



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

### Next Generation detector Controller

#### Optical DCS - User Manual

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

Document Issue: 0.1

Date of Issue: 22/09/2006

Prepared by : Name	Date	Signature
Andrea Balestra Claudio Cumani		
Approved by : Name	Date	Signature
Dietrich Baade		
Released by : Name	Date	Signature
Alan Moorwood		



### CHANGE RECORD

ISSUE	DATE	SECTIONS AFFECTED	REASON/INITIATION DOCUMENTS/REMARKS
0.1	22-09-2006	All	First draft, basic information for optical prototype

## Table of contents

1. Introduction .....	6
1.1. Purpose .....	6
1.2. Scope .....	6
1.3. Applicable Documents .....	6
1.4. Reference Documents .....	6
1.5. Abbreviations and Acronyms .....	6
1.6. Glossary .....	7
1.7. Stylistic Conventions .....	7
1.8. Naming Conventions .....	7
1.9. Problem Reporting/Change Request .....	7
2. Installation .....	8
2.1. Software Modules .....	8
2.2. Installation Scripts .....	8
2.3. Installation with pkgin (IWS only) .....	9
2.4. Driver Installation (NGC-LCU only) .....	9
3. Startup/Shutdown Procedure .....	11
3.1. System Configuration .....	11
3.1.1. NGCOSW operational modes .....	13
3.1.2. NGCOSW platform configuration .....	13
3.2. System Startup .....	14
3.2.1. Startup Procedure .....	14
3.2.2. Changes with respect to FIERA .....	16
3.2.3. Configuration Examples .....	17
3.2.3.1 Startup Configuration file <xx>dcfgCONFIG.cfg .....	17
3.2.3.2 Configuration Set <xx>dcfg<NAME>.cfg .....	18
3.3. NGCOSW operational states .....	19
3.3.1. Changes with respect to FIERA .....	20
3.4. System Shutdown .....	20
3.4.1. Changes with respect to FIERA .....	20
4. Command Interface .....	21



4.1. Changes with respect to FIERA.....	24
5. Multiple Instances of DCS .....	25
6. Database Interface .....	26
6.1. Interface between NGCOSW and the external environment .....	26
6.2. Interface between NGCOSW and TCS .....	27
6.3. Image processing interface.....	28
6.4. ngcoDbPublic.h.....	28
6.5. Changes with respect to FIERA.....	28
7. Setup Command.....	29
7.1. Complete Setup .....	31
7.1.1. ngcoSetupComplete.det .....	31
7.2. Changes with respect to FIERA.....	31
8. Exposure Handling .....	32
8.1. Description.....	32
8.1.1. Exposure types .....	32
8.1.2. Exposure status .....	32
8.1.3. Image data.....	33
8.1.4. Exposure Id.....	33
8.1.5. Changes with respect to FIERA.....	34
8.2. Commands .....	34
8.2.1. Changes with respect to FIERA.....	34
8.3. File Formats.....	35
8.3.1. Changes with respect to FIERA.....	35
8.4. Naming Schemes .....	35
8.4.1. Changes with respect to FIERA.....	35
8.5. FITS-Header Contents.....	35
8.5.1. Changes with respect to FIERA.....	36
8.6. Image processing .....	36
8.6.1. Changes with respect to FIERA.....	37
9. Status Command .....	38
9.1. Changes with respect to FIERA.....	38
10. Synchronisation .....	39



11. Error Definitions .....	40
12. Error and Logging Handling .....	42
13. Real-Time Display Interface .....	43
13.1. Changes with respect to FIERA.....	43
14. Special functionalities for Optical Instruments .....	44
14.1. Shutter Control.....	44
14.2. Telemetry Monitoring .....	44
14.2.1. Changes with respect to FIERA.....	44
14.3. Adaptive Optics.....	44



## 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.

DOORS is the tools used to manage requirements, design and User Manual, and Word documents shall be generated from the DOORS version. As the document has been written directly in DOORS, it is advisable to examine it using DOORS itself. The printed (Word) version loses some of the dynamicity of the document.

### 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 cameras, and therefore it uses common terminology in this field (e.g. pixel, binning, readout, frame-transfer chip, etc.) without further explanation.

### 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".

All references shown in this User Manual refer to the list in the "NGC Project Documentation" document.

### 1.4. Reference Documents

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

All references shown in this User Manual refer to the list in the "NGC Project Documentation" document.

### 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

### 2.1. Software Modules

All software modules are under CMM configuration control. The NGC optical detector control software package (**NGCOSW**) includes the base software modules **ngcdrv**, **ngcb**, **ngcpp** and **ngcdcs**

- **ngcdrv** - The device driver for the PCI-Bus back-end card.
- **ngcb** - The NGC base software module containing the driver interface library for communication and DMA, some basic i/o tools, a portable threads- and priority-control implementation and the C++ base classes for general system access. This module also provides a hardware simulation mechanism for the NGC controller.
- **ngcdcs** - The NGC detector control software base module implementing the classes for the NGC hardware modules (sequencer, CLDC, ADC) and the interfaces to the data acquisition.
- **dicNGC** - Dictionary, common to both infrared and optical systems.
- **ngcocon** - The NGC system coordination module for optical applications. This includes all required scripts for system startup and shutdown.
- **ncgoit** - The NGC Image Transfer module for optical applications.
- **ncgoexp** - The NGC Exposure Coordination 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 Scripts

Installation scripts for the software package are provided in the **ngcoarc** software module and work on both the Instrument Workstation (IWS) and on the NGC-LCU.

The **ngcoarc** module does not include the device driver installation on the NGC-LCU, as this needs to be done manually with *root* privileges (system administrator). See section 2.4 for the device driver installation.

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  
cd ngcoarc  
make prepare_installation
```



2. Go to installation directory:

```
cd ../../INSTALL
```

3. Retrieve all needed modules from the archive:

```
./buildFromArchive
```

4. Build and install modules:

```
./buildNGC
```

Steps 3 to 4 can also be done in one step:

```
./buildAll
```

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.

### 2.3. Installation with `pkgin` (IWS only)

To integrate the `NGCOSW` build on the IWS using `ngcoarc` into an instrument build using `pkginBuild` (see [RD31]), insert in the instrument `<xx>ins/INSTALL.cfg` file the following lines:

```
INSTALL.MODULE<N>.NAME "ngcoarc"  
INSTALL.MODULE<N>.VERSION "<version>"  
INSTALL.MODULE<N>.MAKE "prepare_installation start_installation"  
INSTALL.MODULE<N>.SUBPKG "NGCOSW"  
INSTALL.MODULE<N>.OPTIONS "IGNORE_WARNINGS"
```

replacing `<N>` and `<version>` with the appropriate numbers.

### 2.4. Driver Installation (NGC-LCU only)

NGC driver installation is not required on the IWS.

It is required to install the NGC driver on the NGC-LCU only once, at configuration time.



NGC driver installation requires root privileges and must be performed manually. Use the following instructions to retrieve and install the NGC driver module on the NGC-LCU:

```
cmmCopy ngcdrv  
cp -r ngcdrv /tmp  
[login as root ("su -")]  
cd /tmp/ngcdrv/src  
make all install
```

To have the driver module installed at boot-time, the instruction line:

```
/usr/local/bin/ngcdrv_load
```

shall be added to the system file `"/etc/rc.local"`.

	Next Generation Detector Controller Optical DCS - User Manual	VLT-SPE-ESO-13660-4086 Draft 0.1 22/09/2006 Page 10 of 47	
---	--	--	---

**CAUTION:** Use “`rlogin $HOST -l root`”, “`telnet $HOST`”, or “`su -`” to login as root. The frequently used “`su`” (without “`-`”) does not setup a proper root session and the loading of the module (`ngcdrv_load`) may fail with an error message indicating an “*invalid module format*”



## 3. Startup/Shutdown Procedure

### 3.1. System Configuration

NGCOSW usually (see 3.1.2) runs partly on the IWS and partly on the NGC-LCU (PC running a Linux operating system, kernel 2.4 or higher), 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-LCU (see 3.1.2)

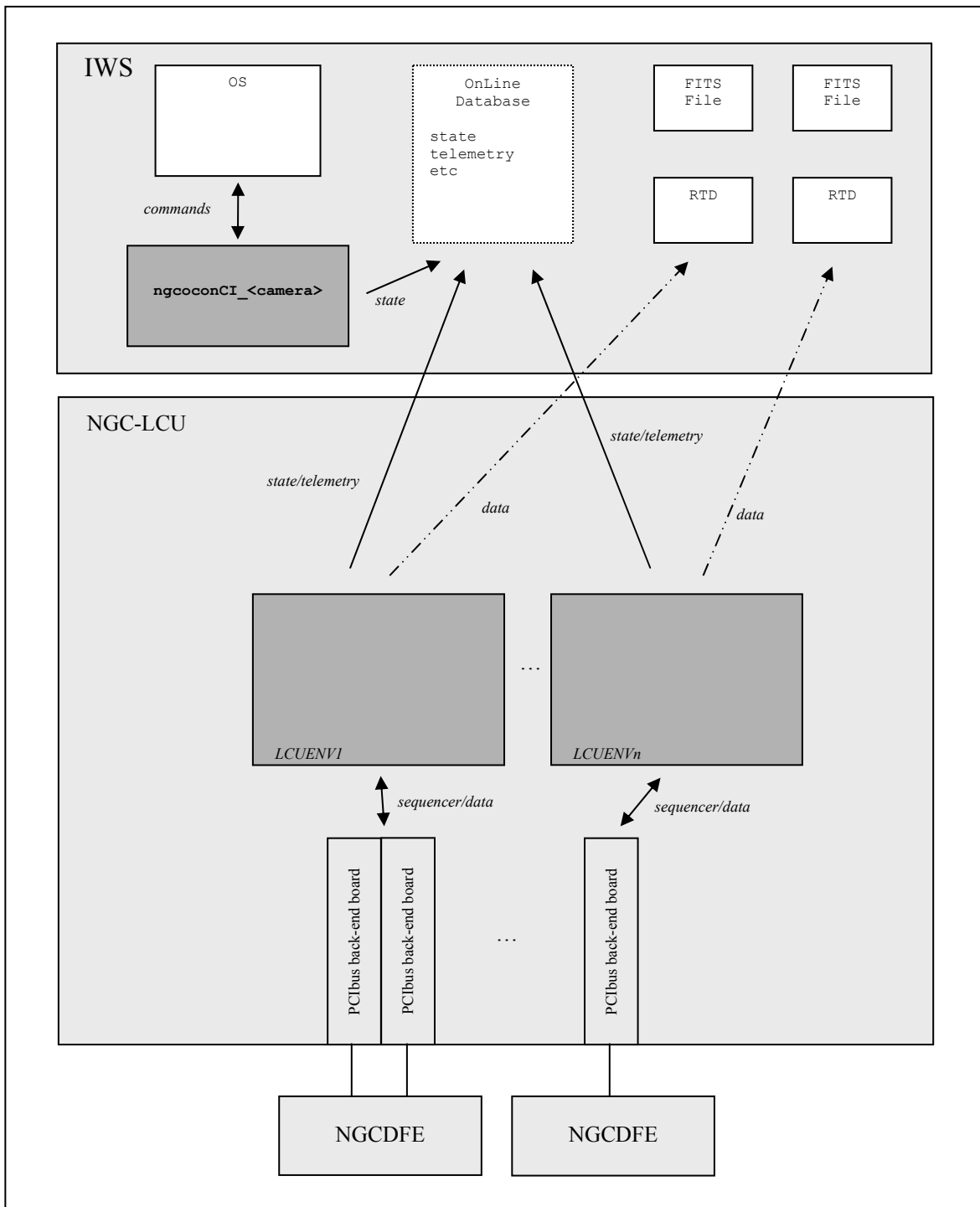


Figure 1 - Optical NGC software, as seen from the User



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 two different modes:

- **Instrument mode**

This is the mode in which the NGCOSW operates when the NGC detector electronics are connected.

- **Simulation mode**

In this mode the NGCOSW simulates the interactions with the NGC detector electronics or other software processes (NGC image transfer, TCS, etc.)

This mode is used by the higher level OS software to test the interface with the NGCOSW during the development phase, when no NGC detector electronics or other software processes are available.

### 3.1.2. NGCOSW platform configuration

NGCOSW can operate in different scenarios:

- **Normal configuration**

In normal operation, the NGCOSW is distributed on both the IWS (where the `IWSENV` online database environment is active) and the NGC-LCU (where the `LCUENV` online database environment is active).

- **Hardware testing configuration**

When operating in the laboratory with no IWS, all the NGCOSW runs on the NGC-LCU (where both the `IWSENV` and the `LCUENV` online database environments are active), controlling the NGC detector electronics.

- **Software testing configuration**

When a Instrument Team is developing the Instrument software and needs to test the interface with the NGCOSW with no NGC-LCU available, all the NGCOSW runs on the IWS (where both the `IWSENV` and the `LCUENV` online database environments are active), simulating the interactions with the NGC detector electronics.

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.

Summarising, these are possible operational scenarios:

- **Normal configuration, instrument mode**

IWSENV runs on the IWS, LCUENV runs on the NGC-LCU. NGC detector electronics are used.

- **Normal configuration, simulation mode**

IWSENV runs on the IWS, LCUENV runs on the NGC-LCU. NGC detector electronics are simulated.

- **Hardware testing configuration, instrument mode**

IWSENV and LCUENV run on the NGC-LCU. NGC detector electronics are used.

- **Hardware testing configuration, simulation mode**

IWSENV and LCUENV run on the NGC-LCU. NGC detector electronics are simulated.

- **Software testing configuration, simulation mode**

IWSENV and LCUENV run on the IWS. NGC detector electronics are simulated.

## 3.2. System Startup

### 3.2.1. Startup Procedure

The startup procedure is based on the common VLT SW configuration tool (“*ctoo*”, [RD75]).

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

- a startup configuration file `<xx>dcfgCONFIG.cfg`
- one or more configuration set(s) `<xx>dcfg<NAME>.cfg`

Each configuration set describes an instance of the NGCOSW, in short FITS format:

Keyword	Type	Description
<b>DET.CON.INSTANCE</b>	String	<p>Defines the instance label for the control server and the database. Used to define the database branch and the appendix “_<i>label</i>” for the Control Coordination Process registered with the CCS environment.</p> <p>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.</p>



Keyword	Type	Description
<b>DET.CON.ENV</b>	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.
<b>DET.CON.LENV</b>	String	Defines the remote online database environment under which the NGC-LCU 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.
<b>DET.CON.SYSCFG</b>	String	Hardware system configuration-file to be loaded by default into this control server instance. Unless an absolute path is given, the file is searched in \$INS_ROOT/\$INS_USER/COMMON/CONFIGFILES If the keyword is not present and not passed as a parameter to the startup script, the value of the \$INS_ROOT and \$INS_USER environment variables is used.
<b>DET.CON.OPMODE</b>	String	Defines the operational mode after starting up. Valid values are "NORMAL", "LCU-SIM" or "HW-SIM". Default is "NORMAL", in case the keyword is not present.
<b>DET.CON.AUTONLIN</b>	Logical	When set to <i>T</i> , the detector system automatically goes to ONLINE at startup. Default is "F", in case the keyword is not present.
<b>DET.CON.GUI</b>	Logical	Launch engineering graphical user interface. Default is "F", in case the keyword is not present.
<b>DET.CON.DICT</b>	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 ...").
<b>DET.CON.XTERM</b>	Logical	Start all (sub-) processes in new terminal. Default is "T", in case the keyword is not present.

Keyword	Type	Description
<b>DET.CON.LOG</b>	Integer	Logging level. Logs system messages in the standard log-file, so that they can be seen in the <i>CCS logMonitor</i> . 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 - Statrup Configuration Keywords

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

The system startup is performed through a startup script by just passing the name of the configuration:

**ngcoDcsStart [config.-set name] [options]**

The startup scripts loads the given startup configuration, 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
-syscfg     - overwrites DET.CON.SYSCFG
-opmode     - overwrites DET.CON.OPMODE
-autonlin   - overwrites DET.CON.AUTOLIN
-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:

```

-warm        - do not reinitialise hardware
  
```

### 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(s).

The script *ngcoDcsStart* replaces the scripts *fedDcsStart* and *fedDcsWarmStart*.

If the startup script of NGCOSW is used without the [config.-set] option, 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;

#
# START CONFIG SET FOR CAMERA 1
# -----
CONFIG.SET1.NAME  "DET1";
CONFIG.SET1.DICT  "NGC_START";
CONFIG.SET1.FILE1 "tcdcfgDET1.cfg";
CONFIG.SET1.PERM1 664; # all
CONFIG.SET1.BACKUP T;
CONFIG.SET1.LOG   T;

#
# START CONFIG SET FOR CAMERA 2
# -----
CONFIG.SET2.NAME  "DET2";
CONFIG.SET2.DICT  "NGC_START";
CONFIG.SET2.FILE1 "tcdcfgDET2.cfg";
CONFIG.SET2.PERM1 664; # all
CONFIG.SET2.BACKUP T;
CONFIG.SET2.LOG   T;
```



```
#
# ctooConfigArchive CONFIG
# -----
CONFIG.ARCHIVE.NAME      "NGCOSW";
CONFIG.ARCHIVE.USER      "";
CONFIG.ARCHIVE.MODULE    "tcdcfg";
CONFIG.ARCHIVE.FILE1     "tcdcfg*.cfg";
CONFIG.ARCHIVE.FILE2     "NGCOSW/*";
CONFIG.ARCHIVE.FILE3     "irtld*.cfg";
CONFIG.ARCHIVE.FILE3     "rtdb*.cfg";
CONFIG.ARCHIVE.FILE3     "RTDB.cfg";
```

```
# ___oOo___
```

### 3.2.3.2 Configuration Set <xx>dcfg<NAME>.cfg

```
#
# Configuration Set
# -----

PAF.HDR.START;
PAF.TYPE          "Configuration";      # Type of PAF
PAF.ID            "@(#) $Id: $";
PAF.NAME          "NGCOSW";             # Name of PAF
PAF.DISC          "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;

# Instance label for server and online database
DET.CON.INSTANCE  "tst";

# HW system configuration file
DET.CON.SYSCFG    "NGCOSW/tst.cfg";

# Startup mode (NORMAL, HW-SIM, LCU-SIM)
DET.CON.OPMODE    "HW-SIM";

# Go online after start
DET.CON.AUTONLIN F;

# Detector system index (DETi.XXX)
DET.CON.DETIDX    1;
```



```
# Dictionaries to load for this detector system
DET.CON.DICT      "";
```

```
# GUI name
DET.CON.GUI      T;
```

```
# ____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 control 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]).

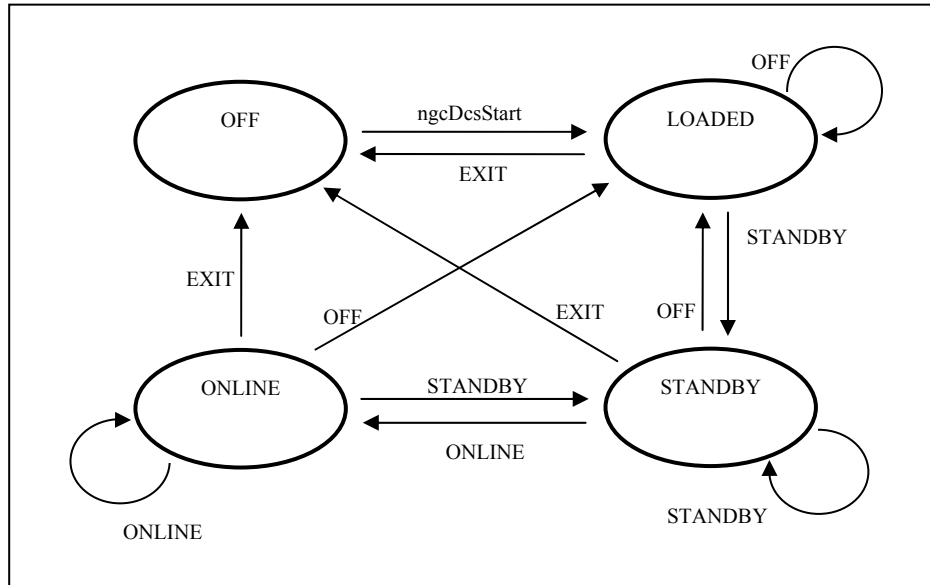


Figure 2 - Operational states and state transitions

### 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 available, which will also wait until the server is actually down (i.e. no more responds to command `PING`):

```
ngcoDcsStop <config.-set name> [options]
```

or

```
ngcoDcsStop -inst <label> [options]
```

The options are:

```
-timeout <ms> - timeout for the shutdown completion
-warm         - do not turn hardware off
```

### 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 `fcDcsStop`.



## 4. Command Interface

The commands which can be issued to the coordination control process are listed in the following table.

Command	Parameters	Format	Description
ABORT	-expold	Integer	Abort exposure with ID <i>&lt;expold&gt;</i> (default last started exposure).
BREAK	<i>none</i>	-	Break execution of current command
CONT	-at	String <YYYY-MM-DD>T<hh:mm:ss> or "now"	Continue a paused exposure at a given time (default <i>now</i> ).
CONFIG	<i>none</i>	-	Read again configuration of the system
DUMP	<i>none</i>	-	Dump the image in memory to disk
END	<i>none</i>	-	End the current exposure(s) and read out the data.
EXIT	<i>none</i>	-	Bring the system to operational state OFF and terminate it.
INIT	-function	String	Initialize the functions contained in the list of arguments (default is <i>all</i> ).
KILL	<i>none</i>	-	Kill this process.
MSGDLOG	<i>none</i>	-	Disable autologging of messages sent or received by the application
MSGELOG	<i>none</i>	-	Enable autologging of messages sent or received by the application



Command	Parameters	Format	Description
OFF	<i>none</i>	-	Bring the system to operational state LOADED.
ONLINE	<i>none</i>	-	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 <i>now</i> ).
PING	<i>none</i>	-	Verify whether this process is able to send or receive messages
SELFTEST	-function	String	Execute a self-test (sw and hw) of the specified function(s).
SETUP	-expold	Integer	Set-up for the exposure with ID <expold> (default next exposure), taking it from file, ore defining it from the parameter function
	-file	String	
	-function	String	
SIM/SIMULAT	-function	String	Put the processes contained in list of arguments into simulation mode.
STANDBY	<i>none</i>	-	Bring the system to operational state STANDBY.
START	-at	String <YYYY-MM-DD>T<hh:mm:ss> or "now"	Start an exposure at a given time (default <i>now</i> ).
STARTLP	-at	String HH:MM:SS.TTT or "now"	Start an infinite loop of repeated exposures at a given time (default <i>now</i> ).
STARTTTL	-period	Integer	Start monitoring of telemetry values.
	-logperiod	Integer	



Command	Parameters	Format	Description
STARTWP	-periodic	Integer	Wipe chip(s) once or periodically.
STATUS	-expold	Integer	Get the status of the functions in the list of arguments
	-function	String	
STOPLP	<i>none</i>	-	Stop an infinite loop of repeated exposures, at the end of the running exposure.
STOPSIM	-function	String	Stop the simulation for the functions contained in the list of arguments.
STOPTL	<i>none</i>	-	Stop monitoring of telemetry values.
STOPWP	<i>none</i>	-	Stop a periodic wipe.
VERBOSE	-on -off	-	Set verbose mode on/off. If <i>-on</i> , the level of the logging is defined by the logging level value (which can be set/modified through the setup keyword <i>DET.CON.LOG</i> )
VERSION	<i>none</i>	-	Return current version of the NGCOSW.



Command	Parameters	Format	Description
WAIT	-expold	Integer	Wait for exposure completion and return exposure status.
	-waitMode	String "Single" / "Global"	<p>-<i>expold</i> indicates the ID of exposure to wait for (see START).</p> <p>0 means: last started exposure.</p> <p>-<i>waitMode</i> can have values:</p> <p><i>Single</i>: wait until the current repetition is completed</p> <p><i>Global</i>: wait until all repetitions are completed.</p> <p>It makes sense only if the setup parameter <i>DET.EXP.NREP</i> is &gt; 1.</p>

Table 2 - 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*).



## 5. Multiple Instances of DCS

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 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

***opMode***. (dbINT32) Camera operational mode

***system.opState*** (dbINT32) System operational state.

***system.setupDirectory*** (dbBYTES128) Setup files directory.

***detector.description*** (dbBYTES32) Detector description.

***detector:chips:chip\_<i>.xLocation*** (dbINT32) Horizontal location of chip in mosaic (n=0,1,...).

***detector:chips:chip\_<i>.yLocation*** (dbINT32) Vertical location of chip in mosaic (n=0,1,...).

***detector:chips:chip\_<i>.xPixels*** (dbINT32) Number of columns.

***detector:chips:chip\_<i>.yPixels*** (dbINT32) Number of rows.

***detector:chips:chip\_<i>.xPixelSize*** (dbINT32) Horizontal pixel size (microns).

***detector:chips:chip\_<i>.yPixelSize*** (dbINT32) Vertical pixel size (microns).

***detector:chips:chip\_<i>.numOutput*** (dbINT32) Number of outputs.

***detector:chips:chip\_<i>.crpix1*** (dbINT32) Value for the CRPIX1 FITS keyword.

***detector:chips:chip\_<i>.crpix2*** (dbINT32) Value for the CRPIX2 FITS keyword.

***detector:chips:chip\_<i>.output\_<j>.id*** (dbINT32) Output Id.

***detector:chips:chip\_<i>.output\_<j>.xLocation*** (dbINT32) Horizontal location of output in chip (n=0,1,...).

***detector:chips:chip\_<i>.output\_<j>.yLocation*** (dbINT32) Vertical location of output in chip (n=0,1,...).

***detector:chips:chip\_<i>.output\_<j>.prescan*** (dbINT32) Number of physical prescan pixels.

***detector:chips:chip\_<i>.output\_<j>.leftToRight*** (dbLOGICAL) Horizontal readout direction, defining the direction of the charge motion towards the output.

***detector:chips:chip\_<i>.output\_<j>.downToUp*** (dbLOGICAL) Vertical readout direction, defining the direction of the charge motion towards the output.

***detector:shutter.description*** (dbBYTES32) Shutter description.

***detector:shutter.available*** (dbLOGICAL) Shutter availability.

***windows>window\_<i>.expType*** (dbINT32). Exposure type.

***windows>window\_<i>.uit<j>*** (dbDOUBLE) Exposure time.

***windows>window\_<i>.xFirst*** (dbINT32) Horizontal coordinate of first pixel.

***windows>window\_<i>.yFirst*** (dbINT32) Vertical coordinate of first pixel.

***windows>window\_<i>.xDim*** (dbINT32) Horizontal dimension.

***windows>window\_<i>.yDim*** (dbINT32) Vertical dimension.



***windows:window\_<i>xBinning*** (dbINT32) Horizontal binning factor for readout.  
***windows:window\_<i>yBinning*** (dbINT32) Vertical binning factor for readout.  
***windows:window\_<i>expRepeat*** (dbINT32) Number of repetitions.  
***windows:window\_<i>expModeIndex*** (dbINT32) Index of the mode chosen for exposure.  
***windows:window\_<i>expFileName*** (dbBYTES32) Name of FITS file where image data are written.  
***windows:window\_<i>:exposure.expId*** (dbINT32) ID of exposure.  
***windows:window\_<i>:exposure.expStatus*** (dbINT32) Status of exposure.  
***windows:window\_<i>:exposure.shutStatus*** (dbINT32) Shutter status.  
***windows:window\_<i>:exposure.timeRem*** (dbDOUBLE). Remaining time.  
***windows:window\_<i>:exposure.readTime*** (dbDOUBLE). Readout time.  
***windows:window\_<i>:exposure.transPercent*** (dbINT32) Percentage of image transferred to WS.  
***windows:window\_<i>:exposure.transTime*** (dbDOUBLE). Time to transfer image to WS.  
***expModes.expMode\_<i>.description*** (dbBYTES32) Description of mode.  
***expModes.expMode\_<i>.wipeSeq*** (dbBYTES32) Sequence for wiping.  
***expModes.expMode\_<i>.wipeVolt*** (dbBYTES32) Voltage table for wiping.  
***expModes.expMode\_<i>.preIntSeq*** (dbBYTES32) Sequence for pre integration.  
***expModes.expMode\_<i>.preIntVolt*** (dbBYTES32) Voltage table for pre integration.  
***expModes.expMode\_<i>.intSeq*** (dbBYTES32) Sequence for integration.  
***expModes.expMode\_<i>.intVolt*** (dbBYTES32) Voltage table for integration.  
***expModes.expMode\_<i>.readSeq*** (dbBYTES32) Sequence for readout.  
***expModes.expMode\_<i>.readVolt*** (dbBYTES32) Voltage table for readout.  
***expModes.expMode\_<i>.outputs*** (Vector of dbINT32) List of the Ids of the outputs used for readout, in the appropriate order.  
***expModes.expMode\_<i>.elAdu*** (Vector of dbDOUBLE) Conversion factor (Electrons per Adu) per output.  
***expModes.expMode\_<i>.ron*** (Vector of dbDOUBLE) readout noise per output.  
***images.imageDirectory*** (dbBYTES128) Directory where images are stored  
***telemetry.enabled*** (dbLOGICAL) Telemetry enabled or not.  
***telemetry.opState*** (dbINT32) Current state of telemetry monitoring.  
***telemetry.current*** (vector of dbDOUBLE) Current telemetry values.  
***periodicWipe.enabled*** (dbLOGICAL) Periodic wipe enabled or not.  
***periodicWipe.opState*** (dbINT32) Current state of periodic wipe.  
***periodicWipe.period*** (dbINT32) Wipe period.

## 6.2. Interface between NGCOSW and TCS

***wcs.ra*** (dbDOUBLE) Centre right ascension in degrees for World Coordinates display (*fcdDB\_WCS\_RA*).  
***wcs.dec*** (dbDOUBLE) Centre declination in degrees for World Coordinates display (*fcdDB\_WCS\_DEC*).



### 6.3. Image processing interface

***windows:window\_<i>:ip.min*** (dbINT32) min pixel value.

***windows:window\_<i>:ip.max*** (dbINT32) max pixel value.

***windows:window\_<i>:ip.rms*** (dbDOUBLE) Rms calculation.

***windows:window\_<i>:ip.xCen*** (dbDOUBLE) Error vector (x component).

***windows:window\_<i>:ip.yCen*** (dbDOUBLE) Error vector (y component).

***windows:window\_<i>:ip.valCen*** (dbDOUBLE) Centroid value.

***windows:window\_<i>:ip.xFwhm*** (dbDOUBLE) Full-width half maximum (x component).

***windows:window\_<i>:ip.yFwhm*** (dbDOUBLE) Full-width half maximum (y component).

***windows:window\_<i>:ip.NumPix*** (dbDOUBLE) Number of pixels above threshold level.

***windows:window\_<i>:ip.BackGnd*** (dbDOUBLE) Background value.

### 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 issued before starting an exposure (unless the new exposure is a perfect copy of the previous one, i.e., no parameter needs to be modified).

Here is a selection of the most important setup keywords (to be completed):

Keyword	Type	Description
<b>DET.READ.CLKIND</b>	Integer	Index of mode used for an exposure (wipe, integrate, readout)
<b>DET.EXP.TYPE</b>	String	Exposure type: <i>Normal</i> one integration, shutter (if any) open <i>Dark</i> one integration, shutter (if any) closed <i>Bias</i> one integration, 0 integration time, shutter (if any) closed <i>Flat</i> one integration, shutter (if any) open <i>Led</i> one integration, shutter (if any) closed, LED light source on <i>LedShut</i> one integration, shutter (if any) open, LED light source on <i>Multiple</i> DET.WIN<i>.NDIT sub-integrations, shutter (if any) open for each integration <i>Burst</i> Multiple frames are read out, shutter (if any) always open
<b>DET.EXP.NREP</b>	Integer	Number of repeated exposures. "0" means "forever"
<b>DET.WIN&lt;i&gt;.UIT1</b>	Double	Integration time (in seconds)
<b>DET.WIN&lt;i&gt;.NDIT</b>	Integer	Number of sub-integrations
<b>DET.WIN&lt;i&gt;.ST</b>	Logical	Window enabled (T) or not (F)
<b>DET.WIN&lt;i&gt;.BINX</b>	Integer	Binning factor along X
<b>DET.WIN&lt;i&gt;.BINY</b>	Integer	Binning factor along Y
<b>DET.WIN&lt;i&gt;.STRX</b>	Integer	First (lower left) window pixel in X direction



Keyword	Type	Description
<b>DET.WIN&lt;i&gt;.STRY</b>	Integer	First (lower left) window pixel in Y direction
<b>DET.WIN&lt;i&gt;.NX</b>	Integer	Number of pixels along X
<b>DET.WIN&lt;i&gt;.NY</b>	Integer	Number of pixels along Y
<b>DET.WIN&lt;i&gt;.MINMAX</b>	Logical	Statistics (minimum/maximum, average) has to be computed on the data for the specified window (T) or not (F)
<b>DET.WIN&lt;i&gt;.CENTROID</b>	String	Kind of centroiding algorithm wanted for the specified window. The following values are supported: <i>none</i> : no centroiding is performed, <i>standard</i> : standard centroiding calculation <i>threshold</i> : threshold based algorithm. background level is defined by the value of <i>DET.WIN&lt;i&gt;.BACKGND</i> . threshold level is defined by the value of <i>DET.WIN&lt;i&gt;.THRMIN</i> .
<b>DET.WIN&lt;i&gt;.BACKGND</b>	Integer	Background level to be used in the centroiding algorithm when parameter <i>DET&lt;i&gt;.WIN&lt;i&gt;.CENTROID</i> is "threshold". It can have any positive value or -1 (the background level is set to the average value over the related window).
<b>DET.WIN&lt;i&gt;.THRMIN</b>	Integer	Threshold level to be used in the centroiding algorithm when parameter <i>DET&lt;i&gt;.WIN&lt;i&gt;.CENTROID</i> is "threshold". It can have any positive value or negative value with special meaning: <i>0&gt;N&gt;-10</i> : threshold level is set to $N * s$ ( $s$ = standard deviation over the related window)
<b>DET.DISPLAY</b>	Integer	Real Time image display -1 no display 0 full frame display 16 bits 1 rapid frame display 16 bits
<b>DET.CHIP&lt;i&gt;.CRPIX&lt;i&gt;</b>	Integer	Reference pixel in <i> direction.
<b>DET.FRAME.SAMPLE</b>	Integer	Image sampling on workstation.

Keyword	Type	Description
<b>DET.READ.NFRAM</b>	Integer	Defines how many sequential image data shall be stored inside a single FITS file. Default is "1"
<b>DET.FRAME.FITSMTD</b>	Integer	Data storage method. Possible values: 0 = 'none' (image is not saved on disk) 1 = 'compressed' 2 = 'uncompressed' 3 = 'both'
<b>DET.FRAME.FILENAME</b>	String	Define the base filename for the data files produced during the exposure

Table 3 - Setup keywords

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. Complete Setup

NGCOSW provides the file `ngcoSetupComplete.det`. It can be used for a general setup of all the keywords relevant to an exposure:

```
msgSend $RTAPENV ngcocon_<label> SETUP \
  "-file ngcoSetupComplete.det"
```

Single keyword values can then be overwritten:

```
msgSend $RTAPENV ngcocon_<label> SETUP \
  "-function DET1.WIN1.BINX 2 DET1.WIN1.BINY 2"
```

### 7.1.1. ngcoSetupComplete.det

To be prepared, once definitively revised the attributes listed above.

## 7.2. Changes with respect to FIERA

The setup keyword `DET.FRAME.FILENAME` replaces `DET.FRAME.FITSUNC` (used also by BOSS, when operating FIERA).



## 8. Exposure Handling

### 8.1. Description

#### 8.1.1. 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).

#### 8.1.2. 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*
- 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*) must 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 data 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>windows>window_<i>:exposure.expStatus.
```

The value of the current exposure status can be:

- 1 NOT ACTIVE
- 2 PENDING (setup)
- 4 INTEGRATING (active)
- 8 PAUSED (active, shutter closed)
- 16 READING (active)
- 32 PROCESSING IMAGE DATA (active)
- 64 TRANSFERRING IMAGE DATA (active)
- 128 COMPLETED (completed successfully)
- 256 FAILED (completed with error)
- 512 ABORTED (completed without data readout, on request)
- 1024 FINITE LOOP OF REPEATED EXPOSURES ACTIVE
- 2048 INFINITE LOOP OF REPEATED EXPOSURES ACTIVE
- 4096 WIPING

### 8.1.3. Image data

Image data are provided by NGCOSW in two ways:

- Raw-data for Real-time display (see section 13)
- FITS files (see section 8.3)

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

```
<alias>windows>window_<i>.expFileName
```

### 8.1.4. Exposure Id

In order to be able to uniquely identify an exposure among those already finished and those running, an identification number (exposure Id) is associated to each exposure.

The exposure Id must be passed to the NGCOSW as a parameter of the command `START`

(as defined in [AD28]).

### 8.1.5. Changes with respect to FIERA

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

Has a new feature, it is 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 is NOT the default behavior).

## 8.2. 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. In this case the generated data file may contain just an intermediate result.

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 0) may be used to wait for a specific state (e.g. *transferring*).

### 8.2.1. Changes with respect to FIERA

NGCOSW implements the same exposure commands of the FIERASW.



### 8.3. 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.

To enable image data storage, the setup parameter `DET.FRAM.FITSMTD` must be set to a value different from 0 (see section 7).

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 from `TBD` (see section 7).

Currently the only format supported for pixels values is 16-bits signed.

Independently from the readout mode used, the complete physical image is stored in one single FITS file. 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]).

#### 8.3.1. Changes with respect to FIERA

NGCOSW implements the same file format of the FIERASW.

### 8.4. 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)

#### 8.4.1. Changes with respect to FIERA

NGCOSW implements the same naming scheme of the FIERASW.

### 8.5. 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.

### 8.5.1. 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.

## 8.6. Image processing

NGCOSW provides facilities to perform real-time image processing on the NGC-LCU while the image data are being readout and before they are transferred to Workstation.

Currently these facilities consist of:

1. Computation of minimum and maximum pixel values in the image (setup parameter `DET<i>.WIN<i>.MINMAX`). The results are stored in the online database in the same sequential order as below:

*images:process>window<i>.ipXMin.* X coordinate of pixel with the minimum value.

*images:process>window<i>.ipYMin.* Y coordinate of pixel with the minimum value.

*images:process>window<i>.ipMinVal.* Minimum pixel value in the frame.

*images:process>window<i>.ipXMax.* X coordinate of pixel with maximum value.

*images:process>window<i>.ipYMax.* Y coordinate of pixel with maximum value.

*images:process>window<i>.ipMaxVal.* Maximum pixel value in the frame.

*images:process>window<i>.ipMean.* Average pixel value in the frame.

2. Computation of rms of the pixel values in the image (setup parameter `DET<i>.WIN<i>.RMS`). The result is stored in the online database in:

*images:process>window<i>.ipRMS.*

3. Calculation of centroiding over an image. It does not implement any pattern recognition algorithm; it simply first subtracts the background level, then applies a threshold (all pixels below the threshold are considered to be 0), and finally computes the centre of gravity of the resulting image. The way that the background and threshold are determined is defined by setup parameter `DET.WIN<i>.CENTROID` (see section 7).

4. Additional processing. An additional function may be invoked after the centroiding algorithm, currently the function IQE is implemented which performs a gaussian fit to determine an accurate centroid and full-width half-maximum (see setup parameter `DET.WIN<i>.IPFUNC`, section 7).



Steps 1-4 can be individually enabled or disabled. If more than one is enabled they are performed sequentially in the order given above. The results of the centroiding and IQE function are stored in the online database attributes below:

*images:process:window<i>.ipXCen*. Difference between X coordinate of centre of gravity and reference position.

*images:process:window<i>.ipYCen*. Difference between Y coordinate of centre of gravity and reference position.

*images:process:window<index>.ipCenVal*. Intensity of the pixel closest to centre of gravity.

*images:process:window<index>.ipBackGnd*. Background intensity.

*images:process:window<index>.ipXFWHM*. X Full-width half-maximum

*images:process:window<index>.ipYFWHM*. Y Full-width half-maximum

*images:process:window<index>.ipNumPix*. Number of pixels over the threshold value

If no object is found during the centroid or IQE function, all attributes are set to 0..

### **8.6.1. Changes with respect to FIERA**

NGCOSW implements the same image processing facilities of the FIERASW.



## 9. Status Command

Here is a selection of the most important keywords (to be completed):

Keyword	Type	Description
<b>DET.READ.AVAIL</b>	String	Returns a list with all read-out modes currently defined. The format of the list is: “<id>:<name> <id>:<name> ...”

Table 4 - Status keywords

### 9.1. Changes with respect to FIERA

The `STATUS` command was not implemented in the FIERASW.



## 10. Synchronisation

TBD



## 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 defines its own errors, which are listed in the following table.

Error	Severity	Description
fcdERR_DB_ROOT	fatal	Wrong root database point. Check if database is running and its consistency.
fcdERR_DB_READ	serious	Failed to read database attribute Check if database is running and its consistency.
fcdERR_DB_WRITE	serious	Failed to write database attribute Check if database is running and its consistency.
fcdERR_OPEN_FILE	serious	Cannot open file. Check if file exists and the write/read/execute attributes of the file itself and the directory where it is (or should be) located.
fcdERR_CMD_WRONG_STATE	warning	Wrong state. Bring the system to the appropriate state, before reissuing the command.
fcdERR_CMD_UNKNOWN	warning	Unknown command.
fcdERR_CMD_NOTIMPLEMENTED	warning	Command not implemented.



Error	Severity	Description
fcdERR_CMD_BUSY	warning	Cannot process command. Wait until process has completed its actual action, before reissuing the command.
fcdERR_CMD_EXECUTE	warning	Failed executing command. Check from logging the reason of the failure.
fcdERR_PARAM_INVALID	warning	Invalid parameter.
fcdERR_PARAM_RANGE	warning	Parameter out of range.
fcdERR_TIMEOUT	serious	Timeout waiting for a response from another process or an external device. Check status of other process or external device.
fcdERR_CCS_CALL	serious	CCS service call failed
fcdERR_FITS_FILE	serious	Error during operations on FITS file
fcdERR_FITS_KEYWORD	warning	Error with FITS keyword
fcdERR_PIXELS	serious	Wrong amount of data received from NGC-LCU. Check consistency of readout sequences.
fcdERR_SHUTTER_STATE	serious	Shutter in bad state. Check shutter.

Table 5 - Errors



## 12. Error and Logging Handling

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

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` whenever the setup parameter `DET.DISPLAY` is set to a value higher than `-1` (see section 7). The setting of this parameter determines the frame where the image will be displayed (currently `rtd` defines frame `ld 0` for the big frame and `4` for the rapid frame, see [RD40]). NGCOSW supports up to 10 different frames `ld` during the same session (10 different images displayed on different frames).

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-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 6.2).

### 13.1. Changes with respect to FIERA

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



## 14. Special functionalities for Optical Instruments

### 14.1. 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. 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 vltscm Exp $"  
# -----  
#  
# Pulpo configuration  
#  
# format is: Pulpo_Unit_Number Full_Device_Path  
1 /dev/ttyc  
2 /dev/ttyd
```

### 14.2. Telemetry Monitoring

NGCOSW provides facilities to monitor telemetry (temperature and pressure) values from the detector.

The telemetry sub-system is enabled and disabled using the commands `STARTTL` and `STOPTL` (see section 4).

The current telemetry values are available in the database attribute `telemetry.current`.

#### 14.2.1. Changes with respect to FIERA

NGCOSW implements the same telemetry interface of the FIERASW.

### 14.3. Adaptive Optics

TBD



## ANNEX A. Example

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

### 1. Start NGCOSW from the Instrument Workstation

```
ngcoDcsStart config.-set xxdcfgCONFIG.cfg -label $CCDNAME \  
-env $RTAPENV -lenv $CCDLENV
```

### 2. Put NGCOSW ONLINE

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

### 3. Perform periodic wiping

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

### 4. Perform Telemetry

```
msgSend $RTAPENV ngcocon_$CCDNAME STARTTTL ""  
dbRead "<alias>${CCDNAME}:telemetry.current"
```

### 5. Stop Telemetry

```
msgSend $RTAPENV ngcocon_$CCDNAME STOPTL ""
```

### 6. Prepare the first exposure (send a complete SETUP to NGCOSW)

```
msgSend $RTAPENV ngcocon_$CCDNAME SETUP \  
"-file ngcoSetupComplete.det"
```

### 7. Start the exposure

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

### 8. Wait until the exposure has been completed

```
msgSend $RTAPENV ngcocon_$CCDNAME WAIT "0"
```

### 9. Prepare the next exposure (change only the binning factor)

```
msgSend $RTAPENV ngcocon_$CCDNAME SETUP \  
"-function DET1.WIN1.BINX 2 DET1.WIN1.BINY 2"
```

### 10. Start the exposure

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

### 11. Wait until the exposure has been completed

```
msgSend $RTAPENV ngcocon_$CCDNAME WAIT "0"
```

### 12. Prepare the next exposure with image statistic

```
msgSend $RTAPENV ngcocon_$CCDNAME SETUP \  
"-function DET1.WIN1.MINMAX T"
```



**13. Start the exposure**

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

**14. Wait until the exposure has been completed**

```
msgSend $RTAPENV ngcocon_${CCDNAME} WAIT "0"
```

**15. Read the image processing results**

```
dbRead "<alias>${CCDNAME}:images:process>window_1.ipMinVal"
```

```
dbRead "<alias>${CCDNAME}:images:process>window_1.ipMaxVal"
```

```
dbRead "<alias>${CCDNAME}:images:process>window_1.ipRMS"
```

**16. Disable image statistic**

```
msgSend $RTAPENV ngcocon_${CCDNAME} SETUP \  
"-function DET1.WIN1.MINMAX F"
```

**17. Prepare the next exposure with image centroid calculation**

```
msgSend $RTAPENV ngcocon_${CCDNAME} SETUP \  
"-function DET1.WIN1.CENTROID threshold"
```

```
msgSend $RTAPENV ngcocon_${CCDNAME} SETUP \  
"-function DET1.WIN1.BACKGND 10"
```

```
msgSend $RTAPENV ngcocon_${CCDNAME} SETUP \  
"-function DET1.WIN1.THRMIN 100"
```

**18. Start the exposure**

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

**19. Wait until the exposure has been completed**

```
msgSend $RTAPENV ngcocon_${CCDNAME} WAIT "0"
```

**20. Read the image centroiding results**

```
dbRead "<alias>${CCDNAME}:images:process>window_1.ipXCen"
```

```
dbRead "<alias>${CCDNAME}:images:process>window_1.ipYCen"
```

**21. Disable image centroid calculation**

```
msgSend $RTAPENV ngcocon_${CCDNAME} SETUP \  
"-function DET1.WIN1.CENTROID none"
```

**22. Put NGCOSW in STANDBY**

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

**23. Exit**

```
ngcoDcsStop -inst &CCDNAME
```



\_\_oOo\_\_