Accepted by ESO Technical Archive
06.11.08

EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral
Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

VERY LARGE TELESCOPE

INSTRUMENTATION DIVISION

New General detector Controller

Optical DCS - User Manual

Document Number: VLT-MAN-ESO-13660-4086

Document Issue: 4.0

Date of Issue: 30/10/2008

	Name	Date	Signature
Prepared by :	Claudio Cumani Andrea Balestra	30.10.2008	
Approved by :	Dietrich Baade	3.11.2008	Dietrich Baade
Released by :	Mark Casali	4/11/08	

CHANGE RECORD

ISSUE	DATE	SECTIONS AFFECTED	REASON/INITIATION DOCUMENTS/REMARKS
0.1	22-09-2006	All	First draft, basic information for optical prototype
1.0	31-07-2007	All	Document widely rewritten, on the base of the comments to the first draft and of the NGCOSW implementation
2.0	31-08-2007	1.1 2.1 3.1.1 3.1.1 3.2.3.2 4 6.1 6.1 7 9.1.1 9.1.2 9.1.2 9.1.4 9.1.5	Optical exposure loop description corrected ngcoctr module added Local-Hardware-Test renamed Local-Software-Test Operational mode macros are defined in ngco.h Configuration values updated Command interface table updated List of public online database attributes updated NOTE added SETUP can be issued also for paused exposure. Exposure types macros are defined in ngco.h Description of exposure status values improved Exposure status macros are defined in ngco.h NGCOSW creates exposure Id, if not passed at START Description of differences with FIERA corrected
3.0		1 2 2.1 2.4 2.5 3.1.1 3.2.1 3.2.1 3.2.1 3.2.3.2 3.2.3.2 3.2.3.2 4 4.1 7.1 7.1 (prev. 7.2) 13 14 15.2 16	Reference to DOORS obsolete Removed NGC base package modules Updated list of NGCOSW modules Online database environment generation added INS_ROOT population added Configuration Set updated “Software-Test” mode removed DET.CON.GUI follows IR definition DET.CON.XTERM added -xterm option added -kill option removed “Local-Software-Test mode” removed Command interface table updated Changes with respect to FIERA updated “Complete Setup” chapter removed (obsolete) List of changes updated rtd interface description updated Description of GUIs added Evaluation of T/P implementation added STANDBY and exposure status readout added

4.0		2.3 (prev 2.4) 2.3.1 2.3.2 2.3.3 2.3.4 2.3.5 2.4 2.5 3.2.3.2 16 17 (prev. 16)	Oldb generation split btw IWS and LLCU cases Online database environment generation added Online database environment generation added Online database verification added Online database startup added Detector user description added Logging configuration added INS_ROOT population description improved Example updated Manpage Chapter added Minor corrections

Table of contents

List of Figures

List of Tables

1. Introduction

The software described in this manual is intended to be used in the ESO VLT project by ESO and authorized external contractors only.

While every precaution has been taken in the development of the software and in the preparation of this documentation, ESO assumes no responsibility for errors or omissions, or for damage resulting from the use of the software or of the information contained herein.

1.1. Purpose

This document is the User Manual of the Next Generation detector Controller (NGC) Control Software for optical instruments (NGCOSW).

It is intended to provide people, who intend to use the NGC Controller for optical Instruments, with all the necessary information to **install** from scratch the NGCOSW, **interact programmatically** with the NGCOSW, operate an optical camera as a simple **standalone** instrument.

The manual assumes that the reader has some knowledge of C/C++ and Tcl/Tk languages, UNIX Operating System, VLT Software, in particular CCS. It is not intended to be an introduction to optical CCD cameras, and therefore it uses common terminology in this field (e.g. pixel, binning, readout, frame-transfer chip, etc.) without further explanation.

The control software for infrared applications (NGCIRSW) is described in a separate manual [RD77]. Basically the NGC electronics [AD8] is the same for both infrared and optical applications. Nevertheless there are many differences concerning the usage of the controller and the data acquisition and data handling procedures. To cover both applications in an effective way and also to have a certain backwards compatibility with the predecessors FIERA and IRACE, different SW-architectures have been chosen, which are described in detail in the NGC SW design documents [AD9], [AD10] and [AD11]. The following paragraph summarizes the main differences:

- **Detector Read-Out Schemes**

  For an infrared detector (CMOS / non destructive readout) the clock-pattern generation runs in an infinite loop and the detector is read-out/reset all the times. An optical detector (CCD / destructive readout) is read-out just once at the end of an exposure.

- **Data Handling**

  The optical application delivers one frame at the end of the exposure and the only processing to be done is pixel sorting, centroiding and possibly and offset correction (if not yet done in HW). The infrared data require some pre-processing depending on the read-out mode of the detector in use. The read-out modes, the pre-processing algorithms and the setup-parameters for these algorithms are manifold and require a very high degree of flexibility. The pre-processing task produces an arbitrary number of different result frame types, which all have to be transferred and/or displayed on demand. This also has an impact on the RTD-interface.

- **Exposure Loops**

  For infrared applications *starting an exposure* basically means starting to transfer the acquired data to a FITS-file (i.e. the server has to attach to and keep step with a running procedure). The end-of-exposure condition is flexible and depends on both the requested frame types and on the number of frames of each type to be produced and stored. The optical exposure always terminates with the saving of the data, which are read at the end of the exposure and follows a much more rigid scheme ("*inactive*" - "*wiping*" [- "*pending*"] - *integrating*" - "*reading*" - "*transferring*" - "*inactive*"). This scheme implies an active intervention of the control-server during the exposure like the application of new voltages in each state and the additional shutter-control, whereas the infrared control-server mainly reacts passively on incoming data-frames once the exposure is started. So basically the demands on process concurrency are very different in both cases.

A conscious effort has been made to maintain a certain degree of backwards compatibility of NGCOSW with FIERASW. Where applicable, a hint to the major changes with respect to FIERASW can be found at the end of each section.

1.2. Scope

Scope of this document is the NGC Control Software for optical instruments (NGCOSW).

1.3. Applicable Documents

Applicable documents used in the NGC project are listed in the document VLT-LIS-ESO-13660-3906 "NGC Project Documentation".

1.4. Reference Documents

Reference documents used in the NGC project are listed in the document VLT-LIS-ESO-13660-3906 "NGC Project Documentation".

1.5. Abbreviations and Acronyms

Abbreviations and acronyms used in the NGC project are listed in [RD64].

1.6. Glossary

All the relevant concepts used within the NGC project are listed in [RD63].

1.7. Stylistic Conventions

The following styles are used:

bold in the text, for commands, filenames, pre/suffixes as they have to be typed.

italic in the text, for parts that have to be substituted with the real content before typing.

`courier` for examples, commands, filenames as they have to be typed.

<name> in the examples, for parts that have to be substituted with the real content

The **bold** and *italic* styles are also used to highlight words.

1.8. Naming Conventions

This implementation follows the naming conventions as outlined in [AD27].

1.9. Problem Reporting/Change Request

The form described in [AD72] shall be used.

2. Installation and configuration

NGCOSW runs on ESO standard Instrument Workstations (IWS) and NGC-LLCUs[1].

The VLTSW (version 2008 or more recent) must have been already installed on the hosts.

2.1. Software Modules

All software modules are under CMM configuration control. Before installing the NGC optical detector control software package (***NGCOSW***), the NGC base software package must be installed (see [RD9] and [AD10]). The NGCOSW package consists of:

- ***ngcocon*** - The NGC system coordination module for optical applications. This includes all required scripts for system startup and shutdown.
- ***ncgoctr*** - The NGC exposure Control module for optical applications.
- ***ncgoexp*** - The NGC Exposure Coordination module for optical applications.
- ***ncgoits*** - The NGC Image Transfer Server module for optical applications.
- ***ncgoitc*** - The NGC Image Transfer Client module for optical applications.
- ***ncgotm*** - The NGC Telemetry module for optical applications.
- ***ngcoui*** - Engineering GUI used for direct system interaction and data acquisition.
- ***ngcoarc*** - Installation scripts for the overall NGCOSW software package.

2.2. Installation

2.2.1. Using installation scripts

Before installing the NGCOSW package via the installation scripts**, be sure that the NGC base software package has been already installed (see [RD9])**.

Installation scripts for the software package are provided in the *ngcoarc* software module and work on both the IWS and on the NGC-LLCU.

The procedure to create the package consists of the following steps:

1. Retrieve from the archive and install the module *ngcoarc*:

```
mkdir <NGCOROOT>
cd <NGCOROOT>
cmmCopy ngcoarc
```

2. Retrieve all needed modules from the archive and install them:

```
cd ngcoarc/src
make all install
```

[1] ESO Standard hardware is described in http://websqa.hq.eso.org/sdd/bin/view/SDDInfo/LinuxStandardHw

Note: unless a INTROOT is defined, all the NGCOSW code will be installed in the VLTROOT. Therefore these scripts must be run by a user with the appropriate read/write privileges.

2.2.2. Using pkgin

The ngcins software module contains a pkgin installation-configuration (**for both NGC IR and OPT software**):

```
cmmCopy ngcins
pkginBuild  ngcins
```

2.3. Online database environment

To automatically generate the online database, the environment variables RTAPENV, CCDLENV and CCDNAME must be defined:

- RTAPENV defines the name of the local online database environment
- CCDLENV on the IWS defines the name of the remote online database environment, on the NGC-LLCU it must be set to 0
- CCDNAME defines the name of the detector camera

NOTE : on the NGC-LLCU the environment variables are defined in the files

```
/etc/pecs/releases/000/etc/locality/apps-all.env
/etc/pecs/releases/000/etc/locality/apps-${HOST}.env
```

On the IWS you could define them in the same files or in

```
~/.pecs/apps-${HOST}.env
```

The script ngcoDcsOldb performs a preliminary system check: if the environment variables are defined, if the ACC server is defined and running, if local and remote environments are defined on the local computer and in the ACC server, if the scanning has been properly configured, if the user which shall run the software is defined on the local and the remote computer, etc. (manpage of ngcoDcsOldb is available in 16.1.1.)

The same script then handles the online database generation, acting in different ways on the IWS (see 2.3.1) and the LLCU (see 2.3.2).

2.3.1. IWS online database generation

On the IWS, the DATABASE.db.NGCOSW and USER.db.NGCOSW templates for the online database environment are installed in the $VLTDATA/ENVIRONMENTS/$RTAPENV/dbl directory by running:

```
ngcoDcsOldb -renv $CCDLENV -host IWS
```

(manpage of ngcoDcsOldb is available in 16.1.1.)

Use the DATABASE.db.NGCOSW and USER.db.NGCOSW templates to edit the DATABASE.db and USER.db files.

In the template `DATABASE.db.NGCOSW` it is described how to describe different configurations (instrument controlling only one camera, instrument controlling more cameras).

Examples:

- Instrument controlling one camera:

  Add in the `DATABASE.db` the following, replacing `<myCCDNAME>` with the name of the camera (e.g., `$CCDNAME`), `<:myPATH>` with the preferred oldb location (e.g., `:DCS:optical`):

```
#undef  CCDNAME
#undef  ngcdcsINSTANCE
#undef  NGCROOT

#define CCDNAME <myCCDNAME>
#define ngcdcsINSTANCE ngcdcs_<myCCDNAME>
#define NGCROOT :Appl_data<:myPATH>:CCDNAME
```

- Instrument controlling four cameras:

  Add in the `DATABASE.db` the following, replacing `<myINSTRUMENT>` with the instrument name, `<myCCDNAME>`, `<myCCDNAME2>`, `<myCCDNAME3>` , `<myCCDNAME4>` with the camera names (the values of $CCDNAME on the different LLCUs), `<:myPATH>` with the preferred oldb location (e.g., `:DCS:optical`):

```
#undef  DCSNAME
#undef  CCDNAME
#undef  ngcdcsINSTANCE
#undef  CCDNAME2
#undef  ngcdcsINSTANCE2
#undef  CCDNAME3
#undef  ngcdcsINSTANCE3
#undef  CCDNAME4
#undef  ngcdcsINSTANCE4
#undef  NGCROOT

#define DCSNAME <INSTRUMENT>
#define CCDNAME <myCCDNAME>
#define ngcdcsINSTANCE ngcdcs_<myCCDNAME>
#define CCDNAME2 <myCCDNAME2>
#define ngcdcsINSTANCE2 ngcdcs_<myCCDNAME2>
#define CCDNAME3 <myCCDNAME3>
#define ngcdcsINSTANCE3 ngcdcs_<myCCDNAME3>
#define CCDNAME4 <myCCDNAME4>
#define ngcdcsINSTANCE4 ngcdcs_<myCCDNAME4>
#define NGCROOT :Appl_data<:myPATH>:DCSNAME
```

Once the `DATABASE.db` and `USER.db` files have been properly edited, generate the environment: in `$VLTDATA/ENVIRONMENTS/$RTAPENV/dbl run`

```
make clean db
```

To initialize and start the environment run:

```
vccEnvInit -e $RTAPENV
vccEnvStart -e $RTAPENV
```

2.3.2. NGC-LLCU online database generation

On the NGC-LLCU, the `RTAPENV` online database environment is automatically generated and started by running:

```
ngcoDcsOldb -renv <IWS_RTAPENV> -host LLCU
```

Manpage of `ngcoDcsOldb` is available in 16.1.1.

2.3.3. Online database verification

On both the IWS and the LLCU verify that the environment has been generated:

```
dbRead "<alias><myCCDNAME>:exposure:control.state"
```

replacing `<myCCDNAME>` with the name of the camera (e.g., `$CCDNAME`).

Verify that the needed processes are running in the CCS environment, by using `ccsPerfMon`. A view similar to the following should be displayed:

Process Name	PNUM	PID	UID	GID	MSGID	MONPID
ccsScheduler	1	4224	3227	300	-1	-1
ccsSHManager	2	4228	3227	300	1582563329	-1
qsemu	3	4231	3227	300	1583415311	-1
evtEventConfg	4	4239	3227	300	1583087623	-1
timsTimeKeeper	5	4233	3227	300	1582825475	-1
scanMngr	6	4248	300	300	1583284234	-1
ccsScan	7	4259	300	300	1583480856	-1
ccsCmdServer	8	4243	3227	300	1583153160	-1
alrmServer	9	4247	3227	300	1583218697	-1
cmdManager	12	4236	3227	300	1582891012	-1
msgServer	13	4237	3227	300	1582956549	-1
logManager	14	4238	3227	300	1583022086	-1
alrmLogger	16	4268	3227	300	1583677452	-1
hisDHMngr	17	4269	300	300	1583743004	4270
ccsPerfMon	22	6532	3227	300	1589313576	-1
dbMQDBM	58	4232	3227	300	1582759938	-1

2.3.4. Online database automatic startup

On both the IWS and the LLCU the online database environment start automatically at boot by adding the following line to the file `/etc/rc.local`:

```
su - <myUser> -c 'rm -f
${VLTDATA}/ENVIRONMENTS/${RTAPENV}/.${RTAPENV}.lock;
vccEnvStart -e $RTAPENV'
```

where `<myUser>` is the user managing the online database.

2.3.5. Detector user

In order for the online database to function correctly, the user which runs NGCOSW must be defined on both the IWS and the LLCU, with the same user id.

2.4. NGC-LLCU logging system configuration

To configure the LLCU to log messages onto the IWS, edit the file `/etc/syslog.conf` while logged in as the user “root”.

```
# ========================================================
# The following three lines configure the VLT logging system
# ========================================================
#*info;mail,local1,local2.none /var/adm/messages
#local1.warning /vltdata/tmp/logFile
#local2.warning /vltdata/tmp/logAuto
*.info;mail,local1,local2.none @myIws
local1.warning @myIws
local2.warning @myIws
```

Substitute the IWS hostname for “`myIws`”.

IMPORTANT: use tabs for spacing!

This change will take effect after rebooting the LLCU, or run

```
kill -HUP `ps -C syslogd -o pid=`
```

to restart the loggin daemon.

2.5. `INS_ROOT` population

To automatically populate the `INS_ROOT` (instrument directory), the environment variable `INS_ROOT` must be defined, the directory `$INS_ROOT` must exist and the “instrument module” must have been installed (“instrument module” is the cmm module containing the detector startup configuration file <xx>dcfgCONFIG.cfg and the configuration set <xx>dcfgCAMERA.cfg - see 3.2.3.1 and 3.2.3.2 - and the voltages, patterns and sequences to drive the detector).

Assuming `<xxdcfg>` be the name of the instrument module, install it:

```
cmmCopy <xxdcfg>
cd <xxdcfg>/src; make all install
```

then populate the `INS_ROOT`:

```
ngcoDcsInstall -config <xxdcfg>
```

Manpage of `ngcoDcsInstall` is available in 16.1.2.

3. Startup/Shutdown Procedure

3.1. System Configuration

NGCOSW usually (see section 3.1.1) runs partly on the IWS and partly on the NGC-LLCU, where the physical interface(s) to the NGC detector front end reside (see Figure 1).

From now on, we will call `IWSENV` the online database environment which "usually" runs on the IWS and `LCUENV` the online database environment which "usually" runs on the NGC-LLCU (see section 3.1.1)

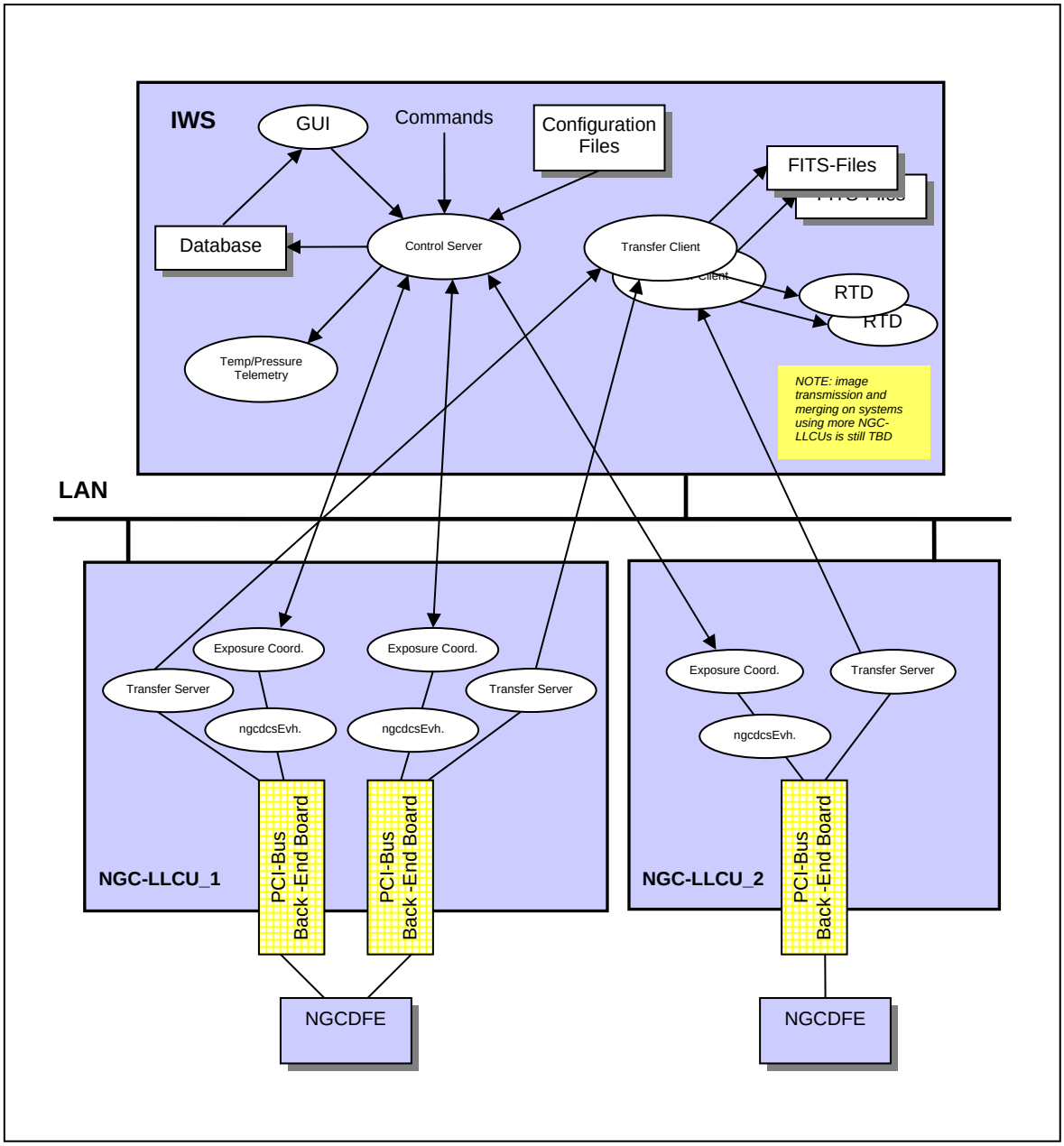

Figure 1 - Optical NGC software Architecture

For each detector system, the configuration files are kept in a separate instrument specific configuration module ***<xx>dcfg***, which is under CMM-control. The configuration module will take care of installing all files at the proper location (i.e. `$INS_ROOT/$INS_USER/COMMON/CONFIGFILES`). In addition to the system and detector configuration file(s) there are still various other files to be maintained in such a

module (e.g. voltage tables, clock pattern definitions, sequencer programs and the startup configuration as described in section 3.2.1).

3.1.1. NGCOSW operational modes

NGCOSW operates in the following different modes:

- **Normal mode**

  In Normal mode, the NGC detector electronics is connected. The NGCOSW can either be distributed on both the IWS (where the `IWSENV` online database environment is active) and the NGC-LLCU (where the `LCUENV` online database environment is active) or run completely on the NGC-LLCU (where both the `IWSENV` and the `LCUENV` online database environments are active).

- **Hardware-Simulation mode**

  In Hardware-Test mode, the NGC detector electronics is simulated (see section 3.1.3). The NGCOSW can either be distributed on both the IWS (where the `IWSENV` online database environment is active) and the NGC-LLCU (where the `LCUENV` online database environment is active) or run on a single host (where both the `IWSENV` and the `LCUENV` online database environments are active).

  This mode can be used by the higher level OS software to test the interface with the NGCOSW, when no NGC detector electronics is available.

By using the ESO VLT message system, the system configuration (i.e., where the NGCOSW processes are running) is completely transparent to the actors (instrument software, operator, engineer, etc.), because the communications between the different processes are performed through the online database environments `IWSENV` and `LCUENV`, independently from the host where these are active.

In this way, always the same software is used in all the different scenarios, in order to guarantee system robustness and behavior consistency.

NGCOSW operational mode is set by the DET.CON.OPMODE setup keyword in the camera configuration set (see section 3.2.3.2) or defined at startup (see section 3.2.1). Valid values are defined in ngco.h.

3.1.2. NGC General Purpose Control Server operational modes

NGCOSW interacts with the NGC back-end boards through the NGC General Purpose Control Server (ngcdcsEvh, see [AD72]). Within NGCOSW, the server operates in the following different modes:

- **Normal mode**

  In Normal mode, the NGC detector electronics is connected.

  This is the normal operational mode (default).

- **Hardware-Simulation mode**

  In Hardware-Simulation mode, the NGC detector electronics is simulated (see

section 3.1.3).

- **LCU-Simulation mode**

  For NGCOWS, this mode is equivalent to Hardware-Simulation.

Server operational mode is set by the DET.CON.DFEMODE setup keyword in the camera configuration set (see section 3.2.3.2) or defined at startup (see section 3.2.1). Valid values are defined in ngco.h.

3.1.3. Simulation of the NGC detector electronics

When the NGC detector electronics is simulated, the images produced by NGCOSW contain a predefined pattern.

3.2. System Startup

3.2.1. Startup Procedure

The startup procedure is based on the common VLTSW configuration tool ("***ctoo***", [RD75]).

Among the other files, an instrument module `<xx>dcfg` (see section 3.1) contains:

- a startup configuration file `<xx>dcfgCONFIG.cfg`
- one configuration set `<xx>dcfgCAMERA.cfg`

The configuration set describes an instance of the NGCOSW, in short FITS format, where (*) means that their usage is not yet implemented:

Keyword	Type	Description
DET.CON.INSTANCE	String	Defines the instance label for the control server and the database. Used to define the database branch and the appendix "_*<label>*" for the Control Coordination Process registered with the CCS environment. If the keyword is not present and not passed as a parameter to the startup script, the value of the $CCDNAME environment variable is used.
DET.CON.ENV	String	Defines the local online database environment under which the NGCOSW instance must run. If the keyword is not present and not passed as a parameter to the startup script, the value of the $RTAPENV environment variable is used.

Keyword	Type	Description
DET.CON.LENV	String	Defines the remote online database environment under which the NGC-LLCU part of NGCOSW instance must run. If the keyword is not present and not passed as a parameter to the startup script, the value of the $CCDLENV environment variable is used.
DET.CON.OPMODE	String	Defines the operational mode after starting up. Valid values are “NORMAL”, “HW-TEST”, “HW-SIM” or “LOCAL-HW-SIM”. Default is "NORMAL", in case the keyword is not present.
DET.CON.DFEMODE	String	Defines the operational mode of the NGC General Purpose Control Server after starting up. Valid values are “NORMAL”, “LCU-SIM” or “HW-SIM”. Default is "NORMAL", in case the keyword is not present.
DET.CON.AUTONLIN	Logical	When set to *T*, the detector system automatically goes to ONLINE at startup. Default is "F", in case the keyword is not present.
DET.CON.GUI	String	Launch graphical user interface with the specified process name. At the moment, only the default program *ngcouiPanel* is used, independently from the process name which is specified.
DET.CON.DICT	String	Defines a list of dictionaries to be loaded. The common “*ESO-VLT-DIC.NGCDCS*” is always loaded into the system and needs not to be specified. The entries are separated by whit-space. Only the last descriptor of the full dictionary name is needed here (e.g. “NGCDCS STOO_CFG ...”).
DET.CON.XTERM	Logical	Start all (sub-) processes in new terminal.
DET.CON.LOG (*)	Integer	Logging level. Logs system messages in the standard log-file, so that they can be seen in the CCS *logMonitor*. The level gives the detail of the messages. Default value is 0 (= no debugging, only error logging), in case the keyword is not present.

Table 1 - Startup Configuration Keywords

The startup configuration file assigns a name and some access-right attributes to the configuration set.

The system startup is performed through a startup script:

ngcoDcsStart [options]

The startup scripts loads the startup configuration defined by the `CCDNAME` environment variable or by the option `–instance` option, starts the coordination control process and waits - with a default timeout - until the coordination control process is active (i.e., it responds to `PING` commands).

options can be used to overwrite the values of the parameters in the configuration set keywords:

`-instance`	- overwrites `DET.CON.INSTANCE`
`-env`	- overwrites `DET.CON.ENV`
`-lenv`	- overwrites `DET.CON.LENV`
`-opmode`	- overwrites `DET.CON.OPMODE`
`-dfemode`	- overwrites `DET.CON.DFEMODE`
`-autonlin`	- overwrites `DET.CON.AUTONLIN`
`-gui`	- overwrites `DET.CON.GUI`
`-dict`	- overwrites `DET.CON.DICT` (*)
`-xterm`	- overwrites `DET.CON.XTERM`
`-log`	- overwrites `DET.CON.LOG` (*)

If no configuration set is given, only the *options* are used.

If no configuration is defined and no *options* are given, the system startup is performed using the environment variables `CCDNAME`, `RTAPENV`, `CCDLENV`, `INS_ROOT`, `INS_USER` (in the same way as in the case of the FIERA controller).

Further special options are:

`-kill`	- kill existing NGCOSW processes, if any, before starting

3.2.2. Changes with respect to FIERA

FIERAsw configuration was online-database-driven, i.e., the configuration of a detector was described within a `.dbcfg` (database configuration) file, which was loaded at startup. NGCOSW uses *ctoo*, the `.dbcfg` file is now substituted by the startup configuration file and the configuration set.

The script `ngcoDcsStart` replaces the script `fcdDcsStart`.

If the startup script of NGCOSW is used without the options, NGCOSW will be started using the `CCDNAME`, `RTAPENV`, `CCDLENV`, `INS_ROOT`, `INS_USER`, similar to the procedure followed by the FIERASW.

3.2.3. Configuration Examples

3.2.3.1 Startup Configuration file <xx>dcfgCONFIG.cfg

```
#
# Startup Configuration File
# ----------------------------------

PAF.HDR.START;
PAF.TYPE          "Configuration";   # Type of PAF
PAF.ID            "@(#) $Id: $";
PAF.NAME          "NGCOSW";          # Name of PAF
PAF.DESC          "NGCOSW Test Camera Startup Configuration";
PAF.CRTE.NAME     "ccumani";         # Name of creator
PAF.CRTE.DAYTIM   "2006-08-21";      # Civil Time for creation
PAF.LCHG.NAME     "         ";       # Name of person/appl. changing
PAF.LCHG.DAYTIM   "         ";       # Timestamp of last change
PAF.CHCK.NAME     "         ";       # Name of appl. checking
PAF.HDR.END;

#
# GENERAL CONFIG Keywords (optional)
# ----------------------------------
CONFIG.CON.LOG          T;
CONFIG.CON.BACKUP       T;
CONFIG.CON.BAKDIR       $VLTDATA/config;

#
# CAMERA CONFIG SET
# -----------------
CONFIG.SET1.NAME        "opd";
CONFIG.SET1.DICT        "NGCDCS NGCCON";
CONFIG.SET1.FILE1       "opdCAMERA.cfg";
CONFIG.SET1.PERM1       644;

#
# ctooConfigArchive CONFIG
# ------------------------
CONFIG.ARCHIVE.NAME     "NGCOSW";
CONFIG.ARCHIVE.USER     "";
CONFIG.ARCHIVE.MODULE   "opdcfg";
CONFIG.ARCHIVE.FILE1    "opdcfg*.cfg";
CONFIG.ARCHIVE.FILE2    "opdcfg/*";

# ___oOo___
```

3.2.3.2 Configuration Set <xx>dcfgCAMERA.cfg

```
PAF.HDR.START;                 # Start of PAF Header
PAF.TYPE          "Configuration"; # Type of PAF
PAF.ID            "";   # ID for PAF
PAF.NAME          "NGCOSW";   # Name of PAF
PAF.DESC          "NGCOSW Startup Configuration";   # Short description of PAF
PAF.CRTE.NAME     "ccumani";   # Name of creator
PAF.CRTE.DAYTIM   "2007-08-31";   # Civil Time for creation
PAF.LCHG.NAME     "         ";   # Name of person/appl. changing
PAF.LCHG.DAYTIM   "         ";   # Timestamp of last change
PAF.CHCK.NAME     "         ";   # Name of appl. checking
```

```
PAF.HDR.END;                    # End of PAF Header


#########################################################################
#                                                                       #
#                        NGCCON DICTIONARY                              #
#                                                                       #
#########################################################################

# System configuration

DET.CON.INSTANCE      "$CCDNAME";          # Instance label
DET.CON.ENV           "$RTAPENV";          # Local online database environment
DET.CON.LENV          "$CCDLENV";          # Remote online database environment
DET.CON.OPMODE        "HW-SIM";            # Operational mode
DET.CON.AUTONLIN      F;                   # Go online after start
DET.CON.GUI           "ngcouiPanel";       # GUI Name
DET.CON.DICT          "NGCDCS NGCCON";     # Dictionary list
DET.CON.XTERM         F;                   # Start in new terminal
DET.CON.LOG           0;                   # Logging level



#########################################################################
#                                                                       #
#                        NGCDCS DICTIONARY                              #
#                                                                       #
#########################################################################

# DETi keywords

DET.ID                "NGC-TEST";      # Detector system Id
DET.DATE              "2006-11-22";    # Installation date
DET.NAME              "NGC-TEST-DCS";  # Name of detector system
DET.CHIPS             1;               # Number of chips in the mosaic
BITPIX               16;               # Number of bits per pixel

CTYPE1                "PIXEL";         # Pixel coordinate system
CRVAL1                1;               # Coordinate value of ref. pixel
CTYPE2                "PIXEL";         # Pixel coordinate system
CRVAL2                1;               # Coordinate value of ref. pixel

#########################################################################
# CHIP description
#########################################################################

DET.CHIP1.ID          "SER-NO=053";    # Detector chip identification
DET.CHIP1.NAME        "Marlene";       # Detector chip name
DET.CHIP1.DATE        "2006-11-22";    # Date of installation [YYYY-MM-DD]
DET.CHIP1.NX          2048;            # Physical active pixels in X
DET.CHIP1.NY          4096;            # Physical active pixels in Y
DET.CHIP1.PRSCX       50;              # Physical prescan pixels in X
DET.CHIP1.PRSCY       0;               # Physical prescan pixels in Y
DET.CHIP1.OVSCX       50;              # Physical overscan pixels in X
DET.CHIP1.OVSCY       0;               # Physical overscan pixels in Y
DET.CHIP1.PSZX        15.0;            # Size of pixel in X (mu)
DET.CHIP1.PSZY        15.0;            # Size of pixel in Y (mu)
DET.CHIP1.OUTPUTS     2;               # Number of outputs per chip
```

```
DET.CHIP1.X           1;              # X location in array
DET.CHIP1.Y           1;              # Y location in array
DET.CHIP1.XGAP        0.0;            # Gap between chips along x (mu)
DET.CHIP1.YGAP        0.0;            # Gap between chips along Y (mu)
DET.CHIP1.RGAP        0.0;            # Angle of gap between chips
DET.CHIP1.INDEX       1;              # Chip index
DET.CHIP1.LIVE        T;              # Detector alive
DET.CHIP1.TYPE        CCD;            # The Type of detector chip
DET.CHIP1.PXSPACE     1E-6;           # Pixel-Pixel Spacing

DET.CHIP1.OUT1.NAME   "NO1";          # Description of output
DET.CHIP1.OUT1.INDEX  1;              # Output index
DET.CHIP1.OUT1.ID     "IdO1";         # Output ID as from manufacturer
DET.CHIP1.OUT1.X      1;              # X location of output
DET.CHIP1.OUT1.Y      1;              # Y location of output
DET.CHIP1.OUT1.READX  -1;             # Horizontal readout direction
DET.CHIP1.OUT1.READY  -1;             # Vertical readout direction

DET.CHIP1.OUT2.NAME   "NO2";          # Description of output
DET.CHIP1.OUT2.INDEX  2;              # Output index
DET.CHIP1.OUT2.ID     "IdO2";         # Output ID as from manufacturer
DET.CHIP1.OUT2.X      2048;           # X location of output
DET.CHIP1.OUT2.Y      1;              # Y location of output
DET.CHIP1.OUT2.READX  1;              # Horizontal readout direction
DET.CHIP1.OUT2.READY  -1;             # Vertical readout direction


#########################################################################
# DEV description
#########################################################################

DET.DEV1.NAME   "/dev/ngc0_com";    # associated device name
DET.DEV1.HOST   "$HOST";            # host where interface resides
DET.DEV1.ENV    "$RTAPENV";         # server environment name
DET.DEV1.SRV    "";                 # optional server name
DET.DEV1.TYPE   "";                 # optional type


#########################################################################
# CLDC description
#########################################################################

DET.CLDC1.DEVIDX   1;               # associated device index
DET.CLDC1.ROUTE    "2";             # route to module
DET.CLDC1.NAME     "CLDC 1";        # optional name
DET.CLDC1.AUTOENA  F;               # auto-enable at online


#########################################################################
# SEQ description
#########################################################################

DET.SEQ1.DEVIDX    1;               # associated device index
DET.SEQ1.ROUTE     "2";             # route to module
DET.SEQ1.NAME      "Sequencer 1";   # optional name
```

```
#########################################################################
# ADC description
#########################################################################

DET.ADC1.DEVIDX    1;               # associated device index
DET.ADC1.ROUTE     "2";             # route to module
DET.ADC1.NAME      "ADC Module 1";  # optional name
DET.ADC1.SIMMODE   0;               # simulation level of ADCs
DET.ADC1.OPMODE    0;               # operational mode of ADC-module
DET.ADC1.OFFSET    2.5;             # offset value for ADC (volt)
DET.ADC1.NUM       4;               # number of ADCs on board
DET.ADC1.FIRST     "T";             # first in chain
DET.ADC1.PKTCNT    0;               # packet routing length
DET.ADC1.PKTSIZE   4;               # packet size
DET.ADC1.CONVERT1  T;               # convert on strobe 1
DET.ADC1.CONVERT2  F;               # convert on strobe 2

DET.ADC1.BITPIX   16;               # ADC bits per pixel
DET.ADC1.MON1      3;               # ADC channel to monitor
DET.ADC1.CLAMP     F;               # Analog Clamp-and-Sample

#########################################################################
# MODE description
#########################################################################

# MODE1

DET.MODE1.NAME         "Test1";          # Exposure mode name
DET.MODE1.DESC         "Test mode 1";    # Exposure mode description
DET.MODE1.TRIGGER      F;                # Enable trigger
DET.MODE1.GAIN         "";               # Gain used
DET.MODE1.BNDWTH       "";               # Bandwidth used
DET.MODE1.WREP         1;                # Wipe sequence repetition number
DET.MODE1.WCLDFIL1     "wipe1.v";        # Name of CLDCi FILE for wipe
DET.MODE1.WCLKFIL1     "wipe1.bclk";     # Name of SEQi CLKFILE for wipe
DET.MODE1.WPRGFIL1     "wipe1.seq";      # Name of SEQi PRGFILE for wipe
DET.MODE1.PREP         1;                # Preint sequence repetition number
DET.MODE1.PCLDFIL1     "preint1.v";      # Name of CLDCi FILE for preintegration
DET.MODE1.PCLKFIL1     "preint1.bclk";   # Name of SEQi CLKFILE for preintegration
DET.MODE1.PPRGFIL1     "preint1.seq";    # Name of SEQi PRGFILE for preintegration
DET.MODE1.DREP         0;                # During int sequence repetition number
DET.MODE1.DCLDFIL1     "";               # Name of CLDCi FILE during integration
DET.MODE1.DCLKFIL1     "";               # Name of SEQi CLKFILE during integration
DET.MODE1.DPRGFIL1     "";               # Name of SEQi PRGFILE during integration
DET.MODE1.RREP         1;                # Readout sequence repetition number
DET.MODE1.RCLDFIL1     "read1.v";        # Name of CLDCi FILE for readout
DET.MODE1.RCLKFIL1     "read1.bclk";     # Name of SEQi CLKFILE for readout
DET.MODE1.RPRGFIL1     "read1.seq";      # Name of SEQi PRGFILE for readout
DET.MODE1.ADCSAMPL     "-1,1";           # ADC data sampling factors

DET.MODE1.OUTPUTS      1                 # Number of outputs used for readout
DET.MODE1.ADC1.ADCS    "1";              # Outputs used for readout

DET.MODE1.OUT1.CHIP    1;                # Index of chip the output belongs to
DET.MODE1.OUT1.INDEX   1;                # Output index on the chip
DET.MODE1.OUT1.XIMA    1;                # Horizontal location of data in image
DET.MODE1.OUT1.YIMA    1;                # Vertical location of data in image
DET.MODE1.OUT1.NX      2048;             # Output data pixels in X
DET.MODE1.OUT1.NY      500;              # Output data pixels in X
```

```
DET.MODE1.OUT1.PRSCX  50;              # Output prescan pixels in X
DET.MODE1.OUT1.PRSCY  0;               # Output prescan pixels in Y
DET.MODE1.OUT1.OVSCX  50;              # Output overscan pixels in X
DET.MODE1.OUT1.OVSCY  0;               # Output overscan pixels in Y
DET.MODE1.OUT1.GAIN   0.3;             # Conversion from electrons to ADU
DET.MODE1.OUT1.CONAD  3.33;            # Conversion from ADUs to electrons
DET.MODE1.OUT1.RON    123;             # Readout noise per output (e-)


# MODE2

DET.MODE2.NAME        "Test2";         # Exposure mode name
DET.MODE2.DESC        "Test mode 2";   # Exposure mode description
DET.MODE2.TRIGGER     F;               # Enable trigger
DET.MODE2.GAIN        "";              # Gain used
DET.MODE2.WREP        4;               # Wipe sequence repetition number
DET.MODE2.WCLDFIL1    "wipe1.v";       # Name of CLDCi FILE for wipe
DET.MODE2.WCLKFIL1    "wipe1.bclk";    # Name of SEQi CLKFILE for wipe
DET.MODE2.WPRGFIL1    "wipe1.seq";     # Name of SEQi PRGFILE for wipe
DET.MODE2.PREP        1;               # Preint sequence repetition number
DET.MODE2.PCLDFIL1    "preint1.v";     # Name of CLDCi FILE for preintegration
DET.MODE2.PCLKFIL1    "preint1.bclk";  # Name of SEQi CLKFILE for preintegration
DET.MODE2.PPRGFIL1    "preint1.seq";   # Name of SEQi PRGFILE for preintegration
DET.MODE2.DREP        0;               # During int sequence repetition number
DET.MODE2.DCLDFIL1    "";              # Name of CLDCi FILE during integration
DET.MODE2.DCLKFIL1    "";              # Name of SEQi CLKFILE during integration
DET.MODE2.DPRGFIL1    "";              # Name of SEQi PRGFILE during integration
DET.MODE2.RREP        1;               # Readout sequence repetition number
DET.MODE2.RCLDFIL1    "read1.v";       # Name of CLDCi FILE for readout
DET.MODE2.RCLKFIL1    "read1.bclk";    # Name of SEQi CLKFILE for readout
DET.MODE2.RPRGFIL1    "read1.seq";     # Name of SEQi PRGFILE for readout
DET.MODE2.ADCSAMPL    "-1,1";          # ADC data sampling factors

DET.MODE2.OUTPUTS     2                # Number of outputs used for readout
DET.MODE2.ADC1.ADCS   "1,3";           # Outputs used for readout

DET.MODE2.OUT1.CHIP   1;               # Index of chip the output belongs to
DET.MODE2.OUT1.INDEX  1;               # Output index on the chip
DET.MODE2.OUT1.XIMA   1;               # Horizontal location of data in image
DET.MODE2.OUT1.YIMA   1;               # Vertical location of data in image
DET.MODE2.OUT1.NX     1024;            # Output data pixels in X
DET.MODE2.OUT1.NY     500;             # Output data pixels in X
DET.MODE2.OUT1.PRSCX  50;              # Output prescan pixels in X
DET.MODE2.OUT1.PRSCY  0;               # Output prescan pixels in Y
DET.MODE2.OUT1.OVSCX  0;               # Output overscan pixels in X
DET.MODE2.OUT1.OVSCY  0;               # Output overscan pixels in Y
DET.MODE2.OUT1.GAIN   0.3;             # Conversion from electrons to ADU
DET.MODE2.OUT1.CONAD  3.33;            # Conversion from ADUs to electrons
DET.MODE2.OUT1.RON    100;             # Readout noise per output (e-)

DET.MODE2.OUT2.CHIP   1;               # Index of chip the output belongs to
DET.MODE2.OUT2.INDEX  2;               # Output index on the chip
DET.MODE2.OUT2.XIMA   2;               # Horizontal location of data in image
DET.MODE2.OUT2.YIMA   1;               # Vertical location of data in image
DET.MODE2.OUT2.NX     1024;            # Output data pixels in X
DET.MODE2.OUT2.NY     500;             # Output data pixels in X
DET.MODE2.OUT2.PRSCX  50;              # Output prescan pixels in X
DET.MODE2.OUT2.PRSCY  0;               # Output prescan pixels in Y
DET.MODE2.OUT2.OVSCX  0;               # Output overscan pixels in X
```

```
DET.MODE2.OUT2.OVSCY  0;              # Output overscan pixels in Y
DET.MODE2.OUT2.GAIN   0.3;            # Conversion from electrons to ADU
DET.MODE2.OUT2.CONAD  3.33;           # Conversion from ADUs to electrons
DET.MODE2.OUT2.RON    200;            # Readout noise per output (e-)


#########################################################################
# SHUT description
#########################################################################

DET.SHUT1.AVAIL        F;             # Shutter available or not
DET.SHUT1.CTRL         "ngc";         # Shutter controller
DET.SHUT1.TYPE         "iris";        # Shutter type
DET.SHUT1.ID           "eso-01";      # Shutter unique identifier

DET.SHUT1.DEVIDX       1;             # Device index
DET.SHUT1.ROUTE        "2";           # Route to module
DET.SHUT1.NAME         "Shutter-1";   # Optional module name


#*************************************************************
# ___oOo___#
```

3.3. NGCOSW operational states

The NGCOSW can be in the following operational states (see [AD28]):

- **OFF**. The NGCOSW is OFF when it is not running. Consequently, the NGCOSW can never reply when it is in the OFF state.
- **LOADED.** When the NGCOSW goes to LOADED state, the database is loaded and all processes are activated. Anyway the access to hardware is not allowed.

  This is the state at the end of a successful startup.
- **STANDBY.** The software and the hardware interfaces are initialized, all hardware components are checked.

  This is the state at the end of a successful STANDBY command.

In detail all actions needed to bring the whole camera to STANDBY state are very dependent on the system hardware architecture and therefore cannot be defined in this document for all cameras. Typically the following actions are implemented:

a. Detector disconnected (voltages not applied).

b. Shutter control hardware is switched off, whenever the hardware architecture allows it.

c. Temperature monitoring remains active

d. LAN connection active (command reception enabled)

- **ONLINE**. This is the only state where the NGCOSW can perform exposures. All software and hardware is loaded, initialized and active. All voltages have been loaded. Telemetry has been acquired and checked. All the voltage switches are closed.

This is the state at the end of a successful `ONLINE` command.

Figure 2 illustrates the NGCOSW operational states and the commands to switch between them (see [AD28]).

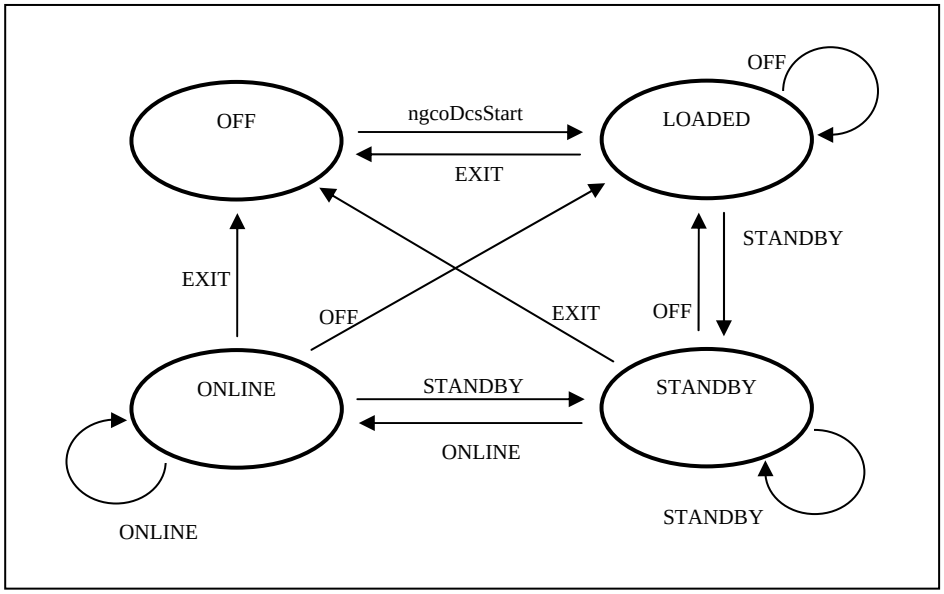

Figure 2 - Operational states and state transitions

3.3.1. Changes with respect to FIERA

NGCOSW implements the same operational states of the FIERASW.

3.4. System Shutdown

The system is shutdown by sending an `EXIT` command (see section 4) to the coordination control process `ngcocon_<label>`. The coordination control process will then shutdown all sub-processes.

A shutdown script is also available:

ngcoDcsStop [option]

The option is:

`-kill` - kill NGCOSW processes not terminated by `EXIT` command

3.4.1. Changes with respect to FIERA

NGCOSW is still shutdown by an EXIT command, like the FIERASW.

The script `ngcoDcsStop` replaces the script `fcdDcsStop`.

4. Command Interface

The coordination control process `ngcocon_<label>` is the only command interface between ICS and NGCOSW.

The commands which can be issued to the coordination control process are listed in the following table, where (*) means that they are not yet implemented:

Command	Parameters	Format	Description
ABORT	*none*	-	Abort running exposure.
BREAK	*none*	-	Interrupt NGC server.
CONT	-at	String `<YYYY-MM-DD>T<hh:mm:ss>` or "now"	Continue a paused exposure at a given time (default *now*).
DUMP (*)	*none*	-	Dump the image in memory to disk
END	*none*	-	End the current exposure(s) and read out the data.
EXIT	*none*	-	Bring the system to operational state OFF and terminate it.
INIT	*none*	-	Initialize the system. The system status goes to LOADED.
OFF	*none*	-	Bring the system to operational state LOADED.
ONLINE	*none*	-	Bring the system to operational state ONLINE.
PAUSE	-at	String `<YYYY-MM-DD>T<hh:mm:ss>` or "now"	Pause exposure at a given time (default *now*).
SELFTST (*)	-function	String	Execute a self-test (sw and hw) of the specified function(s).
SETUP	-file	String	Setup for the next exposure, or the running - but PAUSEd - one.
	-function	String	

Command	Parameters	Format	Description
STANDBY	*none*	-	Bring the system to operational state STANDBY.
START	-expoId	Integer	Exposure ID
	-at	String <YYYY-MM-DD>T<hh:mm:ss> or "now"	Start an exposure or an exposure loop (depending on the value of DET.EXP.NREP keyword, see section 7) at a given time (default *now*).
STARTTL (*)	-period	Integer	Start monitoring of telemetry values.
	-logperiod	Integer	
STARTWP (*)	-periodic	Integer	Wipe chip(s) once or periodically.
STOPLP	*none*	-	Stop a loop of repeated exposures, at the end of the running exposure.
STOPTL (*)	*none*	-	Stop monitoring of telemetry values.
STOPWP (*)	*none*	-	Stop a periodic wipe.
VERBOSE	-on -off	-	Set verbose mode on/off. If *-on*, the level of the logging is defined by the logging level value (which can be set/modified through the setup keyword *DET.CON.LOG*)
VERSION	*none*	-	Return current version of the NGCOSW.
WAIT	-waitMode	String "Single" / "Global"	Wait for exposure completion. Two replies are issued: one immediate with the exposure status, one at the end of the exposure.

Table 2 - Command list

Special commands which can be issued directly to the different processes are listed in the following table, where (*) means that they are not yet implemented:

Command	Parameters	Format	Description
KILL	*none*	-	Kill the process. The system status goes to OFF.
PING	*none*	-	Verify whether the process is able to send or receive messages
SIM/SIMULAT (*)	*none*	-	Put the process into simulation mode.
STATUS	*none*	-	Get the status of the process.
STOPSIM (*)	*none*	-	Stop the simulation.

Table 3 - Special command list

4.1. Changes with respect to FIERA

NGCOSW implements the same commands of the FIERASW.

To keep backward compatibility with the FIERASW as much as possible, but reducing at the same time differences with the NGC software for the infrared detectors, some command aliases have been provided (e.g., `SIM`/`SIMULAT`).

The command `STOPLP` replaces `STOP`.

5. Multiple Instances of DCS

The coordination control process `ngcocon_<label>` is the only command interface between ICS and NGCOSW (see section 4).

If multiple instances of DCS are used (e.g., for instruments which control more than one NGC-LLCU), the coordination control process of the master DCS is the only command interface between ICS and NGCOSW.

Image format is defined in 9.3.

Image transfer and merging for this configuration case are however still TBD.

6. Database Interface

Some attributes of the NGCOSW online database are made public for direct read operations from external software (note: they are read only).

When accessing NGCOSW database attributes with direct CCS db calls, **applications are requested to use the macros defined in** `ngcoDbPublic.h` (see section 6.4): in this way, any change in name or location of the attribute only requires a new compilation.

All database paths below are meant to be relative to the root point for the NCG database branch.

6.1. Interface between NGCOSW and the external environment

Point	Attribute	Type	Description
system	***opmode***	dbINT32	Camera operational mode
system	***state***	dbINT32	System operational state
exposure:config	***id***	dbINT32	Exposure identification number
exposure:config	***expMode***	dbINT32	Exposure mode index
exposure:config	***expModeDescr***	dbBYTES32	Exposure mode description
exposure:config	***wipeSeq***	dbBYTES128	Name of the wipe sequence
exposure:config	***wipeClock***	dbBYTES128	Name of the wipe clock pattern
exposure:config	***wipeVolt***	dbBYTES128	Name of the wipe voltage set
exposure:config	***wipeRep***	dbINT32	Wipe sequence repetition factor
exposure:config	***preintSeq***	dbBYTES128	Name of the preintegration sequence
exposure:config	***preintClock***	dbBYTES128	Name of the preintegration clock pattern
exposure:config	***preintVolt***	dbBYTES128	Name of the preintegration voltage set
exposure:config	***preintRep***	dbINT32	Preintegration sequence repetition factor
exposure:config	***durintSeq***	dbBYTES128	Name of the during integration sequence
exposure:config	***durintClock***	dbBYTES128	Name of the during integr. clock pattern
exposure:config	***durintVolt***	dbBYTES128	Name of the during integration voltage set
exposure:config	***durintRep***	dbINT32	During integr. sequence repetition factor
exposure:config	***readSeq***	dbBYTES128	Name of the readout sequence
exposure:config	***readClock***	dbBYTES128	Name of the readout clock pattern
exposure:config	***readVolt***	dbBYTES128	Name of the readout voltage set
exposure:config	***readRep***	dbINT32	Readout sequence repetition factor

Point	Attribute	Type	Description
exposure:config	***expType***	dbBYTES32	Exposure type (Normal/Bias/Dark/etc)
exposure:config	***expRepeat***	dbINT32	Number of exposure repetitions
exposure:config	***expTime***	dbDOUBLE	Exposure time
exposure:config	***pWipeEnabled***	dbLOGICAL	Periodic wipe enabled or not
exposure:config	***pWipePeriod***	dbINT32	Wipe period
exposure:config	***fileName***	dbBYTES128	Name of FITS file with image
exposure:control	***state***	dbINT32	Current state of exposure
exposure:control	***stateDescr***	dbBYTES32	Description of current state of exposure
exposure:control	***shutter***	dbINT32	Shutter status
exposure:control	***shutterDescr***	dbBYTES32	Description of shutter status
exposure:control	***remTime***	dbDOUBLE	Remaining time to complete exposure
exposure:control	***readTime***	dbDOUBLE	Time to read image data from detector
exposure:control	***tranPercent***	dbINT32	Percentage of image transferred to WS
exposure:control	***tranTime***	dbDOUBLE	Time to transfer image to WS
telemetry:config	***enabled***	dbLOGICAL	Telemetry enabled or not
telemetry:control	***state***	dbINT32	Current state of telemetry monitoring
telemetry:data	***current***	vector of dbDOUBLE	Current telemetry values

Table 4 - Online database attributes for detector system monitoring

NOTE: the interface with BOSS is still under verification, therefore the attributes above could be modified (and the definitions in `ngcoDbPublic.h` will be updated).

6.2. Interface between NGCOSW and TCS

Point	Attribute	Type	Description
wcs	***ra***	dbDOUBLE	Centre right ascension in degrees for World Coordinates display
wcs	***dec***	dbDOUBLE	Centre declination in degrees for World Coordinates display

Table 5 - Online database attributes for TCS

6.3. Image processing interface

TBD.

6.4. ngcoDbPublic.h

For all the above attributes, a macro is defined in the `ngcoDbPublic.h`.

When accessing NGCOSW database attributes with direct CCS db calls, applications are requested to use the macros defined in `ngcoDbPublic.h`: in this way, any change in name or location of the attribute only requires a new compilation.

6.5. Changes with respect to FIERA

NGCOSW keeps the same public online database attributes of the FIERASW.

7. Setup Command

All the parameters which are relevant for an exposure are set via a SETUP command, which must therefore be issues before starting an exposure (unless the new exposure is a perfect copy of the previous one, i.e., no parameter needs to be modified) or while an exposure is paused.

Here is a selection of the most important setup keywords (to be completed), where (*) means that their usage is not yet implemented:

Keyword	Type	Description
DET.MODE.CURID	Integer	Index of mode used for an exposure (wipe, integrate, readout)
DET.EXP.TYPE	String	Exposure type: *Normal* one integration, shutter (if any) open *Dark* one integration, shutter (if any) closed *Bias* one integration, 0 integration time, shutter (if any) closed *Flat* one integration, shutter (if any) open *Led* one integration, shutter (if any) closed, LED light source on *LedShut* one integration, shutter (if any) open, LED light source on *Multiple* DET.WIN<i>.NDIT sub-integrations, shutter (if any) open for each integration *Burst* Multiple frames are read out, shutter (if any) always open
DET.WIN<i>.UIT1	Double	Integration time (in seconds)
DET.WIN<i>.BINX	Integer	Binning factor along X
DET.WIN<i>.BINY	Integer	Binning factor along Y

Table 6 - Basic Setup keywords for single exposure

Keyword	Type	Description
DET.EXP.NREP	Integer	Number of repeated exposures. "0" means "forever"
DET.EXP.TIMEREP (*)	Double	Time between two repeated exposures
DET.EXP.WIPETIM (*)	Integer	Wipe or not before starting exposure in a loop

Table 7 - Additional Setup keywords for loops of exposures

Keyword	Type	Description
DET.WIN<i>.NDIT (*)	Integer	Number of sub-integrations
DET.WIN<i>.UIT<j> (*)	Double	Subintegration time (in seconds)
DET.READ.SHIFT<i> (*)	Integer	Lines shifted between integrations
DET.READ.SHIFTYP (*)	String	Line shift type: 'alternate' : 'alternate' +SHIFT1,-SHIFT1 'idem' as SHIFT1 'list' as defined in list SHIFTi

Table 8 - Setup keywords for multistep exposures

Keyword	Type	Description
DET.WIN<i>.STRX (*)	Integer	First (lower left) window pixel in X direction
DET.WIN<i>.STRY (*)	Integer	First (lower left) window pixel in Y direction
DET.WIN<i>.NX (*)	Integer	Number of pixels along X
DET.WIN<i>.NY (*)	Integer	Number of pixels along Y

Table 9 - Setup keywords for windowing (not yet implemented)

NOTE: in the actual version of NGCOWS, windowing is not implemented.

Keyword	Type	Description
DET.DISPLAY (*)	Integer	Real Time image display -1 no display 0 full frame display 16 bits 1 rapid frame display 16 bits
DET.CHIP<i>.CRPIX<i> (*)	Integer	Reference pixel in <i> direction.
DET.FRAM.SAMPLE (*)	Integer	Image sampling on workstation.
DET.READ.NFRAM (*)	Integer	Defines how many sequential image data shall be stored inside a single FITS file. Default is "1"
DET.FRAM.FITSMTD (*)	Integer	Data storage method. Possible values: 0 = 'none' (image is not saved on disk) 1 = 'compressed' 2 = 'uncompressed' 3 = 'both'
DET.FRAM.FILENAME	String	Define the base filename for the data files produced during the exposure

Table 10 - Setup keywords for image display

Arguments of the `SETUP` command can be file containing sets of keywords (`-file` option) or keywords (`-function` option). For example:

```
msgSend $RTAPENV ngcocon_<label> SETUP \
   "-file mysetup.det"
msgSend $RTAPENV ngcocon_<label> SETUP \
   "-function DET1.WIN1.UIT1 2.5"
```

7.1. Changes with respect to FIERA

The setup keyword `DET.MODE.CURID` replaces `DET.READ.CLKIND`.

The setup keyword `DET.FRAM.FILENAME` replaces `DET.FRAM.FITSUNC`.

The setup keyword `DET.FRAM.FITSMTD` is obsolete and not accepted any more.

8. Status Command

The STATUS command issued to the coordination control process `ngcocon_<label>` returns the status of all the processes.

For debugging, the STATUS command can also be sent to the NGC general Purpose Control Server `ngcdscEvh_<label>` (see section 3.1.2), using the parameters described in [RD77].

8.1. Changes with respect to FIERA

The STATUS command was not implemented in the FIERASW.

9. Exposure Handling

9.1. Description

9.1.1. Exposure types

NGCOSW distinguishes among the different types of exposure defined in the Glossary (see [AD63] for a more detailed description):

- **Normal exposure** (single integration, shutter opened and closed)
- **Dark exposure** (single integration, shutter kept closed)
- **Bias exposure** (0 integration time Dark)
- **Flat Field exposure** (normal exposure, chip exposed to a uniform flux of radiation)
- **LedAndShutter exposure** (normal exposure, chip exposed to the radiation generated by a LED, which is located between the chip and the shutter).

  NOTE: this kind of exposure will be supported or not depending on the capability of the hardware that will be finally chosen**.**
- **Led exposure** (dark exposure, chip exposed to the radiation generated by a LED, which is located between the chip and the shutter)

  NOTE: this kind of exposure will be supported or not depending on the capability of the hardware that will be finally chosen**.**
- **Multiple or Multi-step exposure** (single exposure consisting of more integrations, with same or different duration. After each integration, the exposure is paused. During pauses, rows may be shifted on chip)
- **Burst or Drift Scanning exposure** (during the integration the charges on the CCD are continuously shifted along the parallel registers and read out)

The exposure type is defined by setting the `DET.EXP.TYPE` setup keyword (see section 7).

Accepted values for the `DET.EXP.TYPE` setup keyword are listed in macros which are defined in ngco.h.

9.1.2. Exposure status

When the detector system is `ONLINE`, an exposure can be prepared with a `SETUP` command (see section 7) and executed with a `START` command.
Schematically, starting an exposure means to:

- wipe a chip (depending on setup) : the exposure status will be *wiping*
- wait for the time to open the shutter (depending on START “-at” parameter , usually is “now”) : the exposure status will be *pending*
- open a shutter (depending on setup) : the exposure status will be *integrating*
- collect the radiation on the chip

- close a shutter (depending on setup)
- read the chip : the exposure status will be *reading*
- transfer the data to the IWS: the exposure status will be *transferring*

When the image data have been stored on disk, the exposure status goes to *completed*. If an error occurred during the exposure, the status goes to *failed*. If the exposure was aborted, the status goes to *aborted*.

Generally the field of view can already be changed (e.g. telescope can be moved) when the exposure status changes to *transferring* (all data for this exposure have been read-out).

By default, with NGCOSW it is possible to start an exposure only when one of the completion states (*success*, *failure*, *aborted*) have been reached (i.e., after the image data produced by the previous exposure have been saved on disk).

If the time between end of detector readout and availability of the FITS file on disk becomes a significant overhead, NGCOSW can be instructed to start an exposure right after the end of the transmission of the image date of the previous exposure to the IWS, by using the setup keyword `TBD`.

The current exposure status value is stored in the database attribute

```
<alias>${CCDNAME}:exposure:control.state
```

The value of the current exposure status can be:

INACTIVE
PENDING
WIPING
INTEGRATING
PAUSED (i.e., shutter temporary closed)
READING
PROCESSING (i.e., processing image data, if requested by SETUP)
TRANSFERRING (i.e., transferring image data to IWS)
COMPLETED (i.e., completed successfully)
FAILED (i.e., completed with error)
ABORTED (i.e., completed without data readout, on request)

Macros for the exposure status values and descriptions are defined in ngco.h.

9.1.3. Image data

Image data are provided by NGCOSW in two ways:

- Raw-data for Real-time display (see section 13)
- FITS files (see section 9.3)

Whenever a new data file is created, the full path name is written into the database attribute

```
<alias>${CCDNAME}:exposure:config.fileName
```

9.1.4. Exposure Id

In order to be able to uniquely identify an exposure, an identification number (exposure Id)

is associated to each exposure.
The exposure Id should be passed to the NGCOSW as a parameter of the command START (as defined in [AD28]). If the command START has no exposure Id parameter, the exposure Id is defined by NGCOSW.

The exposure Id is returned as a reply parameter to the command START.

9.1.5. Changes with respect to FIERA

NGCOSW implements the same exposure and status types of the FIERASW.

NGCOSW defines the same exposure status numerical values of the FIERASW, a part from the one labelling the WIPING status (see ngco.h).

NOT YET IMPLEMENTED: Has a new feature, it will be possible to start a new exposure when the data of the previous one have been transmitted to the IWS, but not stored on disk yet (although this will NOT be the default behavior).

9.2. Commands

Exposures are prepared using the **SETUP** command and started using the **START** command.

A timed exposure start can be done using the **-at** option:

```
START –at <YYYY-MM-DD>T<hh:mm:ss>
```

The value of the `-at` parameter defines an absolute time (UTC) for the opening of the shutter (an absolute time for a dark exposure has no sense). Until the actual start time is reached, the exposure status is set to “pending”, which will limit the set of accepted commands during that time.

An exposure can be paused using the command **PAUSE** (note that the time the exposure is PAUSE'd will be added to the dark's time, see [RD63]). The shutter is closed and the counting of the remaining exposure time suspended. The exposure is then restarted by the command **CONTINUE**.

The exposure can be aborted using the command **ABORT**. In this case no data file is generated unless a frame was already received at the time when the command was issued.

The command **END** makes the acquisition process terminate the exposure as soon as possible and generate a data file.

The command **WAIT** can be used to wait for an exposure to complete. A reply message with the current exposure status is sent immediately. When the exposure status is (or becomes) “completed” (i.e. “success”, “failure” or “aborted”), NGCOSW sends the last reply, which again contains the actual exposure status. A running exposure always has to be waited for completion before starting the next one or before issuing a new setup.

Typical command sequences are:

```
a) START – WAIT
b) START – PAUSE - CONTINUE – WAIT
c) START – END – WAIT
```

d) START - ABORT – WAIT

Alternatively, the exposure status attribute in the database (see section 6.1) may be used to wait for a specific state (e.g. `transferring`).

9.2.1. Changes with respect to FIERA

NGCOSW implements the same exposure commands of the FIERASW.

9.3. File Formats

If data storage is enabled, images are saved in the `$INS_ROOT/$INS_USER/DETDATA` directory as FITS files compliant with [RD37], i.e., using the “image extension per chip" format. In this format, data are ordered by chip: each CCD corresponds to an extension. A primary header sits on the top of the file.

NOT YET IMPLEMENTED: To enable "data cubes" (i.e., saving n successive frames into a single FITS file) the setup parameter `DET.READ.NFRAM` must be set to a value different form `TBD` (see section 7).

Currently the formats supported for pixels values are 16-bits and 32-bits.

Independently from the readout mode used, the complete physical image is stored in one single FITS file per camera head (provision for one single FITS file per instrument is under development). Multiple windows are also stored in different IMAGE extensions of a single FITS file. Different frames in data-cube files are also stored in different IMAGE extensions of a single FITS file (see [RD37]).

9.3.1. Changes with respect to FIERA

NGCOSW implements the same file format of the FIERASW.

9.4. Naming Schemes

FITS file names are defined by the setup keyword `DET.FRAM.FILENAME` (see section 7).

In case the number of FITS file to be produced is more than one (`DET.EXP.NREP` setup parameter: see section 7), NGCOSW assumes that all files will have the same name, followed by a sequential integer index, starting from 0.

Example: if `DET.EXP.NREP` is set to 3 and `DET.FRAM.FILENAME` is set to *myImage.fits,* NGCOSW will look for files *myImage.fits* (first exposure), *myImage.1.fits* (second exposure) and *myImage.2.fits* (third exposure)

9.4.1. Changes with respect to FIERA

NGCOSW implements the same naming scheme of the FIERASW.

9.5. FITS-Header Contents

Basic and mandatory primary FITS keywords are included from the dictionary (***dicFITS*** CMM module) `ESO-VLT-DIC.PRIMARY-FITS`.

NGC specific FITS keywords are defined within the `ESO-VLT-DIC.NGCDCS` dictionary (***dicNGC*** CMM module).

Apart from the image raw data, NGCOSW is also responsible for providing keywords for the FITS header. Depending on their type, keywords are treated in two different ways.

- **Standard keywords**. Some basic keywords, needed by any image analysis system to read the FITS file, are written at the beginning of the file.
- **Hierarchical keywords**. They are not strictly needed to interpret the pixel values and normally do not appear at the beginning of the FITS header.

NOTE: at the moment, NGCOSW writes all the information directly in the FITS file, removing the usage of an intermediate `.det` file to be merged with the FITS file by OS, as was done by FIERA. This implementation is still under discussion.

9.5.1. Changes with respect to FIERA

NGCOSW implements the same FITS structure of the FIERASW.

If actual scheme is accepted, no more separate `.det` files are created: NGCOSW writes all the information directly in the FITS file.

9.6. Image processing

TBD.

10. Synchronisation

Synchronization points can be inserted at any place in any clock pattern executed by the sequencer program (i.e. set the "*wait-for-trigger*" bit in the particular state). When reaching such a point, the pattern execution is suspended until the arrival of an external trigger signal (see [RD9] and [RD8] for signal timing and accuracy). Via this external trigger input it is possible to synchronize exposures on multiple NGCOSW instances. The external trigger signal is also used to synchronize detector read-outs with external devices. Using the VLT-TIM for generating the trigger pulse(s), synchronization at absolute times is possible. Some signal lines are available to in turn trigger external devices (e.g. tell another device, that a read-out has finished).

If several sequencers are installed in the same system (i.e. the same instance of NGCOSW), then the exposure start can be synchronized by using the global run-signal, which is raised by one sequencer instance and is propagated to all other sequencer instances having the external run-control enabled ("`DET.SEQi.RUNCTRL = T`" in the detector configuration file, see [RD77]).

If no high accuracy is needed, the synchronization can be also done at command interface level (e.g. issue an exposure start command at the proper time or use the command "`START –at <start-time>`").

11. Error Definitions

The CCS error mechanism [RD32] provides a classification scheme for application specific errors.

The meaning of the error class and the possibly needed interactions are described in a help file (`.hlp`), which can be displayed with the standard CCS-tools (also with the *logMonitor*).

The detailed error reason (e.g., command which failed, wrong parameter issued and boundary values, etc) is given in an associated error message string.

NGCOSW uses the errors defined by *ngcb* (see [RD77]).

In addition, NGCOSW modules define their own errors. The errors which are common to all the modules are listed in the following table:

Error	Severity	Description
ngco<mod>ERR_FATAL	fatal	Fatal internal error: %.40s
ngco<mod>ERR_CREATE	serious	Failure creating: %.40s
ngco<mod>ERR_INIT	serious	Failure initializing: %.40s
ngco<mod>ERR_FUNCTION	serious	Failure invoking function: %.40s
ngco<mod>ERR_NULL_POINTER	warning	Pointer to %.40s is NULL
ngco<mod>ERR_ZERO	warning	Division by zero while %.20s
ngco<mod>ERR_NOT_FOUND	warning	Object %.20s (ID: %d) not found
ngco<mod>ERR_DB_READ	serious	Failed to read from DB %.35s %.35s%.35s
ngco<mod>ERR_DB_WRITE	serious	Failed to write to DB %.35s %.35s%.35s
ngco<mod>ERR_SEND_COMMAND	serious	Failed to send command %.20s to %.40s
ngco<mod>ERR_SEND_REPLY	serious	Failed to send %.20s reply for command %.20s to %.40s
ngco<mod>ERR_REPLY	serious	Error reply to command %.20s received from %.40s

Error	Severity	Description
ngco<mod>ERR_TIMEOUT	serious	Command %.20s sent to %.40s timed out
ngco<mod>ERR_PARAMETER	warning	Parameter %.20s has invalid value
ngco<mod>ERR_ABORTED	warning	Command %.20s aborted
ngco<mod>ERR_STATE	serious	Command %.20s not allowed in state %.40s
ngco<mod>ERR_ACTION	serious	Action %.20s (%.15s) failed in state %.40s
ngco<mod>ERR_EVENT	serious	Event %.20s not handled in state %.40s
ngco<mod>ERR_ASSERT	serious	Assertion Failed %.20s %.40s %d.

Table 11 - Errors common to all NGCOSW modules

The errors which are specific to certain modules of the NGCOSW are listed in the following tables:

Error	Severity	Description
ngcoitcERR_NOT_DEFINED	serious	Variable %.20s not defined
ngcoitcERR_RTD	serious	Error communicating to RTD %.40s

Table 12 - Errors specific to the `ngcoit` module

NOTE: it is under investigation the usage of an error definition file common to all the modules.

12. Error and Logging Handling

Error and system logging is performed using the standard CCS error and logging systems (see [RD32]).

NOT YET IMPLEMENTED: Additionally the verbose output can be logged in a detail depending on the given log-level (see setup keyword `DET.CON.LOG` in section 3.2.1 and command `VERBOSE` in section 4) for maintenance and debugging purposes.

Operational logs are TBD.

13. Real-Time Display Interface

NGCOSW provides **raw data** for the VLTSW real-time display utility `rtd`.

The mechanism to deliver raw data is the same as defined in [RD40].

Raw-data are written in shared memory as they come out from the Detector Electronics, namely with full resolution (16 or 32-bits unsigned integer). No reduction (e.g. to 8-bits) is done by NGCOSW.

In addition to the display of the raw-data, NGCOSW supports also the display of **World Coordinates** through `rtd`. One point in the NGC branch of the online database is dedicated to this feature (see section 6.2).

13.1. Changes with respect to FIERA

NGCOSW provides the same interface to `rtd` of the FIERASW.

14. Graphical User Interface

14.1. Control Panel

A graphical user interface is provided to operate NGCOSW in standalone mode.

One single panel, shown in Figure 3, provides all functionality needed to startup/shutdown the NGC software, define an exposure setup, start and control an exposure, display an image as result of an exposure.

The same panel is used, independently if and which parts of the CCD system used are simulated.

To startup the optical NGC Control Panel run

```
ngcouiPanel &
```

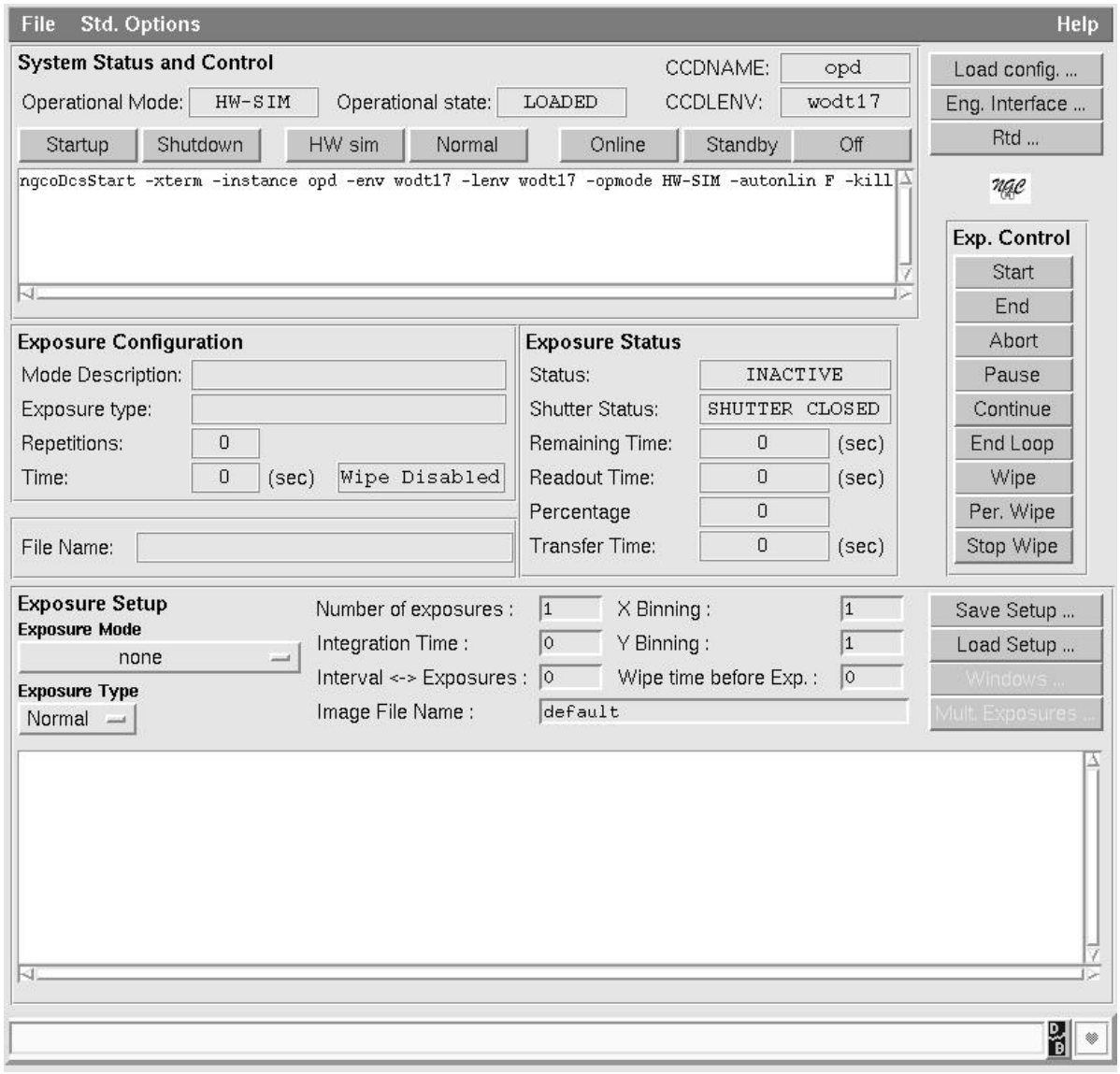

Figure 3 - Optical NGC Control Panel

14.2. Engineering Interface

A GUI panel is provided to help engineers in case of trouble (see Figure 4). It is evoked from the Control Panel and enables the most common operations needed for engineering.

The panel gives freedom to do actions at a low level and must be used with care !

It is assumed that the user knows the NGC sw and the VLT sw environment and is fully aware of the actions associated to each button and possible consequences.

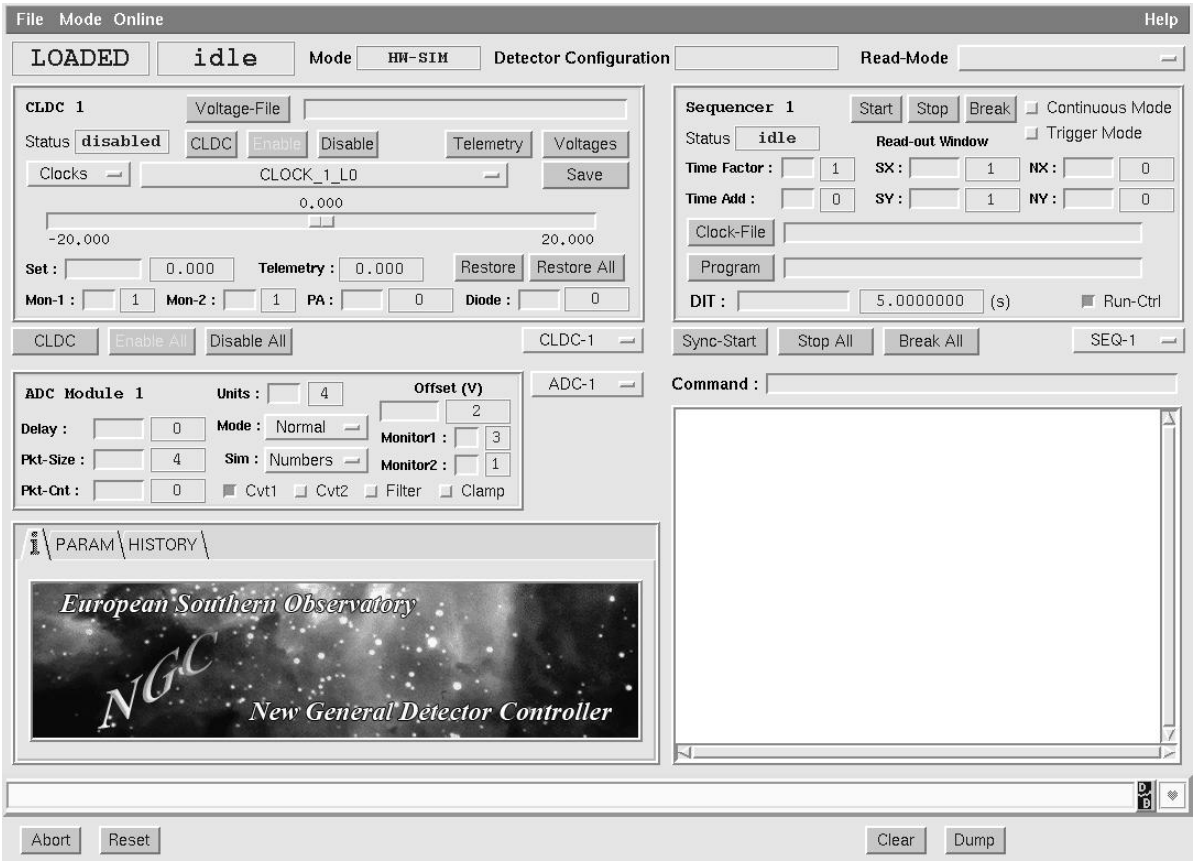

Figure 4 - NGC Engineer Panel

15. Special functionalities for Optical Instruments

15.1. Shutter Control

Shutter configuration for each system is stored within the instrument specific configuration module ***<xx>dcfg***, which is under CMM-control.

At the moment, shutter control is performed via the Pulpo Server (although this should change in a near future). The device used to physically connect to the shutter is defined in the file

```
$INS_ROOT/$SYSTEM/COMMON/CONFIGFILES/$CCDNAME/pulpo.cfg
```

which must correctly set.

Here is a self-explanatory example of a `pulpo.cfg` file for a system with 2 shutters connected via `ttyc` and `ttyd`:

```
#
# "@(#) $Id: pulpo.cfg,v 1.44 2004/05/10 22:47:31 vltsccm Exp $"
# ---------------------------------------------------------------
#
# Pulpo configuration
#
# format is:  Pulpo_Unit_Number  Full_Device_Path
1 /dev/ttyc
2 /dev/ttyd
```

15.2. Temperature/pressure Monitoring

At the moment, the way the temperature/pressure monitoring will be handled by NGC is under evaluation.

15.2.1. Changes with respect to FIERA

If NGCOSW will provide facilities to monitor temperature and pressure values from the detector, no changes are foreseen with respect to the monitoring interface provided by the FIERASW.

15.3. Adaptive Optics

TBD

16. Manpages

16.1.1. ngcoDcsOldb

```
NAME
       ngcoDcsOldb.sh  - Install and generate the online database environment.


SYNOPSIS
       ngcoDcsOldb -host <IWS | LLCU> -renv <renv> [<renv2> [<renv3> <...>]]]
                         [-ccdname <ccdname>] [-env <env>]


DESCRIPTION
       On both the Instrument Workstation (IWS) and the NGC Linux LCU (LLCU)
       this shell script preliminary performs a system check (definition
       of environment variables, definition of local and remote
       environments on the local machine and in the ACC server, scanning,
       user running the software, etc.).
       On an IWS this shell script then installs in the directory
          $VLTDATA/ENVIRONMENTS/$RTAPENV/dbl/
       the template files (DATABASE.db.NGCOSW and USER.db.NGCOSW)
       which can be used to generate the online database for
       an optical NGC system.
       On a LLCU this shell script generates and starts the online
       database environment.

       -host <IWS|LLCU>    Defines if the database must be generated
                           on an IWS or on A NGC LLCU

       -renv <renv>        name of remote online database environment
                           (on the IWS this is $CCDLENV, on the NGC LLCU
                           this is the $RTAPENV of the IWS).
                           ONLY IN THE "IWS" CASE, more <renv> can be given,
                           to check if they are all known by the
                           ACC server.

       -ccdname <ccdname>  detector name (default $CCDNAME)

       -env <env>          name of local online database environment
                           (default $RTAPENV).

       -user <user>        name of the user running NGCOSW


FILES
       Source files:
         $VLTTOP/ENVIRONMENTS/ngco/DATABASE.db
         $VLTTOP/ENVIRONMENTS/ngco/USER.db
       Generated files on IWS:
         $VLTDATA/ENVIRONMENTS/$RTAPENV/dbl/DATABASE.db.NGCOSW
         $VLTDATA/ENVIRONMENTS/$RTAPENV/dbl/USER.dbNGCOSW
       Generated files on LLCU:
         $VLTDATA/ENVIRONMENTS/$RTAPENV/dbl/DATABASE.db
         $VLTDATA/ENVIRONMENTS/$RTAPENV/dbl/USER.db


ENVIRONMENT
       CCDNAME             CCD camera name
       RTAPENV             Online database environment name


RETURN VALUES
```

```
       0 if SUCCESS
       1 if FAILURE


EXAMPLES
       > ngcoDcsOldb -host LLCU -renv wte98
       > ngcoDcsOldb -host IWS -renv wodt8
       > ngcoDcsOldb -host IWS -renv wodt6 wodt8 -ccdname mycam –user myuser
```

16.1.2. ngcoDcsInstall

```
NAME
       ngcoDcsInstall.sh - Install NGCOSW files in INS_ROOT


SYNOPSIS
       ngcoDcsInstall -config <detector_module> [- root <ins_root>]


DESCRIPTION
       This shell script installs all files needed to run an
       optical NGC system (configuration files, CCD voltages, clock
       patterns and sequences) in the instrument directory
       <ins_root>/SYSTEM/COMMON/CONFIGFILES/

       -config <detector_module>   Name of the detector module whose
                                   files in $VLTTOP/config contain
                                   the detector configuration.

       -root <ins_root>            Root directory for the instrument the
                                   NGCOWS belongs to
                                   Default: $INS_ROOT (env. variable)


FILES
       Source files:
       $VLTTOP/config/??dcfgCONFIG.cfg  ctoo configuration file for
                                        the optical NGC detector.
       $VLTTOP/config/??dcfgCAMERA.cfg  system configuration file for
                                        the optical NGC detector.
       $VLTTOP/config/??dcfg/*          Files defining the detector
                                        voltages, clock patterns
                                        and sequences.
       Destination files:
       <ins_root>/SYSTEM/COMMON/CONFIGFILES/${CCDNAME}CONFIG.cfg
       <ins_root>/SYSTEM/COMMON/CONFIGFILES/${CCDNAME}CAMERA.cfg
       <ins_root>/SYSTEM/COMMON/CONFIGFILES/${CCDNAME}/*


ENVIRONMENT
       CCDNAME                     CCD camera name
       RTAPENV                     Online database environment name
       INS_ROOT                    default instrument root directory
       INS_USER                    default to SYSTEM


RETURN VALUES
       0 if SUCCESS
       1 if FAILURE


EXAMPLES
       > ngcoDcsInstall -config opdcfg
       Install all what needed for scientific CCD whose configuration
       files are stored in the "opdcfg" module
```

16.1.3. ngcoDcsStart

```
NAME
       ngcoDcsStart - startup of optical NGC DCS


SYNOPSIS
       ngcoDcsStart [-instance <ccdname>] [-env <env>] [-lenv <lenv>]
                    [-opmode <opmode>] [-gui <T|F>] [-xterm <T|F>]
                    [-autonlin <T|F>] [-kill]


DESCRIPTION
       This shell script performs a startup of NGC optical DCS.

       -instance <ccdname> detector name (default $CCDNAME)

       -env <env>          name of workstation online database environment
                           (default $RTAPENV).

       -lenv <lenv>        name of remote online database environment
                           (default $CCDLENV)
                           If lenv=0, only the NGC LCU processes are started.

       -opmode <opmode>    NGCOSW operational mode (default "NORMAL")
                           Valid values are:
                           NORMAL        - Normal Operational Mode (Default)
                                           NGC HW is used,
                                           NGC SW runs on Instrument
                                           Workstation (IWS) and NGC-LCU
                           HW-SIM        - HW is simulated

       -gui [<guiname>]    Launch the specified grafical user interface.
                           If no <guiname> is given, the default ngcouiPanel
                           is used.
                           At the moment, only the default program ngcouiPanel
                           is used, independently from the process name which
                           is specified.

       -xterm <T|F>        When set to T, all processes are started
                           in new xterminals.
                           Default is "F".

       -autonlin <T|F>     When set to T, the detector system automatically
                           goes to ONLINE at startup.
                           Default is "F".

       -kill               kill all already running processes before starting


ENVIRONMENT
       CCDNAME        default for camera name (e.g. myccd)
       RTAPENV        default for WS local environment (e.g. myws)
       CCDLENV        default for LCU environment (e.g myngc)
       INS_ROOT       default root directory for instrument data


RETURN VALUES
       0 if SUCCESS
       1 if FAILURE
```

```
CAUTIONS
        .rhosts file on LCU system must contain user and hostname
       where this script runs, since it performs remote shell commands


EXAMPLES
       > ngcDcsStart -instance myccd -env myws -lenv myngc
           Start the NGCOSW for camera "myccd",
           WS environment "myws", LCU environment "myngc".

       > ngcDcsStart -instance myccd -env myws -lenv 0
           Start only the NGCOSW LCU processes
           for camera "myccd", WS environment "myws".

       > ngcDcsStart -instance myccd -env myws -lenv myngc -kill -gui
           Kill and restart the NGCOSW for camera "myccd",
           WS environment "myws", LCU environment "myngc".
           Gui is also started


SEE ALSO
       ngcoDcsStop, ngcGetProcNum
```

16.1.4. ngcoDcsStop

```
SYNOPSIS
       ngcoDcsStop [-instance <ccdname>] [-env <env>] [-lenv <lenv>] [-kill]


DESCRIPTION
       This shell script performs a shut-down of NGC optical DCS.
       It does the following steps:
         1 - Verify if the main process is running.
         2 - Try to terminate NGC optical processes in a 'soft' way
             (command EXIT)
         3 - Try to terminate NGC optical processes in a 'hard' way
             (kill) - optional

       -instance <ccdname> detector name (default $CCDNAME)
       -env <env>          name of workstation online database environment
                           (default $RTAPENV).
                           If value is "FALSE", no action on WS part
                           of NGCOSW is taken
       -lenv <lenv>        name of remote online database environment
                           (default $CCDLENV)
                           If value is "FALSE", no action on LCU part
                           of NGCOSW is taken
       -kill               kill all processes
       -killpulpo          kill also pulpo server


ENVIRONMENT
       CCDNAME  default for camera name (e.g. myccd)
       RTAPENV  default for WS local environment (e.g. myws)
       CCDLENV  default for LCU environment (e.g myngc)


RETURN VALUES
       0 if SUCCESS
       1 if FAILURE


CAUTIONS
       a) .rhosts file on LCU system must contain user and hostname
          where this script runs, since it performs remote shell commands
       b) The "-kill" options should be used with care. By killing
          processes 'blindly', the system could remain in a dangerous
          state. To be used only to recover when the system gets stuck.


EXAMPLES
       > ngcDcsStop -instance myccd -env myws -lenv myngc
           Terminate in a 'soft' way the NGCOSW both at WS and LCU level
           for camera "myccd", WS environment "myws",
           LCU environment "myngc"

       > ngcDcsStop -instance myccd -env myws -lenv myngc -kill
           Terminate in a 'hard' way the NGCOSW both at WS and LCU level
           for camera "myccd", WS environment "myws",
           LCU environment "myngc"

       > ngcDcsStop -instance myccd -env myws -lenv FALSE -kill
           Terminate in a 'hard' way the NGCOSW at WS level only
           for camera "myccd", WS environment "myws"
```

```
       > ngcDcsStop -instance myccd -env FALSE -lenv myngc
           Terminate in a 'soft' way the NGCOSW at LCU level only
           for camera "myccd", LCU environment "myngc"


SEE ALSO
       ngcoDcsStart
```

16.1.5. ngcGetProcNum

```
NAME
       ngcGetProcNum - get process number


SYNOPSIS
       ngcGetProcNum <envName> <procName>


DESCRIPTION
       Utility to retrieve the process number for the indicated
       process from the environment in which it is running.
       It is used by the script ngcoDcsStart.
       <envName>     environment name
       <procName>    process name


RETURN VALUES
       Process number if SUCCESS
       0 if FAILURE


EXAMPLES
       > ngcGetProcNum $RTAPENV ngcoexp_$CCDNAME ; echo $?


SEE ALSO
       ngcoDcsStart
```

16.1.6. ngcoClean

```
NAME
       ngcoClean - Clean oldb envirionment and shared memory


SYNOPSIS
       ngcoClean


DESCRIPTION
       This shell script performs a shutdown of the online database
       environment (defined by $RTAPENV), removing shared memory segments
       and zombie processes.


ENVIRONMENT
       RTAPENV        default for oldb environment


RETURN VALUES
       0 if SUCCESS
       1 if FAILURE


EXAMPLES
       > ngcoClean
```

17. Example of NGCOSW usage

Assuming that we are using an XX system (*xxdcfg* instrument module) and that we want to pass relevant parameters using environment variables "à la FIERASW", in the following example the NGCOSW is started and some exposures are performed.

1. Start NGCOSW from the Instrument Workstation

```
ngcoDcsStart -instance $CCDNAME -env $RTAPENV -lenv $CCDLENV \
             -kill
```

2. Put NGCOSW in STANDBY

```
msgSend $RTAPENV ngcocon_$CCDNAME STANDBY ""
```

3. Put NGCOSW ONLINE

```
msgSend $RTAPENV ngcocon_$CCDNAME ONLINE ""
```

4. Perform periodic wiping

```
msgSend $RTAPENV ngcocon_$CCDNAME STARTWP ""
```

5. Prepare the next exposure (set exposure mode, type, time and binning)

```
msgSend $RTAPENV ngcocon_$CCDNAME SETUP \
  "-function DET.MODE.CURID 1 DET1.EXP.TYPE Normal \
  DET1.WIN1.UIT1 10 DET1.WIN1.BINX 1 DET1.WIN1.BINY 1"
```

6. Start the exposure

```
msgSend $RTAPENV ngcocon_$CCDNAME START ""
```

7. Wait until the exposure has been completed

```
msgSend $RTAPENV ngcocon_$CCDNAME WAIT ""
```

8. Check the exposure status

```
dbRead "<alias>${CCDNAME}:exposure:control.state"
```

9. Prepare the next exposure (change exposure mode, type, time and binning)

```
msgSend $RTAPENV ngcocon_$CCDNAME SETUP \
  "-function DET1.MODE.CURID 3 DET1.EXP.TYPE Dark \
  DET1.WIN1.UIT1 20 DET1.WIN1.BINX 2 DET1.WIN1.BINY 2"
```

10. Prepare the next exposure (define a loop of exposures)

```
msgSend $RTAPENV ngcocon_$CCDNAME SETUP \
  "-function DET1.EXP.NREP 10"
```

11. Start the loop of exposures

```
msgSend $RTAPENV ngcocon_$CCDNAME START ""
```

12. Wait until the last exposure has been completed

```
msgSend $RTAPENV ngcocon_$CCDNAME WAIT ""
```

13. Check the exposure status

```
dbRead "<alias>${CCDNAME}:exposure:control.state"
```

14. Prepare the next exposure (single exposure)

```
msgSend $RTAPENV ngcocon_$CCDNAME SETUP \
  "-function DET1.EXP.NREP 1"
```

15. Start the exposure

```
msgSend $RTAPENV ngcocon_$CCDNAME START ""
```

16. Pause the exposure

```
msgSend $RTAPENV ngcocon_$CCDNAME PAUSE ""
```

17. Modify the exposure time

```
msgSend $RTAPENV ngcocon_$CCDNAME SETUP \
  "-function DET1.WIN1.UIT1 60"
```

18. Continue the exposure

```
msgSend $RTAPENV ngcocon_$CCDNAME CONT ""
```

19. Wait until the exposure has been completed

```
msgSend $RTAPENV ngcocon_$CCDNAME WAIT ""
```

20. Check the exposure status

```
dbRead "<alias>${CCDNAME}:exposure:control.state"
```

21. Stop periodic wiping

```
msgSend $RTAPENV ngcocon_$CCDNAME STOPWP ""
```

22. Exit

```
ngcoDcsStop -kill
```

__oOo__