK" : "FAILED"}]"  
  
# PixelData  
set sOK [dhs::PixelData $eFD $array $nelms {10 10 0 0 8 1 1 1 1 1 1} $exID $obID]  
puts "dhs::PixelData (eFD=${eFD}) [expr {$sOK == 0 ? "OK" : "FAILED"}]"  
  
# MetaData (FITS)  
set sOK [dhs::MetaData $eFD $fits $nitems {1 3 {8 20 46} {1 1 1}} $exID $obID]  
puts "dhs::MetaData (FITS) (eFD=${eFD}) [expr {$sOK == 0 ? "OK" : "FAILED"}]"  
  
# MetaData (AVP)  
set sOK [dhs::MetaData $eFD $avp $nlines {2 3 {32 32 64} {1 1 1}} $exID $obID]  
puts "dhs::MetaData (AVP) (eFD=${eFD}) [expr {$sOK == 0 ? "OK" : "FAILED"}]"  
  
# CloseExp  
set sOK [dhs::CloseExp $eFD $exID]  
puts "dhs::CloseExp (eFD=${eFD}) [expr {$sOK == 0 ? "OK" : "FAILED"}]"  
  
# CloseConnect  
set sOK [dhs::CloseConnect $cFD]  
puts "dhs::CloseConnect (cFD=${cFD}) [expr {$sOK == 0 ? "OK" : "FAILED"}]"  
  
# SysClose  
set sOK [dhs::SysClose $sFD]  
puts "dhs::SysClose (sFD=${sFD}) [expr {$sOK == 0 ? "OK" : "FAILED"}]"
```

Acknowledgments. PND thanks Jay Elias and Mark Trueblood for corrections.

References

1. Elias, J., 2009, 'KOSMOS Work Breakdown Structuer Summary', v1.9, Kosmos Project, NOAO.
2. Daly, P. N. et al., 2008, 'The NEWFIRM Observing Software: From Design To Implementation', in *Advanced Software and Control for Astronomy II*, edited by Alan Bridger and Nicole M. Radziwill, Proc. SPIE, Volume 7019, pp701913-1 to 701913-15.
3. Elias, J., 2009, 'Operating Concepts Definition Document', v1.9, Kosmos Project, NOAO.
4. Buchholz, N. and Daly, P. N., 2004, 'The Generic Pixel Server Dictionary', in *Advanced Software, Control and Communications Systems for Astronomy*, edited by Hilton Lewis and Gianni Raffi, Proc. SPIE, Vol. 5496, pp167-177.
5. Daly, P. N., 2010, ICD 6.1, Kosmos Project, NOAO.
6. Daly, P. N., 2010, ICD 5.1, Kosmos Project, NOAO.
7. Daly, P. N., 2010, ICD 3.1, Kosmos Project, NOAO.
8. Pogge, R. et al., 2010, ICD 3.2, Kosmos Project, NOAO.
9. Farrell, T. J., Bailey, J. A. and Shortridge, K., 1995, 'Tcl/tk with DRAMA—A Natural for Building User Interfaces to Instrumentation Systems?', in *Astronomical Data Analysis Software and Systems IV*, ASP Conf. Ser., Vol. 77, edited by R. A. Shaw, H. E. Payne and J. J. E. Hayes.
10. Gillies, K., 1996, 'Programmers Guide to the Generic WIYN Client (GWC)', Mountain Programming Group, NOAO.

A Observation Header(s) System GWC Interface

```

##
# nohs.gwc
# - 200100706: original version - Phil Daly (pnd@noao.edu)
#-
##
#KEYWORD\STREAM                                \COMMENT                                \DUMMY VALUE
#-
#0000000\000000000011111111222222222222\0000000000111111112222222222333333333344444444445555555556\00000000001111111112
#2345678\0123456789012345678901234567890\012345678901234567890123456789012345678901234567890123456789012345678901234567890
AIRMASS \tcs.telescope.airmass \Airmass \0 [S]
ALT \tcs.telescope.alt \Telescope altitude \00:00:00 [S]
AZ \tcs.telescope.az \Telescope azimuth \00:00:00 [S]
DEC \tcs.target.dec \Declination \00:00:00.0 [S]
DECDIFF \tcs.telescope.dec_diff \[arcsec] Dec diff \0 [S]
DECINDEX \tcs.telescope.dec_index \[arcsec] Dec index \0 [S]
DECINST \tcs.telescope.dec_inst_center \[arcsec] Dec instrument center \0 [S]
DECOFF \tcs.telescope.dec_offset \[arcsec] Dec offset \0 [S]
DECZERO \tcs.telescope.dec_zero \[arcsec] Dec zero \0 [S]
DOMEAZ \tcs.dome.az \[deg] Dome position \0 [S]
DOMEERR \tcs.dome.error \[deg] Dome error as distance from target \0 [S]
EPOCH \tcs.target.epoch \[yr] epoch \2007 [S]
EQUINOX \tcs.telescope.equinox \[yr] Equinox \2007 [S]
RA \tcs.target.ra \Right Ascension \00:00:00.00 [S]
RADIFF \tcs.telescope.ra_diff \[arcsec] RA diff \0 [S]
RAINDEX \tcs.telescope.ra_index \[arcsec] RA index \0 [S]
RAINST \tcs.telescope.ra_inst_center \[arcsec] RA instrument center \0 [S]
RAOFF \tcs.telescope.ra_offset \[arcsec] RA offset \0 [S]
RAZERO \tcs.telescope.ra_zero \[arcsec] RA zero \0 [S]
ST \tcs.time.st \Sidereal time \00:00:00 [S]
TCPGRD \tcs.guider.mode \Guider status (on/off/lock) \off [S]
TCPHA \tcs.telescope.ha \Telescope hour angle \0 [S]
UT \tcs.time.ut \Universal time \00:00:00 [S]
UTDATE \tcs.time.utdate \UT date \01/01/2007 [S]
ZD \tcs.telescope.zenithdist \[deg] Zenith distance \0 [S]
ACTPRIM \tcs.telescope.active_primary \primary mirror mode \off [E]
ALRTDEC \tcs.alert.dec \Dec alert \off [E]
ALRTHA \tcs.alert.ha \HA alert \off [E]
ALRTSERV \tcs.alert.servo \Servo alert \off [E]
ALRTSTOW \tcs.alert.stowed \Stow alert \off [E]
DECPRE \tcs.target.dec_preset \declination preset \00:00:00.0 [E]
DECSERVO \tcs.telescope.decservo \[deg] dec servo position \0 [E]
DOMEODE \tcs.dome.mode \Dome mode \off [E]
DOMESTAT \tcs.dome.status \Dome status \ready [E]
EPOCHPRE \tcs.target.epoch_preset \[yr] Epoch preset \2007 [E]
FIELD \tcs.telescope.field \Telescope field \0 [E]
FOCI \tcs.telescope.fratio \Telescope foci \0 [E]
FOCUS \tcs.telescope.focus \[mm] Telescope focus \0 [E]
HASERVO \tcs.telescope.haservo \[deg] HA servo position \0 [E]
INSTR \tcs.instrument.name \Instrument used to obtain these data \mosaic [E]
PARALL \tcs.telescope.parallactic \[deg] parallactic angle \0 [E]
RAPRE \tcs.target.ra_preset \right ascension preset (HH:MM:SS.SS) \00:00:00.00 [E]
TCPMODE \tcs.telescope.mode \telescope mode \ready [E]
TCPBLEW \tcs.telescope.slew \telescope slew status \off [E]
TCPTRACK \tcs.telescope.tracking \telescope tracking status \off [E]
TCSCOMPU \tcs.main.computer \name of tcs4m computer \onion [E]
TCSHOST \tcs.main.host \host for tcs4m \onion [E]
TCSLINK \tcs.main.link \TCS link (up/down) \down [E]
TCSOFST \tcs.telescope.offset \RA, Dec offset (arcsec) \0 0 [E]
TCSPID \tcs.main.pid \process id of tcs4m \101 [E]
TCSPOS \tcs.telescope.position \RA, Dec and epoch of target position \00:00:00.00 00:00:00.0 2007 [E]
TCSUPDAT \tcs.main.updates \tcs4m polling enable (on/off) \off [E]
PMTCTCFR \pmtc.thermocouple.cfr \[celsius] Reference temp. of cold point \0.0 [E]
PMTCTCSB \pmtc.thermocouple.cib \[celsius] Chimney south inside: bottom \0.0 [E]
PMTCTCSM \pmtc.thermocouple.cim \[celsius] Chimney south inside: middle \0.0 [E]
PMTCTCST \pmtc.thermocouple.cit \[celsius] Chimney south inside: top \0.0 [E]
PMTCTCTS \pmtc.thermocouple.cos \[celsius] Chimney outside top: south \0.0 [E]
PMTCTCTW \pmtc.thermocouple.cow \[celsius] Chimney outside top: west \0.0 [E]
PMTCTCEB \pmtc.thermocouple.eib \[celsius] Mirror inner bottom edge: east \0.0 [E]
PMTCTEIT \pmtc.thermocouple.eit \[celsius] Mirror inner top edge: east \0.0 [E]
PMTCTEOB \pmtc.thermocouple.eob \[celsius] Mirror outer bottom edge: east \0.0 [E]
PMTCTEOT \pmtc.thermocouple.eot \[celsius] Mirror outer top edge: east \0.0 [E]
PMTCTHXA \pmtc.thermocouple.hxa \[celsius] Heat exchanger ambient air: out \0.0 [E]
PMTCTHXD \pmtc.thermocouple.hxd \[celsius] Heat exchanger ambient air: in \0.0 [E]
PMTCTHXI \pmtc.thermocouple.hxi \[celsius] Heat exchanger glycol: in \0.0 [E]
PMTCTHXO \pmtc.thermocouple.hxo \[celsius] Heat exchanger glycol: out \0.0 [E]
PMTCTMHS \pmtc.thermocouple.mhs \[celsius] Mirror cover hinge: south \0.0 [E]
PMTCTMHW \pmtc.thermocouple.mhw \[celsius] Mirror cover hinge: west \0.0 [E]
PMTCTNIB \pmtc.thermocouple.nib \[celsius] Mirror inner bottom edge: north \0.0 [E]
PMTCTNIT \pmtc.thermocouple.nit \[celsius] Mirror inner top edge: north \0.0 [E]
PMTCTNOB \pmtc.thermocouple.nob \[celsius] Mirror outer bottom edge: north \0.0 [E]
PMTCTNOT \pmtc.thermocouple.not \[celsius] Mirror outer top edge: north \0.0 [E]
PMTCTSIB \pmtc.thermocouple.sib \[celsius] Mirror inner bottom edge: south \0.0 [E]
PMTCTSIT \pmtc.thermocouple.sit \[celsius] Mirror inner top edge: south \0.0 [E]
PMTCTSOB \pmtc.thermocouple.sob \[celsius] Mirror outer bottom edge: south \0.0 [E]
PMTCTSOT \pmtc.thermocouple.sot \[celsius] Mirror outer top edge: south \0.0 [E]
PMTCTSTB \pmtc.thermocouple.tsb \[celsius] Truss south: bottom \0.0 [E]
PMTCTSTM \pmtc.thermocouple.tsm \[celsius] Truss south: middle \0.0 [E]
PMTCTSTT \pmtc.thermocouple.tst \[celsius] Truss south: top \0.0 [E]
PMTCTWIB \pmtc.thermocouple.wib \[celsius] Mirror inner bottom edge: west \0.0 [E]
PMTCTWIT \pmtc.thermocouple.wit \[celsius] Mirror inner top edge: west \0.0 [E]
PMTCTWOB \pmtc.thermocouple.wob \[celsius] Mirror outer bottom edge: west \0.0 [E]
PMTCTWOT \pmtc.thermocouple.wot \[celsius] Mirror outer top edge: west \0.0 [E]
KSIDENT \kosmos.main.ident \KOSMOS control program id and version \unknown [E]
KSDPOS \kosmos.disperser.pos \KOSMOS disperser demand (lo|med|high|narrow|other|none) \unknown [S]
KSDNAME \kosmos.disperser.pos \KOSMOS disperser name (lo|med|high|narrow|other|none) \unknown [S]
KSFILTER \kosmos.filter.name \KOSMOS filter name (U|B|V|R|I|open|between) \unknown [S]
KSFILCMD \kosmos.filter.pos \KOSMOS filter demand (U|B|V|R|I|open|between) \unknown [S]
KSFWIPOS \kosmos.filter.fw1 \KOSMOS filter wheel 1 number (1-6) \1 [S]
KSFW2POS \kosmos.filter.fw2 \KOSMOS filter wheel 1 number (1-6) \1 [S]
KSSPOS \kosmos.slit.pos \KOSMOS slit demand (long|lpx|2px|narrow|other|none) \unknown [S]
KSSNAME \kosmos.slit.name \KOSMOS slit name (long|lpx|2px|narrow|other|none) \unknown [S]
KSCFOC \kosmos.collimator.focus \[mm] KOSMOS collimator focus (value) \0 [S]
KSTEMP1 \kosmos.thermal.temp1 \[kelvin] KOSMOS temperature sensor 1 \0.0 [S]
KSTEMP2 \kosmos.thermal.temp2 \[kelvin] KOSMOS temperature sensor 2 \0.0 [S]

```

```
KSTEMP3 \kosmos.thermal.temp3      \[kelvin] KOSMOS temperature sensor 3      \0.0 [S]
KSTEMP4 \kosmos.thermal.temp4      \[kelvin] KOSMOS temperature sensor 4      \0.0 [S]
KSTEMP5 \kosmos.thermal.temp5      \[kelvin] KOSMOS temperature sensor 5      \0.0 [S]
KSTEMP6 \kosmos.thermal.temp6      \[kelvin] KOSMOS temperature sensor 6      \0.0 [S]
KSTEMP7 \kosmos.thermal.temp7      \[kelvin] KOSMOS temperature sensor 7      \0.0 [S]
KSTEMP8 \kosmos.thermal.temp8      \[kelvin] KOSMOS temperature sensor 8      \0.0 [S]
```